Obfuscated Fuzzy Hamming Distance and Conjunctions from Subset Product Problems

Steven D. Galbraith, Lukas Zobernig

Department of Mathematics, The University of Auckland, {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 [4], a large amount of research has gone into constructing solutions in special cases. One special case that has attracted a lot of attention is evasive functions [3, 40]. 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 [13], logical formulae defined by many conjunctions [10, 11], pattern matching with wild cards [5, 8], root of a polynomial system of low degree [3], compute-and-compare programs [27, 51], and more [40].

More general obfuscation tools are a lot less practical. For example, there are candidates for indistinguishability obfuscation (iO) [25], but the downside of all of these schemes is they are completely infeasible in terms of runtime and are only 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 [24]). 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.

Fuzzy matching has already been treated by many authors and there is a large literature on it. For example, Dodis et al. [19, 20, 21] 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. [12] this leads to "a trade-off between the correction capacity of and the security properties of the scheme". Since the security analysis in [21] is information-theoretic, there is no discussion about the "form" of the leakage, although Dodis et al. prove that the secure sketch preserves *entropic security*. A related issue with secure sketches and

fuzzy extractors is *reusability* [9]. In general, fuzzy extractors are not secure if the same or correlated values are encoded multiple times. 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 [34], based on computational assumptions related to the discrete logarithm problem (we sketch their scheme in Appendix A). Note that Karabina and Canpolat [34] do not mention obfuscation or give a security proof. Wichs and Zirdelis [51] 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.

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 [34]; it is many orders of magnitude more compact than [15, 51]; for full discussion see Section 7.5. Our solution is related to [34], but we think our approach is simpler and furthermore we give a complete security analysis.

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 *(Problem 5.2) is hard, 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, the Hamming distance obfuscator for a uniformly distributed $x \in \{0,1\}^n$ is distributional VBB secure.*

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 secure point function obfuscator (see [6, 40, 50]) and is perfectly correct.

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 $1 \le j \le n$.

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 [5, 8, 10, 11]. See Appendix D for a summary of these. 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 [33, 39, 48, 49]. 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. We will study some applications in more detail in Appendix B.

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 Section 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 [44] where they are used to construct a *number theoretic error correcting code*. Some major differences to our scheme are: in [44] the parameters $(p_i)_{i=1,\ldots,n}$ and q are fixed for all messages; the encoding X is appended to each message. Thus we stress that this application to error correcting codes has no cryptographic security in mind.

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 (continued in Appendix C) briefly discusses our solution to obfuscating conjunctions. Appendix B discusses applications of fuzzy matching, and discusses leakage. Appendix A and Appendix D discuss other solutions to the problem. Appendix F discusses a number-theoretical issue that arises in the security analysis. Appendix G presents the mathematics behind continued fractions in polynomial rings.

2 Hamming Distance

We want to obfuscate the function that computes the distance between two binary vectors and tells if they are close to each other. 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 reflects 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. See Appendix E for formal definitions of these concepts.

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 Figure 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 too unevenly.

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*

$$r \leq \frac{n}{2} - \sqrt{n\lambda \log(2)}. \tag{2.1}$$

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

$$\Pr_{y\leftarrow\{0,1\}^n}[y \in B_{H,r}(x)] \leq \frac{1}{2^\lambda}.$$

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

$$\Pr_{y\leftarrow\{0,1\}^n}[y \in B_{H,r}(x)] = \frac{h_r}{2^n} = \frac{1}{2^n}\sum_{k=0}^{r}\binom{n}{k}.$$

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 $r < n/2$ we finally obtain $h_r/2^n \leq \exp\left(-(n/2-r)^2/n\right)$ and so the probability in question is upper-bounded by $1/2^\lambda$ if Equation (2.1) holds. □

[1] Chernoff bound for binomial distribution tail [1, 16].

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.*

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 Figure 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.

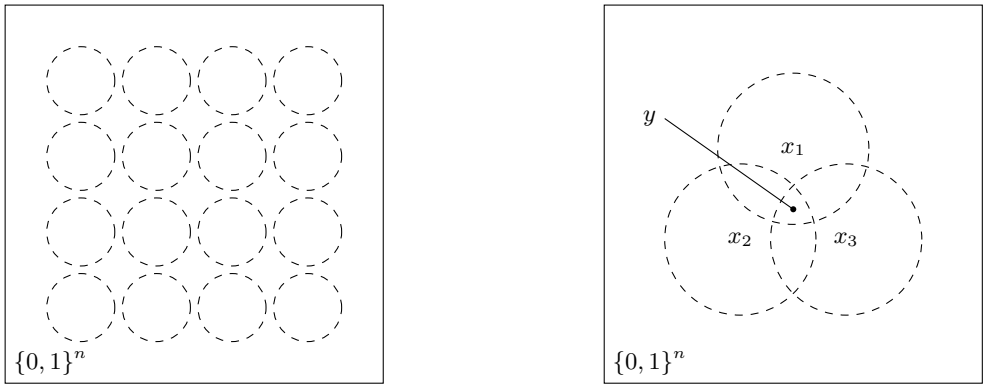


Figure 1: Two example cases of Hamming ball distributions. The left side depicts the ideal distribution of Hamming ball centers. The right one shows what happens if the balls overlap.

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 $X \stackrel{c}{\approx} Y$ 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 *[24, 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

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

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 Figure 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 Section 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 matches 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 [3, 4]). *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 $|\mathcal{O}(P)| \leq \mathrm{poly}(|P|, \lambda)$.*
- *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 black box access to the program P itself.

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

Definition 4.2 (Distributional Indistinguishability [51]). *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 Equation (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 [51]). *For a distribution class $\mathcal{D}$, its* augmentation under predicates $\mathrm{aug}(\mathcal{D})$ *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 $\mathrm{aug}(\mathcal{D})$ 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 [51]). *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).*

Proof. See [11, Lemma 2.2]. □

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

Definition 4.4 (Input Hiding Obfuscator [3]). *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.

Definition 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 prime q such that*

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

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 ($\mathrm{MSP}_{r,n,D}$) *is to find x.*

We also state a decisional version of the problem.

Definition 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 Equation (5.1), and X satisfies Equation (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 the same as $\mathcal{D}_0$, but

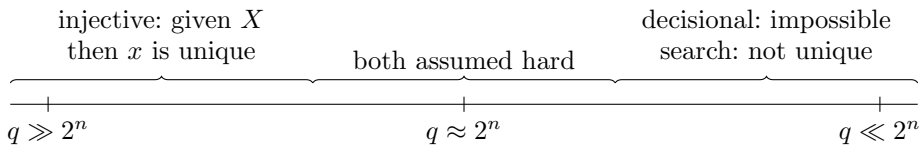
$$X' \leftarrow \mathbb{Z}/q\mathbb{Z}.$$

*Then the (r, n, D)-*decisional distributional modular subset product problem ($\text{D-MSP}_{r,n,D}$) *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 $\text{D-MSP}_{r,n,D}$ (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 expect the values $X = \prod_{i=1}^{n} p_i^{x_i} \mod q$ to be 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. The situation is summarised in the follow 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 Figure 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 [32, 43]. One can obtain the result that a decision oracle for Problem 5.2 in this case can be used to solve the search problem Definition 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.

Theorem 5.1. *Fix $r, n \in \mathbb{N}$ such that $r < n/2$ and let D be the uniform distribution on $\{0,1\}^n$. Let q be prime such that $q \leq 2^n$. Suppose that the distribution $\prod_{i=1}^{n} p_i^{x_i} \mod q$ over $x \leftarrow D$ is within constant multiplicative factor of the uniform distribution on $\mathbb{Z}_q^*$. Suppose $MSP_{r,n,D}$ (Problem 5.1) can be solved in time T. Then there is an algorithm to solve the DLP in $(\mathbb{Z}/q\mathbb{Z})^*$ with expected time $\tilde{O}(nT)$.*

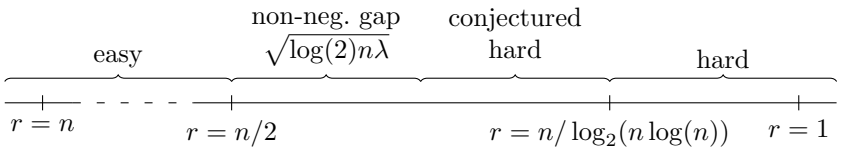
Proof. Let $g, h \in \mathbb{Z}_q^*$ be a DLP instance and let $\mathcal{A}$ be an oracle for Problem 5.1. Let g be a generator of $(\mathbb{Z}/q\mathbb{Z})^*$ so that its order is $M = q - 1$. Choose random $1 < a < q$ and compute $C = g^a \pmod q$. Call $\mathcal{A}$ on C. Due to the assumption in the theorem, with probability bounded below by a constant there is a solution x. Store (a, x). Note that each relation implies a linear relation

$$a \equiv \sum_{i=1}^{n} x_i \log_g(p_i) \pmod M.$$

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 $C = hg^b \pmod q$. Call $\mathcal{A}$ on C to get, with high probability, one more relation. Knowing $\log_g(p_i)$ we now compute $\log_g(h) = -b + \sum_i x_i \log_g(p_i)$. □

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 Figure 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 Section 7.3. Our feeling is that the subset product problem is not really a lattice problem but a number-theoretical problem. As evidence: references [22, 44] 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 [22].

Ultimately, we are making a new assumption based on our experience and knowledge. We hope it will inspire further cryptanalysis.

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. Indeed, even though the discrete logarithm problem is easy for a quantum computer, it would still be necessary to solve a subset sum problem and this is believed to be hard for quantum computers. Hence we believe that both of the computational problems are post-quantum hard.

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 problem. Such a modular subset sum problem may be classified by its density d, see [17, 35]. In our case it is given by $d = n/\log_2(q)$.

There is a polynomial time algorithm for low-density subset sum instances where $d < 0.645$ [35] which was later improved to $d < 0.941$ [17]. This algorithm requires access to a perfect lattice oracle (just using LLL [38] 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 $d \sim n/(r \log_2(n \log n))$, and so for a requested density of $d > 1$ we have to constrain the Hamming distance threshold r to

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

6 Continued Fractions

The background can be found in any number theory textbook, such as [28]. In this section we summarise one key ingredient of our obfuscators. 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 $x = [a_0; a_1, a_2, \dots, a_N]$ 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 any longer: $x = [a_0; a_1, a_2, \dots]$.

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 [31]). *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.*

Proof. The proof of the second half of this theorem is analogous to the proof of a generalised version given by Theorem G.1. □

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 algorithms find 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 [18, 29].

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 Section 7.1. We start with the definition of a safe prime.

Definition 7.1 (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.

The Hamming Distance Obfuscator Let $r, n \in \mathbb{N}$ with $r < n/2$. Choose a random sequence of small distinct primes $(p_i)_{i=1,\dots,n}$ (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 $I \subset \{1, \dots, n\}$ with $|I| \leq r$. The prime q should be sampled to satisfy the bound $q < (1 + o(1)) \max\{p_i\}^r$ as in Equation (5.1). We refer to the discussion regarding Equation (9.3) of 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 Equation (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 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 *point function obfuscator* [6, 40, 50] 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 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 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 1.

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

```
procedure EXECUTE(y ∈ {0,1}^n)
    x' = DECODE(n, (p_i)_{i=1,...,n}, q, X, y)
    if x' =⊥ then return 0
    return Q(x')
end procedure
```

The encoder (Algorithm 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 $((p_i)_{i=1,\dots,n}, q, X)$.

Algorithm 2 Encoding (Obfuscation)

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

The constrained factoring algorithm (Algorithm 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 3 Constrained Factoring

procedure CFACTOR($n, (p_i)_{i=1,\dots,n}, x \in \mathbb{N}$)
  set $F = \{\}$
  for $i = 1,\dots,n$ **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 4) receives as an input an encoding in the form of a triple $((p_i)_{i=1,\dots,n}, q, X)$ 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 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**
    $F \leftarrow$ CFACTOR($n, (p_i)_{i=1,\dots,n}, k$), $F' \leftarrow$ 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

$$E = \prod_{i=1}^{n} p_i^{\epsilon_i} \mod q. \tag{7.3}$$

where $\epsilon_i \in \{-1, 0, 1\}$. First, we note that Equation (7.3) 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 2) and* Decode *(Algorithm 4). For every $r < n/2 \in \mathbb{N}, x \in \{0,1\}^n$, for every*

$$((p_i)_{i=1,\dots,n}, q, X) \leftarrow \textsc{Encode}(r, n, x)$$

and for every $y \in \{0,1\}^n$ such that $d_H(x,y) < r$ it holds that

$$\textsc{Decode}(n, (p_i)_{i=1,\dots,n}, q, X, y) = x.$$

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 Section 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 Section 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

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

Let y be a false accept, which means we have Equation (7.3) with $x_i, y_i \in \{0,1\}$ and $\epsilon_i \in \{-1,0,1\}$. 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 short non-zero vector in the lattice

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

For more details about this lattice see Appendix F.

We now explain why Equation (7.4) implies that Λ is not expected to have a short enough vector to admit a false accept. Equation (F.2) tells us that the expected length of the shortest vector in Λ is $\lambda_1 \sim \sqrt{\frac{n}{2\pi e}} q^{\frac{1}{n}} = \sqrt{\frac{n}{2\pi e}}(n\log(n))^{\frac{r}{n}}$. Hence, to minimise the probability of false accepts, we should choose $r, n \in \mathbb{N}$ such that $\lambda_1 > \sqrt{2n}$, i.e, such that

$$(n\log(n))^{\frac{r}{n}} > 2\sqrt{\pi e}. \tag{7.5}$$

Equation (7.5) 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

$$\lambda_{r,n} = -\log_2\left(\frac{h_r}{2^n}\right)$$

where h_r is defined in Lemma 2.1. We give some example parameter sets along with their bit-security in Figure 2.

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

(a) $n = 512$ ($r_f = 112$, $r_{\mathrm{PQ}} = 44$)

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

(b) $n = 1024$ ($r_f = 204$, $r_{\mathrm{PQ}} = 80$)

Figure 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 $q \sim (n \log n)^r$. The false-acceptance threshold is denoted by r_f (see Equation (7.4)), the post-quantum bound by r_{PQ} (see Equation (5.3)).

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 [49] and [34]. 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 [34] 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. [26] and an LWE based implementation was presented by Chen et al. [15]. 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 minutes. 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 [15] is orders of magnitude worse than in our scheme. Also for comparison, the scheme of Bishop et al [8] (using the optimised variant in [5]) 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.

Due to lack of space we do not give further details. To deflect any criticism of our performance analysis, we remark that none of the works [5, 8, 15] give any performance comparison with previous proposals.

7.6 Polynomial Ring Variant

Here we will discuss a possible 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.

Consider a field k and the ring of polynomials $k[z]$ respectively the field of fractions $k(z)$. One can show that a similar statement to Theorem 6.1 holds; see Appendix G for a summary of the relevant theory and especially Theorem G.1 for the generalised statement.

Let $R = k[z]$, then 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 $q \in R$. For the ground field we may take a finite field of suitable order.

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

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

for all $I \subset \{1,\dots,n\}$ with $|I| \leq r$. To encode $x \in \{0,1\}^n$, publish $X = \prod_{i=1}^n p_i^{x_i} \mod q$ along with the polynomials $(p_i)_{i=1,\dots,n}$ 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 are asserted that this works by Theorem G.1.

Decoding Condition Again, we find that Equation (7.3) can be written as $ED = N + sq$, $N = \prod_{i=1}^n p_i^{\mu_i}$, and $D = \prod_{i=1}^n p_i^{\nu_i}$ for some polynomial $s \in R$. Hence we have that $\left|\frac{E}{q} - \frac{s}{D}\right| = \left|\frac{N}{qD}\right|$. Theorem G.1 asserts us that s/D is a convergent of E/q if $\left|\frac{E}{q} - \frac{s}{D}\right| < \frac{1}{|D|^2}$, i.e. for correctness in the polynomial case we find the relaxed requirement $|ND| < |q|$.

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 $(p_i)_{i=1,\dots,n}$ 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 Section 5.

8.1 Security of the Obfuscator

Here we will focus on showing the security of the Hamming distance 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, Problem 5.1 and Problem 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 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 5 which will generate the first half of the eventual output.

Algorithm 5 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). 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 5 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 Section 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 Section 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 Equation (7.4).

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

Theorem 8.2. *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 for $r > r_f$ (recall Equation (7.4)). Then the Hamming distance obfuscator $\mathcal{O}_H$ is input hiding.*

Proof. Suppose the Hamming weight obfuscator is not input hiding (recall Definition 4.4), i.e., there exists a PPT adversary $\mathcal{A}$ such that

$$\Pr_{P \leftarrow \mathcal{P}_n} [P(\mathcal{A}(\mathcal{O}_H(P))) = 1] > \epsilon(n)$$

for infinitely many n.

Let $((p_i)_{i=1,\ldots,n}, q, X)$ be an instance of Problem 5.1. Since $r > r_f$ 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. We run the adversary on $\mathcal{A}((p_i)_{i=1,\ldots,n}, q, X)$. Then $\mathcal{A}$ outputs a vector $y \in \{0,1\}^n$ that is accepted by P with non-negligible probability; 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 $r > r_f$. Finally, decoding y solves Problem 5.1 with non-negligible probability, as required. □

9 Obfuscating Conjunctions

In this section we describe a new obfuscator for conjunctions. We will also see how it is related to the Hamming distance obfuscator of Section 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 that 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 with $W_x = \{i \,|\, x_i = \star\}$ the set of indices such that x_i is a wildcard. Assume we can choose then 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 $r < n/2$ as mentioned, such that $\prod_{i=1}^{n} p_i^{\sigma(x_i)} > q$.

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. Equation (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^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 uses Algorithm 7 to output 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 Algorithm 8 succeeds for an input $y \in \{0,1\}^n$ and if Q run on the decoded conjunction with its wildcards replaced with 0 outputs 1, else the output is 0. Formally, the obfuscated program is given in Algorithm 6.

Algorithm 6 Obfuscated Program (with embedded data $(p_i)_{i=1,\dots,n}, q, X, Q$)

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

Parameters The same considerations regarding the use of a safe prime and decoding efficiency as in Section 7 apply here. Let us now argue that a safe prime q which is bounded as in Equation (9.1) exists. We use the following heuristic: The density of Sophie Germain primes is given by $\pi_{\text{SG}}(n) \sim 2Cn/\log^2(n)$ for a constant $2C \approx 1.32032$ [45]. An asymptotic inverse is given by $n\log^2(n)$ and so we can expect the $2Cn$-th Sophie Germain prime to be given by $p_{\text{SG}}(n) \sim n\log^2(n)$ (one may check that $\lim_{n\to\infty} \pi_{\text{SG}}(p_{\text{SG}}(n)) = 2C$ and that $\lim_{n\to\infty} p_{\text{SG}}(\pi_{\text{SG}}(n)) = 2C$). 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.

As an example, consider the explicit family of parameters $n = n(\lambda) = 6\lambda$ and $r = r(\lambda) = \lambda$ that admits a security parameter of at least λ. The left plot in Figure 3 exhibits how we can fit $m\log^2(m)$ for some $m \in \mathbb{N}$ between the bounds for an example interval for $\lambda = 30, \dots, 45$. Denote with $\pi_{\text{SG}}[r, r+1]$ the number of Sophie Germain primes in the interval $[((n+1)\log(n+1))^{2r}, (n\log(n))^{2(r+1)}]$ (where the number of Sophie Germain primes in the interval $[a, b]$ is given by $\pi_{\text{SG}}(b) - \pi_{\text{SG}}(a)$). The right plot of Figure 3 shows $\pi_{\text{SG}}[r, r+1]$.

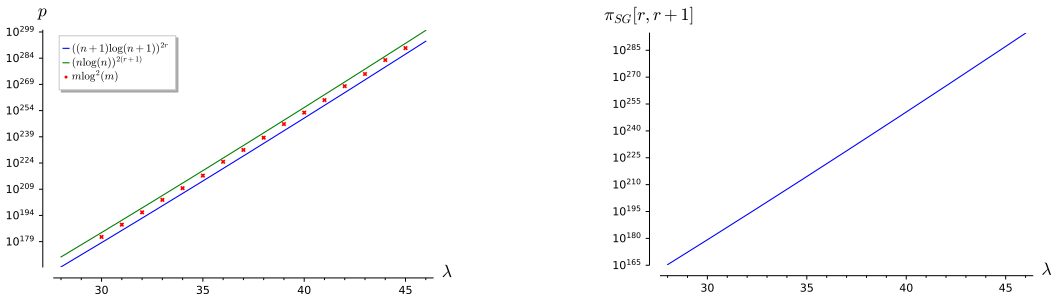


Figure 3: A plot of the heuristic given by Equation (9.3) for the explicit family $n = n(\lambda) = 6\lambda$ and $r = r(\lambda) = \lambda$ and the expected number of Sophie Germain primes in the interval $[((n+1)\log(n+1))^{2r}, (n\log(n))^{2(r+1)}]$.

9.1 Relation to Hamming Distance

Our conjunction obfuscator construction is related to our Hamming distance obfuscator (cf. Section 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 ($0 \le r \le n(q-1)$ for $\mathbb{Z}/q\mathbb{Z}$) using

$$|B_{1,q,r}| = \sum_{k=0}^{r} \left\langle {n \atop k} \right\rangle_q$$

where the *q-nomial triangle* $\left\langle {n \atop k} \right\rangle_q$ is defined as

$$\left\langle {n \atop k} \right\rangle_q = \sum_{i=0}^{n} (-1)^i \binom{n}{i} \binom{k+n-1-iq}{n-1}.$$

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 [7].

– Finally, an input conjunction $x \in \{0,1,\star\}^n$ needs to be evasive. Assuming a uniform conjunction, this will be the case if

$$\frac{|B_{1,3,r}|}{3^n} < \frac{1}{2^\lambda}$$

for $1/2^\lambda$ negligible.

9.2 Parameter Choices

In Section 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 Section 9.1 that the bit-security is given by

$$\lambda_{r,n} = \min \left\{ n - r, -\log_2 \left(\frac{|B_{1,3,r}|}{3^n} \right) \right\}.$$

On the other hand, the conjunction obfuscators given in [5, 8] allow for a wider range of $r < n - O(\log(n))$ at the cost of assuming a generic group model, see Section D.2. Brakerski et al. [11] give no estimate of encoding size or security parameters for their graded coding scheme based obfuscator, see Section D.1. Chen et al. [15] found experimentally that "The TBO of 32-bit conjunctions is close to being practical, with a total evaluation runtime of 11.6 milliseconds, obfuscation runtime of 5.1 minutes, 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 [5, 8].

9.3 Security

Showing security of the conjunction obfuscator works in essentially the same way as for the Hamming distance obfuscator in Section 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 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 $r > r_f$ (recall Equation (7.4)).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 and whether the VBB property of our obfuscator also allows for reusability as a secure sketch. More speculative open problems include obfuscating fuzzy matching with respect to edit distance or other metrics.

Acknowledgements

We thank Zhijie Li for several corrections and comments. We thank reviewers for suggestions.

References

[1] Alon, N., Spencer, J.H.: The probabilistic method (1992)
[2] Avidan, S., Elbaz, A., Malkin, T., Moriarty, R.: Oblivious image matching. In: Protecting Privacy in Video Surveillance, pp. 49–64. Springer (2009)
[3] Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. In: Theory of Cryptography Conference. pp. 26–51. Springer (2014)
[4] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im) possibility of obfuscating programs. In: Annual International Cryptology Conference. pp. 1–18. Springer (2001)
[5] Bartusek, J., Lepoint, T., Ma, F., Zhandry, M.: New techniques for obfuscating conjunctions. In: Advances in Cryptology – EUROCRYPT 2019. pp. 636–666. Springer (2019)
[6] Bellare, M., Stepanovs, I.: Point-function obfuscation: A framework and generic constructions. In: Theory of Cryptography - 13th International Conference. pp. 565–594 (2016)
[7] Bergum, G.E.: Applications of Fibonacci Numbers Volume 4 Proceedings of 'The Fourth International Conference on Fibonacci Numbers and Their Applications', Wake Forest University, N.C., U.S.A., July 30-August 3, 1990. Springer Netherlands : Imprint : Springer, Dordrecht (1991)
[8] Bishop, A., Kowalczyk, L., Malkin, T., Pastro, V., Raykova, M., Shi, K.: A simple obfuscation scheme for pattern-matching with wildcards. In: CRYPTO 2018, Part III. Lecture Notes in Computer Science, vol. 10993, pp. 731–752. Springer (2018)
[9] Boyen, X.: Reusable cryptographic fuzzy extractors. In: Proceedings of the 11th ACM conference on Computer and communications security. pp. 82–91. ACM (2004)
[10] Brakerski, Z., Rothblum, G.N.: Obfuscating conjunctions. Journal of Cryptology **30**(1), 289–320 (2017)
[11] Brakerski, Z., Vaikuntanathan, V., Wee, H., Wichs, D.: Obfuscating conjunctions under entropic ring LWE. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. pp. 147–156. ACM (2016)
[12] Bringer, J., Chabanne, H., Cohen, G., Kindarji, B., Zemor, G.: Theoretical and practical boundaries of binary secure sketches. IEEE Transactions on Information Forensics and Security **3**(4), 673–683 (2008)
[13] Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: Theory of Cryptography Conference. pp. 72–89. Springer (2010)
[14] Chatterjee, R., Athayle, A., Akhawe, D., Juels, A., Ristenpart, T.: password typos and how to correct them securely. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 799–818. IEEE (2016)
[15] 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
[16] Chernoff, H., et al.: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. The Annals of Mathematical Statistics **23**(4), 493–507 (1952)
[17] Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low-density subset sum algorithms. computational complexity **2**(2), 111–128 (1992)
[18] Dixon, J.D.: The number of steps in the Euclidean algorithm. Journal of number theory **2**(4), 414–422 (1970)
[19] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM journal on computing **38**(1), 97–139 (2008)
[20] Dodis, Y., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 523–540. Springer (2004)
[21] Dodis, Y., Smith, A.: Correcting errors without leaking partial information. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. pp. 654–663. ACM (2005)
[22] Ducas, L., Pierrot, C.: Polynomial time bounded distance decoding near minkowski's bound in discrete logarithm lattices. Designs, Codes and Cryptography (2018)

[23] Frykholm, N., Juels, A.: Error-tolerant password recovery. In: Proceedings of the 8th ACM conference on Computer and Communications Security. pp. 1–9. ACM (2001)
[24] Fuller, B., Reyzin, L., Smith, A.: When are fuzzy extractors possible? In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 277–306. Springer (2016)
[25] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM Journal on Computing **45**(3), 882–929 (2016)
[26] Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pp. 555–564. ACM (2013)
[27] Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017. pp. 612–621. IEEE Computer Society (2017)
[28] Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers. Oxford, fourth edn. (1975)
[29] Hensley, D.: The number of steps in the Euclidean algorithm. Journal of Number Theory **49**(2), 142–182 (1994)
[30] Hoffstein, J., Pipher, J., Silverman, J.: An Introduction to Mathematical Cryptography. Undergraduate Texts in Mathematics, Springer, second edition edn. (2014)
[31] Hurwitz, A.: Über die angenäherte darstellung der irrationalzahlen durch rationale brüche. Mathematische Annalen **39**(2), 279–284 (1891)
[32] Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. Journal of cryptology **9**(4), 199–216 (1996)
[33] Jain, A.K., Nandakumar, K., Nagar, A.: Biometric template security. EURASIP Journal on advances in signal processing **2008**, 113 (2008)
[34] Karabina, K., Canpolat, O.: A new cryptographic primitive for noise tolerant template security. Pattern Recognition Letters **80**, 70–75 (2016)
[35] Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. Journal of the ACM (JACM) **32**(1), 229–246 (1985)
[36] Lang, S.: Algebra, vol. 211. Springer New York (2002)
[37] Lasjaunias, A.: A survey of diophantine approximationin fields of power series. Monatshefte für Mathematik **130**(3), 211–229 (2000)
[38] Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (1982)
[39] Li, Q., Sutcu, Y., Memon, N.: Secure sketch for biometric templates. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 99–113. Springer (2006)
[40] Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 20–39. Springer (2004)
[41] Malagoli, F.: Continued fractions in function fields: polynomial analogues of mcmullen's and zaremba's conjectures. arXiv preprint arXiv:1704.02640 (2017)
[42] Manber, U., Wu, S.: An algorithm for approximate membership checking with application to password security. Inf. Process. Lett. **50**(4), 191–197 (1994)
[43] Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Annual Cryptology Conference. pp. 465–484. Springer (2011)
[44] Naccache, D.: A number-theoretic error-correcting code. In: Innovative Security Solutions for Information Technology and Communications: 8th International Conference, SECITC 2015, Bucharest, Romania, June 11-12, 2015. Revised Selected Papers. vol. 9522, p. 25. Springer (2016)
[45] Shoup, V.: A computational introduction to number theory and algebra. Cambridge university press (2009)
[46] Stein, A.: Baby step-giant step-verfahren in reell-quadratischen kongruenzfunktionenkörpern mit charakteristik ungleich 2 (1992), diploma Thesis, Universität des Saarlandes, Saarbrücken
[47] Stichtenoth, H.: Algebraic function fields and codes, vol. 254. Springer Science & Business Media (2009)
[48] Sutcu, Y., Li, Q., Memon, N.: Protecting biometric templates with sketch: Theory and practice. IEEE Transactions on Information Forensics and Security **2**(3), 503–512 (2007)
[49] Tuyls, P., Akkermans, A.H., Kevenaar, T.A., Schrijen, G.J., Bazen, A.M., Veldhuis, R.N.: Practical biometric authentication with template protection. In: International Conference on Audio-and Video-Based Biometric Person Authentication. pp. 436–446. Springer (2005)

[50] 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)
[51] Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). pp. 600–611. IEEE (2017)

A The Scheme of Karabina and Canpolat

Karabina and Canpolat [34] gave a secure fuzzy matching scheme, that uses a particular subgroup of a finite field that has a representation as an algebraic torus. We briefly describe their scheme here, changing the notation to be consistent with our work and also making some simplifications to the presentation.

Let (r, n) be such that we want to do fuzzy matching on $x \in \{0,1\}^n$ up to Hamming distance $r < n/2$. First Karabina and Canpolat choose $p > 2n$ and $m = 2r$ and work in the subgroup G of $\mathbb{F}^*_{p^{2m}}$ of order $p^m + 1$ (technically they work with a representation of this group as the algebraic torus $T_2(\mathbb{F}_{p^m})$).

To encode a string $x \in \{0,1\}^n$ they choose n elements g_i in a special subset S of G that has size p. It is also important that if $g_i \in S$ then $g_i^{-1} \notin S$. The encoding of x is then

$$X = \prod_{i=1}^{n} g_i^{-2x_i+1}.$$

Given an input $y \in \{0,1\}^n$ one can determine if y is within Hamming distance r of x by computing $Y = \prod_i g_i^{-2y_i+1}$ and then trying to write XY^{-1} as a product of r elements. This is done in [34] using some techniques from discrete logarithms in algebraic tori (namely, Weil descent and reduction to solving a system of polynomial equations).

We remark that there is no difference in security between the encoding $X = \prod_i g_i^{-2x_i+1}$ and the encoding $\tilde{X} = \prod_i g_i^{x_i}$, as one can convert between them: given $\tilde{X}$ one has $X = \tilde{X}^2(\prod_i g_i)^{-1}$; given X one has $\tilde{X} = \sqrt{X(\prod_i g_i)}$, and it is easy to compute square roots in finite fields and to make the correct choice of sign.

Karabina and Canpolat [34] do not discuss obfuscation or give a formal security analysis. Their analysis is for uniform distributions of $x \in \{0,1\}^n$, although they mention in *Remark 1* that in real situations the entropy of x may be lower.

In terms of size, the encoding in their scheme is an element of $\mathbb{F}_{p^m}$ where $p^m \approx (2n)^{2r}$. For our scheme, an encoding is an element of $\mathbb{Z}_q$ where $q \approx 2(n\log(n))^r$. Hence the scheme of Karabina and Canpolat is more compact than ours. On the other hand, their scheme is more complex to understand and implement and the performance is weaker, cf. Section 7.5.

Security Analysis We define two computational problems in order to formalise the security of the scheme. Under the assumption that these problems are hard, the VBB and input hiding security of the obfuscator follows immediately by the same arguments as used to prove Theorem 8.1 and Theorem 8.2.

Definition A.1 (Distributional Torus Subset Product Problem). *Let $r < n \in \mathbb{N}$, D be a distribution over $\{0,1\}^n$. Set $m = 2r$. Given a prime $p > 2n$, the subgroup $G < \mathbb{F}^*_{p^{2m}}$ of order $p^m + 1$, $S \subset G$ as in [34], a sequence $(g_i)_{i=1,\ldots,n}$ with $g_i \in S$, and an element*

$$X = \prod_{i=1}^{n} g_i^{-2x_i+1}$$

for some binary vector $x \leftarrow D$, the distributional torus subset product problem *is to find x.*

Definition A.2 (Decisional Distributional Torus Subset Product Problem). *Let $r < n \in \mathbb{N}$, D be a distribution over $\{0,1\}^n$. Given parameters as in Problem A.1, the* decisional distributional torus subset product problem *is to distinguish between the two distributions on tuples $((g_i), X)$ corresponding to*

$$X = \prod_{i=1}^{n} g_i^{-2x_i+1}$$

for some binary vector $x \leftarrow D$, and uniformly random

$$X' \leftarrow G.$$

B Applications

In this section we will list several example applications of our obfuscated Hamming weight scheme. However, we emphasize that the main motivation for this work is the new obfuscation construction rather than any specific application.

B.1 Biometric Matching

The main application for obfuscated Hamming weight is biometric matching. Some of the best examples of this type of matching are fingerprint and DNA. Let us state the general problem: A party wants to securely identify or grant access to individuals based on biometric features (such as mentioned by comparing fingerprint or DNA information) but should not be able to *see* the actual biometrics of the enrolled users. Often we also ask for *anonymity* in the sense that the enrolled users should not be identifiable. Furthermore, the actual biometrics of the users might change over time (DNA might mutate/degrade) or the measurement of the biometric may be noisy (e.g., due to a smudgy fingerprint scanner or other measurement effects). These requirements imply that the biometric features need to be stored in a cryptographically secure way and that we need a way for *fuzzy matching* against this encoded data.

Dodis et al. [19, 21] introduced the notion of a "secure sketch" which we want to view from the perspective of giving rise to an obfuscation scheme for fuzzy matching. We now survey this work. The first step is to construct a random error-correcting linear code over $\mathbb{F}_2$ of length n with generator matrix G and a secure cryptographic hash function H. To encode a vector $x \in \{0,1\}^n$ one chooses a random vector s (of dimension equal to the dimension of the code) and encodes x as the pair (X, h) where $X = x \oplus Gs$ and $h = H(s)$. Note that the target vector x needs to be of high enough entropy in order to sufficiently hide all information about it in the encoding X. To test whether a vector y is close, one can compute $X \oplus y$, decode the result (using the decoding algorithm for the code C) to find a binary vector s' and test whether $H(s') = h$. If $y = x \oplus e$ then $X \oplus y = e \oplus Gs$ and so the process is correct.

Since a decoding algorithm is part of the secure sketch, an attacker can efficiently compute a parity check matrix P such that $PG = 0$. Then, given X, one can compute $PX = Px$, which is a system of $n-k$ explicit linear equations in the secret vector x. This leaks partial information about x. In [21] it is shown that *semantic security* can be achieved for randomised encodings of vectors x with high enough entropy by using random codes. Their definition of semantically secure encodings is similar to VBB obfuscation. However, we stress that this leakage is not "theoretical" but very concrete linear equations that could potentially be used by an adversary in some practical settings.

However, this leakage imposes constraints on the parameters in the secure-sketch setting. One needs to consider families of codes with efficient decoding algorithms, and the resulting constraints on their parameters. The choice of codes issue of entropy-loss has been investigated by Bringer et al. [12]. For example, binary (n, k, d) Reed-Solomon codes limit the Hamming distance threshold to $r \leq d/2 \leq (n-k)/(2 \log n)$. Here n is the length of a code word, k is the code dimension (which gives the entropy of the scheme), and $d \geq 2r$ is the minimum distance of the code. Tuyls et al. [49] give example parameters for BCH codes: $(n, k, d) = (511, 76, 85)$ and $(511, 40, 95)$. The encodings reveal 435 bits respectively 471 bits as all the entropy stems from the 76 bit respectively 40 bit inputs. The Hamming distance r for these examples is only 42 and 47 respectively, whereas our parameters in Figure 2 for $n = 512$ allow r as large as 155.

It is of course important that the biometric features are distributed with high enough entropy in the sense of Definition 2.5. Entropy of biometric features was studied in [39]. Other secure sketch schemes exhibit *leakage* that lowers the entropy of the underlying features [48]. In short, the leakage from secure sketches combined with constraints on the properties of codes (driven by the need for efficient decoding algorithms in order to have practical solutions) leads to quite strong restrictions on the parameters (r, n).

Specifically, in a code based scheme, encodings leak a number of (random) linear equations constraining the secret. In contrast, our scheme is based on computational hardness rather than information-theoretic hardness, and is free from such leakage. As explained in Section 7, the only explicit leakage is a single linear equation obtained by considering the Legendre symbol.

This enables it to be implemented for a much wider range of parameters. Indeed, it was claimed in [33] that matching of encrypted biometric templates is impossible, but our methods (and those of Karabina and Canpolat [34]) show this claim was too pessimistic. We can reach greater Hamming

distance thresholds r and better security while keeping the target vector size the same. The decoding process is also very efficient in practice, see Section 7.5.

B.2 Spelling Errors in Passwords

A similar application of obfuscated Hamming distance is secure fuzzy password matching. We can consider two situations: user input contains spelling mistakes and we want to accept passwords that are r-close to the real one; a user remembers his password up to some positions and wants to recover it without enrolling a fresh one. These situations have been considered before in the literature [14, 23, 42].

To see why our obfuscation scheme is an ideal solution, remember that the commonly accepted way to store passwords is in the form of a *salted* hash (using a cryptographic hash function) for which it is hard to compute a preimage. This way, even if the password database falls into the hands of an adversary, the cleartext passwords are not compromised.

The downside of storing them in hashed form, is that fuzzy password recovery is impossible. On the other hand, our obfuscation scheme is VBB secure and so automatically fulfills all the required properties that a secure password scheme should have. Additionally, it has the property that passwords can be recovered if a user input is r-close to the stored encoding.

B.3 Other applications

Another application of obfuscated Hamming distance is *oblivious image matching*. Here, "two parties want to determine whether they have images of the same object or scene, without revealing any additional information" [2]. This is an application related to privacy in video surveillance. Note that the treatment of applying our obfuscator in this case is similar to biometric matching and a discussion of image pre-processing and feature extraction is outside the scope of this work.

C Obfuscating Conjunctions

In this section we present further details of the conjunction obfuscator.

C.1 Algorithms

We will now present algorithms that detail the encoding part and the evaluation part of our conjunction obfuscator.

The encoder (Algorithm 7) receives as an input a vector $x \in \{0,1,\star\}^n$ corresponding to a conjunction $\chi : \{0,1\}^n \to \{0,1\}$. It then outputs the encoding represented by a triple $((p_i)_{i=1,\ldots,n}, q, X)$.

Algorithm 7 Encoding

procedure ENCODE($n \in \mathbb{N}; x \in \{0,1,\star\}^n$)
  sample a random sequence of distinct primes $(p_i)_{i=1,\ldots,n}$ from $[(n\log(n))^2, ((n+1)\log(n+1))^2]$
  let $W_x = \{i | x_i = \star\}$
  sample safe prime q such that $\prod_{i\in W_x} p_i < q/2 < \prod_{i\in W_x\cup\{j\}}$ for all $j \in \{1,\ldots,n\} \setminus W_x$
  compute $X = \prod_{i=1}^{n} p_i^{\sigma(x_i)} \mod q$
  return $((p_i)_{i=1,\ldots,n}, q, X)$
end procedure

The evaluator (Algorithm 8) 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 conjunction with the wildcards replaced with 0 if $\chi(y) = 1$. The evaluator further uses Algorithm 3 for constrained factoring.

Algorithm 8 Evaluation

procedure EVALUATE$(n, (p_i)_{i=1,\dots,n}, q \in \mathbb{N};\ X \in \mathbb{Z}/q\mathbb{Z};\ y \in \{0,1\}^n)$
  compute $Y^{-1} = \prod_{i=1}^{n} p_i^{-\sigma(y_i)} \mod q$
  compute $E = XY^{-1} \mod q$
  determine 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 $m = (1, \dots, 1) \in \{0,1\}^n$ the vector of all ones
      for $i = 1, \dots, n$ **do**
        if $p_i \in F \cup F'$ **then** set $m_i = 0$
      end for
      return $y \wedge m$
    end if
  end for
  return $\perp$
end procedure

C.2 Decoding

We once again argue about the correctness of the obfuscator.

Lemma C.1 (Correctness). *Consider the algorithms* ENCODE *(Algorithm 7) and* EVALUATE *(Algorithm 8). For every $r < n/2 \in \mathbb{N}, x \in \{0,1,\star\}^n$ corresponding to the conjunction $\chi : \{0,1\}^n \to \{0,1\}$, for every*

$$((p_i)_{i=1,\dots,n}, q, X) \leftarrow \text{ENCODE}(n, x)$$

and for every $y \in \{0,1\}^n$ such that $\chi(y) = 1$ it holds that

$$\text{EVALUATE}(n, (p_i)_{i=1,\dots,n}, q, X, y) = 1.$$

Proof. We find the requirement $ND < q/2$ analogously to the proof of Lemma 7.1.

The 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 the wildcard positions can the immediately be recovered. This all works assuming $\chi(y) = 1$ 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 $\chi(y) = 0$ then with high probability the factors of N and D will not be unique and/or not contained in $(p_i)_{i=1,\dots,n}$. □

D Existing Conjunction Obfuscators

Conjunction obfuscators have been considered before [5, 8, 10, 11]. Let us quickly remind the reader of these schemes.

D.1 LWE-based

One way to tackle the problem of construction a conjunction obfuscator is to use a *graded encoding scheme* [10]. Brakerski et al. [11] give an explicit construction of such a conjunction obfuscator along with a security reduction to *entropic ring LWE*.

Before we summarise this obfuscation construction, we introduce a derived definition. A *directed encoding scheme* is a variant of a multilinear map. Let $\mathcal{R}$ be a ring such that $\mathcal{R}$ has a notion of *short* elements. Let $m \in \mathbb{N}$. For public keys $a_1, a_2 \in \mathcal{R}^m$, define the encoding of a short element $s \in \mathcal{R}$ along the path $(1 \to 2)$ as a short matrix $C \in \mathcal{R}^{m \times m}$ such that

$$a_1 C = s a_2 + e$$

where $e \in \mathcal{R}^m$ is a short Gaussian error. Two such encodings C_1 and C_2 can be multiplied if the end point of C_1 coincides with the start point of C_2. Encodings along the same path may furthermore be added and subtracted, and we are also able to test whether an encoding is the encoding of $0 \in \mathcal{R}$. This also allows us to test whether two encodings with the same start- and end-points are equal.

If we now wish to obfuscate a conjunction $\chi : \{0,1\}^n \to \{0,1\}$ represented by a vector $x \in \{0,1,\star\}^n$ for $n \in \mathbb{N}$, we first sample a collection of random vectors a_i for all $i = 0, 1, \ldots, n, n+1$ representing the paths $(0 \to 1), \ldots, (n \to n+1)$. Next, we sample random short elements $s_{i,b}, r_{i_b} \in \mathcal{R}$ for all $i = 1, \ldots, n$ and all $b \in \{0,1\}$ such that $s_{i,0} = s_{i,1}$ if $x_i = \star$. We then sample encodings $R_{i,b}$ of $r_{i,b}$ and $S_{i,b}$ of $s_{i,b}$ using a_{i-1} and a_i, respectively. Finally, sample a random short element $r_{n+1} \in \mathcal{R}$ and sample the encodings R_{n+1} of r_{n+1} and S_{n+1} of

$$r_{n+1} \prod_{i=1}^{n} s_{i,x_i}$$

where we set $s_{i,\star} = s_{i,0} = s_{i,1}$ when $x_i = \star$. These two encodings are sampled along the path $(n \to n+1)$ using a_n and a_{n+1}.

The obfuscated program consists of the tuple

$$((a_i)_{i=0,\ldots,n+1}, (S_{i,b})_{i=1,\ldots,n;b\in\{0,1\}}, (R_{i,b})_{i=1,\ldots,n;b\in\{0,1\}}, S_{n+1}, R_{n+1}).$$

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

$$S = \left(\prod_{i=1}^{n} S_{i,y_i}\right) R_{n+1}, \quad R = \left(\prod_{i=1}^{n} R_{i,y_i}\right) S_{n+1}.$$

If $\chi(y) = 1$, then S and R are encodings of the same value along the path $(0 \to n+1)$. Otherwise, if $\chi(y) = 0$, they do not encode the same value with overwhelming probability.

The obfuscated programs consists of $2(2n+1)m^2 + (n+2)m = O(nm^2)$ elements of $\mathcal{R}$. No proper analysis in terms of $\mathcal{R}$ and security parameters is provided though it is apparent that the elements of the base ring respectively the dimension m needs to be quite large. It needs to be large to guarantee a suitable security parameter and to make sure that the accumulated error with respect to the direct encoding scheme is bounded.

Security The security of this scheme is based on the security assumptions about the underlying directed encoding scheme. In this case, the entropic ring LWE assumption says that for random short ring elements $s_1, \ldots, s_n \in R$ and a secret vector $x \in \{0,1\}^n$, the ring LWE assumption holds for the secret $\prod_{i=1}^{n} s_i^{x_i}$ given that x is sampled from a high entropy distribution, cf. [11, Section 3.1].

D.2 Generic Group/DLP-based

Here we summarise the scheme presented by Bishop et al. [8]. Furthermore, a generalisation thereof was given by Bartusek et al. [5].

Suppose we are given a conjunction $\chi : \{0,1\}^n \to \{0,1\}$ represented by a vector $x \in \{0,1,\star\}^n$ for $n \in \mathbb{N}$. To obfuscate this conjunction, select a random prime p and sample $a_1, \ldots, a_{n-1}$ uniformly random from $\mathbb{Z}/p\mathbb{Z}$. These coefficients determine a polynomial

$$f(X) = \sum_{i=1}^{n-1} a_i X^i \in (\mathbb{Z}/p\mathbb{Z})[X].$$

Let now G be a *generic group* with a generator g of prime order $p > 2^n$. The obfuscated program consists of the elements $h_{i,j}$ (for all $i = 1, \ldots, n$ and all $j \in \{0,1\}$) where $h_{i,j} = g^{f(2i+j)}$ if $x_i = j$ or $x_j = \star$. Otherwise $h_{i,j}$ is a randomly sampled element of G. Thus the obfuscated program is given by $2n$ group elements.

To evaluate the obfuscated program on and input $y \in \{0,1\}^n$, compute the *Lagrange interpolation* coefficients

$$C_i = \prod_{j \neq i} \frac{-2j - y_j}{2i + y_i - 2j - y_j}$$

and then evaluate the interpolation polynomial using

$$T = \prod_{i=0}^{n-1} (h_{i,y_i})^{C_i}.$$

If $\chi(y) = 1$, then we have $T = g^0$ and otherwise if $\chi(y) = 0$ then T will be a random element of G with overwhelming probability.

Assuming the size of one generic group element is $\log_2(q)$ (q the order of G), then the size of the obfuscated program is $2n \log_2(q)$ in the case of [8] and $(n+1)\log_2(q)$ for the *dual* scheme of [5].

Security The security proof of this scheme in [8] works in the generic group model to show that the pattern matching with wildcards obfuscator is distributional VBB for $x \in \{0,1,\star\}^n$ if the number of wildcards satisfies $|W_x| < 0.774n$. The dual scheme presented in [5] attains distributional VBB security in the generic group model and replaces the limitation on the number of wildcards with the natural bound $|W_x| < n - \omega(\log(n))$ respectively $|W_x| < n - O(\log(n))$.

E Hamming Weight/Distance/Ball

In this section we give formal definitions of the key concepts Hamming weight, Hamming distance and Hamming ball.

Definition E.1 (Hamming Weight). *Let $x \in \{0,1\}^n$ be a binary vector. Then the* Hamming weight *of x is given by $w_H(x) = |\{i \,|\, x_i \neq 0\}|$.*

Definition E.2 (Hamming Distance). *Let $x, y \in \{0,1\}^n$ be two binary vectors. Then the metric given by*

$$\begin{aligned} d_H : \{0,1\}^n \times \{0,1\}^n &\longrightarrow \mathbb{N}, \\ (x,y) &\longmapsto w_H(x-y) \end{aligned}$$

is called the Hamming distance *between x and y.*

Definition E.3 (Hamming Ball). *A* Hamming ball $B_{H,r}(x) \subset \{0,1\}^n$ *of radius r around a point $x \in \{0,1\}^n$ is given by*

$$B_{H,r}(x) = \{y \in \{0,1\}^n \,|\, d_H(x,y) \leq r\}\,.$$

We denote with $B_{H,r}$ the Hamming ball around an unspecified point.

Remark 3. Note that Definition E.1, Definition E.2, and Definition E.3 make sense also for q*-ary* vectors in $(\mathbb{Z}/q\mathbb{Z})^n$ for some prime q.

F The Relation Lattice

The first half of this section closely follows [22].

Let $n \in \mathbb{N}$ and let $(p_i)_{i=1,\ldots,n}$ be a sequence of distinct primes. Let q be a prime distinct to p_i for all $i = 1, \ldots, n$ and consider the following group morphism

$$\begin{aligned} \phi : \mathbb{Z}^n &\to (\mathbb{Z}/q\mathbb{Z})^*, \\ (x_1, \ldots, x_n) &\mapsto \prod_{i=1}^{n} p_i^{x_i} \mod q. \end{aligned} \tag{F.1}$$

If any of the p_i is a primitive root modulo q, then ϕ is surjective. The kernel of ϕ defines the relation lattice

$$\Lambda = \ker \phi = \left\{ x \in \mathbb{Z}^n \;\middle|\; \prod_{i=0}^{n} p_i^{x_i} = 1 \mod q \right\}.$$

We have that the volume of the lattice is given by $|\operatorname{im} \phi|$ and thus bounded by

$$\operatorname{vol}(\Lambda) \leq q - 1$$

since $\varphi(q) = q - 1$ (φ is *Euler's totient function*). Note that equality holds if and only if $\{p_1, \ldots, p_n\}$ generates $(\mathbb{Z}/q\mathbb{Z})^*$.

If the sequence of primes $(p_i)_{i=1,\ldots,n}$ is sufficiently *random* (and thus the generated lattice is sufficiently *random*), then we may employ the Gaussian heuristic (see [30, Section 7.5.3]) to estimate the size of the shortest lattice vector

$$\lambda_1 \sim \sqrt{\frac{n}{2\pi e}} \operatorname{vol}(\Lambda)^{\frac{1}{n}} \leq \sqrt{\frac{n}{2\pi e}} (q-1)^{\frac{1}{n}}, \tag{F.2}$$

where again equality holds if the p_i generate $(\mathbb{Z}/q\mathbb{Z})^*$.

G Diophantine Approximation of Laurent Series

In this section we will give a proof for the Diophantine approximation theorem in a more general form. Instead of approximating elements in $\mathbb{R}$ with fractions in $\mathbb{Q}$, we will be approximating formal Laurent series over some field k with fractions of polynomials in $k[z]$. This generalisation of continued fractions and the theory of Diophantine approximation from $\mathbb{R}$ to formal Laurent series is an established theory and has been studied many times; for a survey on the matter see for example [37].

Consider the field of fractions $k(z)$. Remember, that a possible valuation on $k(z)$ is given by $\nu(f) = \deg q - \deg p$ for elements $f = p/q$ with $p, q \in k[z]$. This induces a norm on $k(z)$, namely

$$|f| = \begin{cases} c^{-\nu(f)} & , f \neq 0 \\ 0 & , f = 0 \end{cases} \tag{G.1}$$

for some fixed constant $c \in \mathbb{R}$ with $c > 1$. This norm is non-Archimedean, i.e. it satisfies

$$|f + g| \leq \max\{|f|, |g|\} \tag{$\star$}$$

with equality when $f \neq g$ (cf. definition of an *ultrametric*) [36, 47].

Definition G.1 (Formal Laurent Series). *Let k be a field, then the* formal Laurent series *over k are given by*

$$L = k((z^{-1})) = \left\{ \sum_{i \leq N} a_i z^i \,\middle|\, N \in \mathbb{Z}; a_i \in k, i \in (-\infty, N] \right\}.$$

One can show that L is a field, since for a non-zero $\alpha = \sum_{i \leq N} a_i z^i$ with $a_N \neq 0$ there exists its inverse $\alpha^{-1} = \sum_{j \leq -N} a'_j z^j$ with coefficients $a'_{-N} = a_N^{-1}$ and $a'_{j-N} = -a_N^{-1} \sum_{i=N+j}^{N-1} a_i a'_{j-i}$ where $j < 0$ [41].

Furthermore, one can show that L is the completion of $k(z)$ with respect to the norm given by Equation (G.1). The valuation on $k(z)$ can be extended to a valuation on L in a natural way; for $\alpha = \sum_{i \leq N} a_i z^i \in L$ we set $\nu(\alpha) = -N$ and so $|\alpha| = c^N$.

Continued Fractions over L We want to consider continued fractions over L. Given an element $f \in k(z)$, we can define a *continued fraction expansion* of the form

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

with a sequence of polynomials $a_i \in k[z]$. As a shorthand notation we again use $f = [a_0; a_1, \ldots, a_N]$. An element $\alpha \in L$ has a similar expansion, though it is not necessarily finite anymore: $\alpha = [a_0; a_1, a_2, \ldots]$.

Note that continued fractions may be nested:

$$[a_0; a_1, \ldots, a_n, [a_{n+1}; a_{n+2}, \ldots]] = [a_0; a_1, \ldots, a_n, a_{n+1}, a_{n+2}, \ldots].$$

They may also be contracted in the following sense:

$$[a_0; a_1, \ldots, a_n, a_{n+1}, \ldots] = [a_0; a_1, \ldots, \alpha_n]$$

where now $\alpha_n \in L$. We can immediately verify that for the convergents defined by Equation (6.1) the following identities hold:

$$\begin{aligned} [a_0; a_1, \ldots, \alpha_n] &= \frac{\alpha_n h_{n-1} + h_{n-2}}{\alpha_n k_{n-1} + k_{n-2}}, \\ (-1)^n &= h_{n-1} k_n - h_n k_{n-1}. \end{aligned} \tag{G.2}$$

Furthermore, let $\alpha = [a_0; a_1, \ldots, a_n, \alpha_{n+1}]$, then by Equation (G.2) we have

$$\left| \alpha - \frac{h_n}{k_n} \right| = \frac{1}{|k_n(\alpha_{n+1} k_n - k_{n-1})|}. \tag{G.3}$$

Much like in the rational case, we find that the convergents of a *fraction* $p/q \in k(z)$ are exactly produced by the polynomials $s_i, t_i \in k[z]$ (cf. Equation (6.2)) in the steps of the extended Euclidean algorithm applied to the polynomials p and q.

Inspecting degrees yields that for convergents $h_n/k_n \in k(x)$ we have $|k_{n-1}| < |k_n| = |a_1| \cdots |a_n|$ for all $n \geq 0$. Hence, from Equation (G.3), we find for $\alpha = [a_0; a_1, \ldots, a_n, \alpha_{n+1}]$ that

$$\left|\alpha - \frac{h_n}{k_n}\right| = \frac{1}{|\alpha_{n+1} k_n^2|},$$

as $|k_n(\alpha_{n+1}k_n - k_{n-1})| = |\alpha_{n+1}k_n^2|$ by the ultrametric property ($\star$). Now since $|\alpha_n| = |a_n|$ and $|k_n| < |k_{n+1}|$, we arrive at the following useful (in)equality

$$\left|\alpha - \frac{h_n}{k_n}\right| = \frac{1}{|k_n k_{n+1}|} < \frac{1}{|k_n|^2}. \tag{G.4}$$

Equality holds since the metric on L is non-Archimedean. In the case of an ordinary metric we would have to replace $=$ with $<$ in Equation (G.4).

Diophantine Approximation in L Finally, we want to show a statement that can be thought of as a generalisation of Theorem 6.1, albeit in the case that the norm in question is non-Archimedean.

Theorem G.1 ([46]). *Let k be a field. Let L be the formal Laurent series over k. Let $\alpha \in L$ and $p, q \in k[z]$ with $\gcd(p(z), q(z)) = 1$ such that*

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{|q|^2}.$$

Then p/q is a convergent of α.

Proof. Consider the convergents of α. Let $n \in \mathbb{N}$ be the index such that $|k_n| \leq |q| < |k_{n+1}|$. Then by our assumption we have

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{|q|^2} \leq \frac{1}{|q||k_n|},$$

On the other hand, by Equation (G.4), we have

$$\left|\alpha - \frac{h_n}{k_n}\right| = \frac{1}{|k_n k_{n+1}|} < \frac{1}{|q||k_n|}.$$

Hence we find

$$\left|\frac{p}{q} - \frac{h_n}{k_n}\right| < \frac{1}{|q||k_n|} \tag{G.5}$$

by the ultrametric property ($\star$).

Now assume that $p/q \neq h_n/k_n$ for all $i \geq 0$, i.e. $pk_n - qh_n \neq 0$. Then, by inspection of degrees, $|pk_n - qh_n| \geq 1$ for all $n \geq 0$. Thus we end up with the following inequality

$$\left|\frac{p}{q} - \frac{h_n}{k_n}\right| = \frac{|pk_n - qh_n|}{|q||k_n|} \geq \frac{1}{|q||k_n|},$$

a contradiction to Equation (G.5). □