Homomorphic Encryption for Finite Automata

Nicholas Genise (UCSD) Craig Gentry (IBM) Shai Halevi (IBM)
Baiyu Li (UCSD) Daniele Micciancio (UCSD)

February 18, 2019

Abstract

We describe a somewhat homomorphic GSW-like encryption scheme, natively encrypting matrices rather than just single elements. This scheme offers much better performance than existing homomorphic encryption schemes for evaluating encrypted (nondeterministic) finite automata (NFAs). Differently from GSW, we do not know how to reduce the security of this scheme to LWE, instead we reduce it to a stronger assumption, that can be thought of as an inhomogeneous variant of the NTRU assumption. This assumption (that we term iNTRU) may be useful and interesting in its own right, and we examine a few of its properties. We also examine methods to encode regular expressions as NFAs, and in particular explore a new optimization problem, motivated by our application to encrypted NFA evaluation. In this problem, we seek to minimize the number of states in an NFA for a given expression, subject to the constraint on the ambiguity of the NFA.

Keywords. Finite Automata, Inhomogeneous NTRU, Homomorphic Encryption, Regular Expressions.

1 Introduction

Homomorphic encryption (HE) [39] enables computation on encrypted data even without knowing the secret key. Ten years after Gentry described the first scheme capable of supporting arbitrary computations [18], we now have an arsenal of several different schemes and variations, with various capabilities and tradeoffs (see, e.g., [42, 9, 8, 31, 16, 21, 13] for a few examples).

Our original motivation for the current work is the simple example of encrypted virus scan: consider a center that deploys many remote systems, operating in many different environments, and wants to protect them against viruses that it knows about. The center would like to periodically send updated virus signatures to all its systems, and have them scan their systems to check for infections. The virus signatures, however, could be sensitive, perhaps because some of them are not yet widely known and exposing the signatures could tip the hand of the center as it develops countermeasures.

A plausible solution would have the center encrypt the virus signatures, the remote systems could then perform the virus scan on the encrypted signatures, and report the (encrypted) results to the center. The center could then decrypt, and take appropriate actions when infections are detected. As virus signatures usually take the form of many small regular expressions, this application calls for a homomorphic encryption scheme that can quickly test for a match against

many small regular expressions. Equivalently, it should quickly evaluate (many, encrypted) nondeterministic finite automata (NFAs) on a given cleartext file. Notice that this is quite different from, and incomparable to, the DFA computation problem studied in previous works on homomorphic encryption, like [17, 14, 15]. Specifically, nondeterminism aside, the crucial difference is that those works consider the evaluation of a plaintext automaton on an encrypted file. In other words, the roles of the input and the program are reversed. In our motivating application, the problem studied in [17, 14, 15] would correspond to searching for arbitrary (possibly nonregular) patterns, on files described by regular languages, a very unlikely scenario.

Evaluating an encrypted NFA on a cleartext string can be done by computing a product of a single vector (representing the initial state of the NFA) by many matrices (representing the transition matrices of the NFA associated to each input symbol). Namely the operation that we want to support is computing

$$\mathbf{u} := (\prod_{i=1}^{k} M_i) \times \mathbf{v},$$

(with operations over the integers), where the matrices M_i and the vector $\mathbf{v}$ are encrypted.[1] Most of the HE schemes from above can be used to carry out this computation, but none of them is ideal for the job. For practical purposes, the homomorphic schemes that offer the best performance are either the BGV-type schemes (scale-invariant or not), or GSW-type schemes.

BGV-type schemes. These schemes have an advantage that they can use *packed ciphertexts*, where each ciphertext encrypts not just one plaintext element but a vector of them, and each ciphertext operation affects all the elements of the vector simultaneously, cf. [41]. Moreover, they can even be made to support efficient matrix-vector operations, as was demonstrated in [22].[2]

However, for BGV-type schemes it is crucial to keep the computation multiplicative depth to a minimum, which in our case means using a binary multiplication tree. But this means that we have to use matrix-matrix multiplication[3] (rather than the matrix-vector products that are computed in the sequential procedure). This increases the total work (and hence the computation time) by a factor equal to the dimension of these matrices — which must be substantial for security reasons.

GSW-type schemes. A major advantage of GSW-like schemes is the asymmetric noise growth, that makes it possible to handle sequential processing of products [11]. For our purposes, it lets us evaluate the product while performing only matrix-vector multiplications.

While "textbook GSW" can only encrypt individual elements, it is possible to adapt the ciphertext-packing techniques from [41] also to GSW, as long as we have a priori bound on the size of the plaintext vectors that occur in the computation. However porting the matrix-multiplication optimizations from [22] is far from simple, and we expect significant overhead when trying to implement it in practice.

[1] The initial vector $\mathbf{v}$ is not required to be encrypted, as it reveals no information about the automaton. However, the intermediate vectors obtained after each matrix vector multiplication should be kept secret. So, we will need a scheme supporting matrix-vector multiplication where both the matrix and the vector are encrypted.

[2] The techniques in [22] only handle multiplication of plaintext matrices by encrypted vectors, but many of these tools can be adapted to the case of encrypted matrices.

[3] Technically, the nodes on the rightmost path of the tree can use matrix-vector multiplications, but this makes hardly any difference on the efficiency of the overall computation.

In [24], Hiromasa, Abe, and Okamoto proposed a GSW-like FHE scheme that is capable of encrypting square matrices and doing homomorphic matrix addition and multiplication. The HAO15 FHE scheme can be viewed as a matrix extension of the standard GSW-FHE scheme, where the secret key $S = [I| - S']$ consists of a random secret matrix S'. Like in GSW [21], the decryption invariant for a ciphertext C encrypting a message M relative to the secret key S is

$$S \times C = M \times S \times G + E \pmod{q},$$

where E is a low-norm error and G is the "gadget matrix" from [35]. Notice that M and S are both matrices in the matrix-FHE case, whereas in GSW M is a scalar and S is a vector. The GSW security reduction [21] to the learning-with-errors (LWE) problem still applies to the HAO15 scheme, except that an additional circular security assumption is required. Being able to encrypt matrices in an atomic operation and support homomorphic matrix operations makes the HAO15 scheme an interesting candidate to use in our application of homomorphic NFA evaluation. Moreover, as we will show in Section 3.1, the HAO15 scheme with some modification can also encrypt vectors and homomorphically multiply an encrypted matrix by an encrypted vector. However, the HAO15 scheme is not optimal due to overhead in the size of keys and ciphertexts. So we seek to find a better solution that would allow us to scan longer strings with faster execution times in practice.

1.1 Our New HE Scheme

In this work we introduce a new scheme, that can be viewed as another GSW-type encryption for matrices but with a different hardness assumption. (Alternatively, it can be viewed as a variant of the GGH15 graded encoding [19], but with no zero-test parameter.) In addition, our scheme can also encrypt vectors and natively support homomorphic matrix-vector multiplication. Similar to the HAO15 scheme, the decryption invariant in our scheme for a ciphertext $C \leftarrow \mathsf{Enc}_S(M)$ encrypting a matrix M is also $S \times C = MSG + E \pmod{q}$, where E is a low-norm error matrix.[4] Differently from the HAO15 scheme, in our construction we assume that the key S is a square invertible matrix, and so we can express the ciphertext as $C := S^{-1}(M \times S \times G + E) \bmod q$. As a result, both keys and ciphertexts are smaller in our scheme.

The operations of the scheme, and the analysis of the noise development are identical to the GSW scheme, except that here we typically cannot ensure that the plaintext size never grows, and instead must use properties of the application to reason about the plaintext size.

When it comes to security, however, we can no longer use the GSW reduction [21] to the LWE problem. That reduction relies heavily on the scalar M commuting with the vector S, which no longer holds in our case. Instead, we reduce the security of this scheme to a stonger assumption, that can be viewed as an inhomogeneous version of NTRU (or alternatively as an LWE instance with an additional hint).

1.2 The iNTRU Hardness Assumption

Recall that in LWE, we are given two matrices $A, B \in \mathbb{Z}_q^{n \times m}$ $(m > n)$, with A a uniformly random matrix, and need to decide if B is also a uniformly random matrix, or it is chosen as $B = SA + E$ with a uniform $S \in \mathbb{Z}_q^{n \times n}$ and a low-norm $E \in \mathbb{Z}_q^{n \times m}$.

It is easy to see that this problem becomes easy if we are also given a trapdoor for the matrix A, in this case it is even easy to recover the secret matrix S when $B = SA+E$. But what if we are given

[4] As we describe later, we use a slightly different variant to encrypt the vector $\mathbf{v}$.

a trapdoor for the matrix B instead? In this case we do not know of any effective distinguisher, so we assume that the decision problem is still hard.

Once we know a trapdoor for B, we might as well consider the case where B is the gadget matrix G (for which everyone knows a trapdoor). Namely we assume that the following decision problem is hard:

iNTRU. As in LWE, we have the parameters n, m, q, with $m > n \log q$ and $q > m$. The input is a matrix $A \in \mathbb{Z}_q^{n \times m}$, which is either uniform in $\mathbb{Z}_q^{n \times m}$, or is set as $A := S^{-1}(G - E) \bmod q$ (with $S \in \mathbb{Z}_q^{n \times n}$ a random invertible matrix, G the gadget matrix, and E a low-norm matrix). The goal is to decide which is the case.

One can think of the above problem as an inhomogeneous version of NTRU, over matrices, as follows. Recall that in the NTRU cryptosystem [25], the secret key is given by two polynomials (or ring elements) with small coefficients f, g, and the corresponding public key is the product $h = f^{-1} \cdot g$. The NTRU cryptosystem can be proved secure under the assumption that this public key h is pseudorandom, i.e., indistinguishable from a uniformly random polynomial (or ring element) with arbitrary coefficients. We extend this assumption as follows. First, we replace g with a sequence of vectors $g_1, \ldots, g_k$, chosen independently at random, with small coefficients. Then, the assumption is that $f^{-1}g_1, f^{-1}g_2, \ldots, f^{-1}g_k$ is pseudorandom. This is a simple syntactic extension of NTRU (that would allow, for example, the encryption of longer messages), akin to changing some parameter, and not a qualitative change in the security assumption. Next, we add a (known, constant) "shift", replacing each g_i with $(2^{i-1} - g_i)$, and still requiring $f^{-1}(1 - g_1), f^{-1}(2 - g_2), \ldots, f^{-1}(2^{k-1} - g_k)$ to be indistinguishable from uniform. We call this the "inhomogeneous" NTRU assumption. Finally, instead of working over a ring of polynomials of degree n, we replace each $f, g_1, \ldots, g_k$ with a square $n \times n$ random matrix with small entries. Intuitively, moving from polynomial rings (which are commutative) to the ring of matrices, should only make the assumption weaker, though we do not know how to prove a formal relation between the two problems. This last problem is essentially equivalent to the pseudorandomness of $A = S^{-1}(G - E)$, where $E = [E_0 | \ldots | E_k]$ is a random matrix with small entries, and $G = [0 | I | 2I | \ldots | 2^{k-1}I]$ is a constant known matrix. In fact, putting A in Hermite Normal Form [34] "cancels out" the S matrix, and gives a sequence of square matrices $-E_0^{-1}(2I^{i-1}E_i)$, corresponding to the matrix version of our inhomogeneous NTRU problem with $f = -E_0$ and $g_i = E_i$.

1.3 From Regular Expression to NFAs

While our scheme directly supports the evaluation of (encrypted) NFAs, patterns (e.g., virus signatures) are typically, and most conveniently, represented by regular expressions. Since the noise growth of our homomorphic encryption scheme depends on the details of the NFA being evaluated and its computations, the conversion of regular expressions to NFA is a critical part of our application. In Section 5 we describe a specific conversion following the method of [12, 3] based on the use of *partial derivatives* of regular expressions, which is both very elegant and efficient. Derivatives of regular expressions [12] are themselves regular expressions and they are defined similarly to formal derivatives of arithmetic expressions, e.g., $\mathrm{d}_a(e_0 + e_1) = \mathrm{d}_a(e_0) + \mathrm{d}_a(e_1)$ for the sum (set union) operation, and $\mathrm{d}_a(e^*) = \mathrm{d}_a(e)e^*$ for exponentiation (Kleene star). Informally, when parsing an input string according to regular expression e, the derivative $\mathrm{d}_a(e)$ represents the part of the input to be expected after reading a first symbol "a". A regular expression e can be converted into an automaton with states labeled by derivatives (modulo a natural equivalence relation on

regular expressions), and transitions of the form $e \xrightarrow{a} \mathrm{d}_a(e)$. A classical result of Brzozowski [12] shows that this produces an automaton with a finite number of states, and, in fact, the minimal DFA of the regular expression. As our homomorphic encryption scheme supports the evaluation of nondeterministic automata, we are interested in the conversion of regular expressions to NFAs, which are potentially much smaller than the equivalent minimal DFA. However, optimizing NFAs in our application is far from trivial. To start with, in stark contrast to the DFA case, minimizing the number of states of an NFA is a PSPACE-complete problem. Moreover, due to noise growth, minimizing the number of states may not even be the right goal for our homomorphic encryption application. We address the first issue by using the *partial derivative* construction of [3], where a partial derivative $\partial_a(e)$ maps an expression e to a *set* of regular expressions (representing possible nondeterministic choices), and in particular $\partial_a(e_0 + e_1) = \partial_a(e_0) \cup \partial_a(e_1)$. This construction results in NFAs that, while not necessarily minimal, have a very small number of states, bounded by the number of alphabet symbols in the input regular expression. In order to bound the noise growth, we show that a simple optimization of the homomorphic NFA evaluation procedure[5] allows to relate the noise growth to the *degree of ambiguity* of the NFA, a standard quantity studied in automata theory, which can be evaluated in polynomial time [44]. We reduce the problem of finding an optimal noise to a variant of NFA minimization problem with bounded ambiguity. Although solving this optimization problem is hard in general, we use techniques of determining ambiguity in Section 5 to explore some tradeoffs between automata size and degree of ambiguity/noise growth.

1.4 Implementation and Performance

We implemented our scheme in C++ using the Number Theory Library (NTL) and describe its details in Section 6. Despite being a simple implementation without optimizations, the on-line pattern matching was exceptionally fast. For example, we could homomorphically match a 65536 bit string in 409 seconds on an encrypted NFA with 1024 states of size 59Mb. Using the same set of parameters, we estimate that an HAO15 implementation can only match up to 16000 bits with a slower execution time and a bigger program size.

1.5 Related Work

As already mentioned, the problem of homomorphically evaluating finite automata or branching programs has been considered before [11, 17, 14, 15], but in a very different context, where the branching program or automaton are publicly known, and the computation is performed homomorphically on an encrypted input string. This is motivated, for example, by applications to FHE bootstrapping, where the program is specified by the publicly known decryption/refreshing procedure, and the input in the (encrypted) secret key. In our setting, the role of the program and input are reversed, and we want the computation to be homomorphic on the automaton, rather than the input string. In the case of general computation, program and input are easily interchanged using a universal Turing machine. But in the case of restricted models of computation, like finite automata, swapping the program and the input results in a completely different problem.

On the relation with other matrix-FHE schemes. As we mentioned earlier, the HAO15 [24] FHE scheme is also capable of encrypting square matrices and doing homomorphic matrix addition

[5] Namely, one can let the initial state vector $\mathbf{v}$ be an "errorless" encryption, because the initial state does not reveal any information about the rest of the automaton.

and multiplication on ciphertexts. In the private-key version of their scheme, the secrete key is $S = [I_r| - S']$ for a secret matrix S', and a matrix $M \in \mathbb{Z}^{r\times r}$ is encrypted as

$$C = \left(\frac{S'A + E}{A}\right) + \left(\frac{MS}{0}\right) \times G \bmod q,$$

where $A \leftarrow \mathbb{Z}_q^{n\times N}$, $E \leftarrow \chi^{r\times N}$ for $N = (n + r)\lceil \log q \rceil$.

It may be tempting to claim that our scheme is the same as the HAO15 scheme due to having the same decryption invariant $SC = MSG + E$. However, these two schemes are not quite identical. The relation between them is very similar to the relation between NTRU and RLWE Regev-like schemes[6], where the difference is that the secret key S is a small square matrix for NTRU (representing multiply-by-s in the ring), whereas the secret key is $S = [I|S']$ in RLWE (where S' represents multiply-by-s' in the ring). Notice that, instead of the Regev invariant, both the HAO15 scheme and our scheme use the GSW-like invariant $SC = MSG + E$ for a small noise matrix E.

More specifically, in our scheme the secret key S is a small square matrix that must be invertible, while in HAO15 we have $S = [I| - S']$ where S' can be any random matrix. Consider the "leveled versions" of the HAO15 scheme and our scheme, in which the secret key matrices $S_0, S_1, \ldots, S_L$ are generated such that S_i is used to encrypt the matrices in level i of the computation. In both schemes it holds that

$$S_i C_i = M S_{i+1} G + E_i.$$

The security of the HAO15 scheme can be reduced to the standard LWE assumption, while our scheme relies on the NTRU-like assumption that we introduce. On the other hand, our scheme is more efficient: we encrypt a matrix $M \in \mathbb{Z}_q^{r\times r}$ in a ciphertext matrix of dimension $\max(r, \lambda)$, whereas the HAO15 scheme requires a dimension $r + \lambda$ ciphertext matrix. One can view our scheme as an NTRU-like variant of the HAO15 scheme (or perhaps an NTRU-like variant of the GSW scheme). From that viewpoint, we introduce in this work the assumption that lets us adapt NTRU to get a GSW-like scheme.

When applied to homomorphically evaluating NFAs, the efficiency advantage of our scheme is more significant. Note that the HAO15 scheme can be used to do homomorphic matrix-vector multiplication as well. But, since we rely on an NTRU-like assumption, the noise bound in our scheme is smaller than the noise bound in the HAO15 scheme, which allows us to homomorphically evaluate longer strings with the same lattice parameters. In terms of the complexity of the homomorphic computation on encrypted NFAs, our scheme runs faster than the HAO15 scheme in practice due to smaller ciphertexts. For more detailed performance comparison, we refer the readers to Appendix C.

Recently, Wang et. al. [43] proposed another matrix-FHE scheme, similar to [8], that has smaller ciphertexts than the HAO15 scheme and can be reduced to the standard LWE assumption. We note that it is possible to perform homomorphic matrix-vector multiplication in their scheme. However, their scheme relies heavily on tensor product to perform homomorphic multiplication, so the security and the complexity of applying their scheme to homomorphic NFA computation is at least on the same level as the HAO15 scheme.

[6]Consider writing both NTRU and RLWE-Regev in matrix form, representing ring elements by their matrices: In both NTRU and RLWE-Regev we have a ciphertext matrix C encrypting a plaintext matrix M relative to the secret matrix S (and plaintext space mod p) if $SC = M + pE \bmod q$.

2 Preliminaries

2.1 Leftover Hash Lemma

A distribution $\mathcal{D}$ over a finite set X is ϵ-uniform if its statistical distance from the uniform distribution is at most ϵ, where the statistical difference between two distributions $\mathcal{D}_1, \mathcal{D}_2$ over a finite domain X is $\frac{1}{2}\sum_{x\in X}|\mathcal{D}_1(x) - \mathcal{D}_2(x)|$. We denote by $x \leftarrow \mathcal{D}$ drawing x from the distribution $\mathcal{D}$, and for a set X we denote by $x \leftarrow X$ drawing x uniformly at random from X. A distribution $\mathcal{D}$ over X has min-entropy k if $\max_{x\in X}\mathcal{D}(x) = 2^{-k}$. A family $\mathcal{H}$ of hash functions from X to Y (with Y a finite set) is said to be 2-universal if for all distinct $x, x' \in X$, $\Pr_{h\leftarrow\mathcal{H}}[h(x) = h(x')] = 1/|Y|$.

Lemma 2.1. *(Leftover Hash Lemma [23]). Let $\mathcal{H}$ be a family of 2-universal hash functions from X to Y, and let $\mathcal{D}$ be a distribution over X with min-entropy k. Suppose that $h \leftarrow \mathcal{H}$ and $x \leftarrow \mathcal{D}$ are chosen independently, then, $(h, h(x))$ is $(\frac{1}{2}\sqrt{|Y|/2^k})$-uniform over $\mathcal{H} \times Y$.*

In this work we apply Lemma 2.1 to the hashing family $\mathcal{H} : \mathbb{Z}_q^m \to \mathbb{Z}_q^n$ defined by

$$\mathcal{H} = \{h_A(\mathbf{v}) = A\mathbf{v} \bmod q\}_{A\in\mathbb{Z}_q^{n\times m}},$$

(which is clearly 2-universal). In particular we use the following corollary:

Corollary 2.2. *Fix the integers k, n, m, m', q, and let $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$ be independent distributions over $\mathbb{Z}_q^m$, all with min-entropy at least k. Let $\mathcal{D}$ be a distribution over matrices $R \in \mathbb{Z}_q^{m\times m'}$, where the i'th column is drawn from $\mathcal{D}_i$. Then the distribution*

$$\{(A, AR \bmod q) : \ A \leftarrow \mathbb{Z}_q^{n\times m}, R \leftarrow D\}$$

is $(\frac{m'}{2}\sqrt{q^n/2^k})$-uniform over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times m'}$.

2.2 Gadget Lattice Sampling

Definitions We consider the norm of a matrix as the length of its longest column in the l_2 norm. A lattice Λ is a discrete subgroup of $\mathbb{R}^n$ (we only consider full-rank, integer lattices). It can be represented by a basis $B \in \mathbb{Z}^{n\times n}$ where the lattice is the set of all integer combinations of B's columns. Let $G = [I|2I|\cdots|2^{l-1}I] \in \mathbb{Z}_q^{n\times nl}$ where $l = \lceil\log_2(q)\rceil$. The G-lattice for a fixed modulus q is $\Lambda_q^\perp(G) = \{\mathbf{x} \in \mathbb{Z}^{nl} : G\mathbf{x} \mod q = \mathbf{0}\}$. The distribution sampled over $\Lambda_q^\perp(G)$ and its integer cosets is the discrete gaussian, a gaussian distribution conditioned on being in the lattice. The probability a sample equals some lattice coset vector $\mathbf{y}$ is proportional to $\exp(-\pi\|\mathbf{y}\|^2/s^2)$ where $s > 0$ is the width of the gaussian (we are only concerned with $\mathbf{0}$-centered distributions). Denote a discrete gaussian of width s on a lattice coset $\Lambda + \mathbf{c}$ as $\mathcal{D}_{\Lambda+\mathbf{c},s}$. We can efficiently sample from $\mathcal{D}_{\Lambda_q^\perp(G)+\mathbf{v},s}$ for any $q \geq 2$ and $s \geq \sqrt{5\ln(2nl+4)/\pi}$ (Theorem 4.1 [35] and Lemma 2.3 [10]). We denote $G^{-1}(\mathbf{v})$ as a discrete gaussian vector $\mathbf{y}$ such that $G\mathbf{y} = \mathbf{v} \mod q$. Further, we assume the width is set just above twice the smoothing parameter (defined below) of the G-lattice.

Concentration and min-entropy. The smoothing parameter [36] of a lattice is needed for our purposes, and it is denoted as $\eta_\epsilon(\Lambda)$ for an $\epsilon > 0$. Informally, this is the smallest width for which a discrete gaussian shares many properties of the continuous gaussian distribution. If B is a basis with minimum Gram-Schmidt norm, we can bound the smoothing parameter $\eta_\epsilon(\Lambda) \leq \|\widetilde{B}\|\omega(\sqrt{\log n})$ for negligible $\epsilon(n) = n^{-\omega(1)}$ [20]. Discrete gaussian samples' l_2 norms are bounded by their width as follows.

Lemma 2.3. *(Lemma 1.5 [5]) Let $\Lambda \subset \mathbb{R}^n$ be a lattice, $r \geq \eta_\epsilon(\Lambda)$ for some $\epsilon \in (0,1)$, and $c \in \mathbb{R}^n$. Then,*

$$\Pr(\|\mathcal{D}_{\Lambda+\mathbf{c},r} \geq r\sqrt{n}\|) \leq 2^{-n} \cdot \left(\frac{1+\epsilon}{1-\epsilon}\right).$$

Therefore, we can efficiently sample a discrete gaussian $G^{-1}(\cdot)$ with length less than $\widetilde{O}(\sqrt{n \log q})$[7] with overwhelming probability, and assume $G^{-1}(\cdot)$'s support is $\mathbb{Z}_q^{nl}$. Since we will be using the leftover hash lemma on discrete gaussian input, we will use the following lemma on the min-entropy of a discrete gaussian.

Lemma 2.4. *(Lemma 2.11 [37]) Let $\Lambda + \mathbf{v} \subset \mathbb{R}^n$ be a lattice coset, $c > 0$, and $s \geq 2^{1+c}\eta_\epsilon(\Lambda)$ for $\epsilon \in (0,1)$. Then for any $\mathbf{y} \in \Lambda + \mathbf{v}$ and for an $\mathbf{x}$ sampled from $\mathcal{D}_{\Lambda+\mathbf{v},s}$,*

$$\Pr(\mathbf{x} = \mathbf{y}) \leq 2^{-n(1+c)} \left(\frac{1+\epsilon}{1-\epsilon}\right).$$

Leftover Hash Lemma with $G^{-1}(\cdot)$. Let $m = nl$, now we can replace the distributions $\mathcal{D}_i$ in Corollary 2.2 with independent discrete gaussian samples $G^{-1}(\mathbf{v})$ (with potential repeats in the coset vector $\mathbf{v}$). Let $R \leftarrow G^{-1}(X)$ in Corollary 2.2 for some $X \in \mathbb{Z}_q^{n \times m'}$ with R's columns sampled independently. Then by the lemmas above, the min-entropy a column of R is at least $n(1+c)\log q - 2$ whenever $G^{-1}(\cdot)$'s width is just above twice $\eta_\epsilon(\Lambda_q^\perp(G))$ for any $\epsilon \in (0, 1/2]$. Say we let $c = \log_q(2)$ in Lemma 2.4. This implies the distribution

$$\{(A, AR \bmod q) : \ A \leftarrow \mathbb{Z}_q^{n \times m}, R \leftarrow G^{-1}(X)\}$$

is $O(m' 2^{-n/2})$-uniform for any $X \in \mathbb{Z}_q^{m \times m'}$.

3 The Schemes

Given an NFA $\mathcal{M}$ of r states over a finite alphabet Σ, we denote by $M_\sigma \in \{0,1\}^{r \times r}$ the transition matrix of $\mathcal{M}$ for each input symbol $\sigma \in \Sigma$, and let $\mathbf{v} \in \{0,1\}^r$ be the vector representing the initial states. To check if a string $w \in \Sigma^*$ is accepted by $\mathcal{M}$, we simply check whether there are any non-zero entries in the vector $(\prod_{i=k}^{1} M_{w_i}) \times \mathbf{v}$ that corresponds to final states. So we need a scheme that can compute matrix-vector multiplication homomorphically over encrypted matrices and vectors.

3.1 The HAO15 matrix-FHE scheme [24]

The FHE scheme from [24] can be extended to support homomorphic matrix-vector multiplication. We first recall the private-key version of the HAO15 scheme, and we then slightly extend it for vector encryption and homomorphic matrix-vector multiplication. For a given security parameter λ, choose lattice parameters n, m, q and a noise distribution χ over $\mathbb{Z}_q$. Let $\ell = \lceil \log q \rceil$, $m = (n+r)\log q$, and $N = (n+r)\ell$. Here we describe a leveled version of the HAO15 scheme that supports multiplication depth up to $k \geq 1$. We abuse notation and have $G = [I|2I|\cdots|2^{l-1}]$ in this subsection.

[7] $\widetilde{O}(\cdot)$ hides poly-logarithmic factors in n.

Key generation. In HAO15, the secret key for level $i \geq 0$ is set to $\mathsf{sk}_i := S_i = [I_r | -S'_i]$, where $A_i \leftarrow \mathbb{Z}_q^{n \times m}$ and $S'_i \leftarrow \chi^{r \times n}$.

Matrix encryption. Given a plaintext matrix $M \in \mathbb{Z}_q^{r \times r}$ and a level $i \geq 0$, to encrypt it for the i-th level of computation, the HAO15 scheme outputs

$$C := \mathsf{HAO.MatEnc}_{\mathsf{sk}_i}(M) = \begin{pmatrix} S'_i A' + E \\ A' \end{pmatrix} + \begin{pmatrix} MS_{i-1} \\ 0_{n \times (n+r)} \end{pmatrix} G \bmod q,$$

where $A' \leftarrow \mathbb{Z}_q^{n \times N}$ and $E \leftarrow \chi^{r \times N}$. For $i = 0$, we consider $S_{-1} = [I_r | 0_{r \times n}]$. Notice that $C \in \mathbb{Z}_q^{(r+n) \times N}$. The decryption procedure is exactly the same as in [24], but we skip it as it is not needed in our application.

Vector encryption and decryption. For a vector $\mathbf{v} \in \mathbb{Z}_q^r$, we can follow the same idea as in the matrix encryption procedure, except that we do not multiply $\mathbf{v}$ by S nor G. We always encrypt a vector using the secret key for the first level:

$$\mathbf{c} := \mathsf{HAO.VecEnc}_{\mathsf{sk}_0}(\mathbf{v}) = \begin{pmatrix} S'_0 \mathbf{a} + \mathbf{e} \\ \mathbf{a} \end{pmatrix} + \begin{pmatrix} \mathbf{v} \\ \mathbf{0}_n \end{pmatrix} \bmod q,$$

where $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \chi^r$. The ciphertext vector has dimension $r + n$. To decrypt a ciphertext vector $\mathbf{c}$ from the i-th level of a computation, output the vector

$$\mathbf{v}' := \mathsf{HAO.VecDec}_{\mathsf{sk}_i}(\mathbf{c}) = \lceil S_i \mathbf{c} \rfloor_2 .$$

Homomorphic operations. To add and multiply two ciphertext matrices C_1 and C_2, we follow [24]: $\mathsf{HAO.Add}(C_1, C_2) = C_1 + C_2$, and $\mathsf{HAO.Mul}(C_1, C_2) = C_1 \times G^{-1}(C_2)$. To multiply a ciphertext matrix C by an encrypted vector $\mathbf{c}$, output

$$\mathsf{HAO.Mul}(C, \mathbf{v}) := C \times G^{-1}(\mathbf{c}).$$

The security of this extended scheme can be proved in the same way as in [24], reducing to the standard $\mathrm{DLWE}_{n,m,q,\chi}$ hardness assumption. It is easy to check that, if C is an encryption of $M \in \{0,1\}^{r \times r}$ for level i and $\mathbf{c}$ is an encryption of $\mathbf{v}$ of level $i-1$, then $S_i \times (C \times G^{-1}(\mathbf{c})) = M\mathbf{v} + \mathbf{e}'$ for some low norm error vector $\mathbf{e}'$. More generally, for any $M_i \in \{0,1\}^{r \times r}$ for $i = 1, \ldots, k$ and $\mathbf{v} \in \mathbb{Z}_q^r$, if $C_i \leftarrow \mathsf{MatEnc}_{\mathsf{sk}_i}(M_i)$ with an error matrix E_i for each i, $\mathbf{c}_0 \leftarrow \mathsf{VecEnc}_{\mathsf{sk}_0}(\mathbf{v})$ with an error vector $\mathbf{e}$, and $\mathbf{c}_i \leftarrow \mathsf{HAO.Mul}(C_i, \mathbf{c_{i-1}})$ for $i = 1, \ldots, k$, then $S_k \times \mathbf{c}_k = (\prod_{j=k}^{1} M_j)\mathbf{v} + \mathbf{e}_k$ where

$$\mathbf{e}_k = E_k G^{-1}(\mathbf{c}_{k-1}) + \sum_{i=2}^{k} (\prod_{j=k}^{i} M_j) E_{i-1} G^{-1}(\mathbf{c}_{i-2}) + (\prod_{j=k}^{1} M_j)\mathbf{e}.$$

The l_∞ norm of $\mathbf{e}_k$ can be bounded by

$$\|\mathbf{e}_k\|_\infty \leq \chi N (1 + k \max_{1 \leq i \leq k} \| \prod_{j=k}^{i} M_j \|_\infty).$$

To successfully decrypt $\mathbf{c}_k$, we require $\|\mathbf{e}_k\|_\infty \leq q/8$ as in [24].

3.2 Our new matrix-HE scheme

To achieve sufficient level of security and a desired capability of homomorphic NFA evaluation, we may need to use a large lattice dimension n in practice. The above extension of the HAO15 scheme seems to be suboptimal with an overhead n in ciphertext dimension. In this section we describe a new matrix homomorphic encryption scheme that supports atomic matrix and vector encryption and matrix-vector multiplication. Our scheme is more efficient in practical applications.

Fix integer parameters n, m, q (to be determined later) and an error distribution χ over $\mathbb{Z}_q$ that outputs with high probability integers of magnitude $\ll q$. For our application we use two variants of (private-key) encryption, one for matrices and the other for vectors. Both variants share a noise-sampling procedure, that takes as input the secret key and another vector (that comes from the plaintext) and outputs a noise vector for use in the encryption (which may be different than just sampling from χ). We denote this procedure by $\mathbf{e} \leftarrow \mathsf{NoiseSamp}(\mathsf{sk}, \mathbf{v})$, and will describe it later in this section.

Key generation. We draw two matrices using χ, a square matrix $S \in \chi^{n \times n}$ and a rectangular $E \in \chi^{n \times m}$ (which is only used in the $\mathsf{NoiseSamp}$ procedure). We insist that S is invertible, and re-sample if it is not (which happens with a small probability $\approx 1/q$). The secret key is $\mathsf{sk} = (S, E)$.

The $\mathsf{NoiseSamp}$ procedure. To prove semantic security of our encryption method, we need a somewhat convoluted procedure for sampling the noise. Specifically, the procedure $\mathsf{NoiseSamp}((S, E), \mathbf{v})$ begins by sampling $\mathbf{r} \leftarrow G^{-1}(\mathbf{v})$, then outputs $\mathbf{e} := E \times \mathbf{r} \bmod q$.

Basic "encryption" transformation. Underlying both the vector and matrix encryption procedure, is the following "encryption" procedure (in quotes, since it does not have a matching decryption procedure). Given the secret key $\mathsf{sk} = (S, E)$ and a vector $\mathbf{v} \in \mathbb{Z}_q^n$, we draw a noise vector $\mathbf{e} \leftarrow \mathsf{NoiseSamp}(\mathsf{sk}, \mathbf{v})$, then output the "ciphertext"

$$\mathbf{c} := \mathsf{Enc}^*_{\mathsf{sk}}(\mathbf{v}) = S^{-1}(\mathbf{v} + \mathbf{e}).$$

We remark that the low-order bits of $\mathbf{v}$ are lost in this transformation, due the added noise. Still, the "ciphertext" satisfies the property that $S\mathbf{c} \approx \mathbf{v}$, up to the low-norm noise vector $\mathbf{e}$.

Later in the paper we prove that the procedure above provides semantic security for $\mathbf{v}$, under the inhomogeneous NTRU hardness assumption.

Vector encryption and decryption. As with Regev encryption [38], to convert the above to real encryption we just need to multiply $\mathbf{v}$ by a large enough scalar δ so that $\|\mathbf{e}\|_\infty < \delta$ with high probability. Let b be an upper bound on the ℓ_∞ norm of vectors that can be dealt with (which depends on the application), we assume that $b \ll q$ and set $\delta := \lfloor q/b \rfloor$.

To encrypt a vector $\mathbf{v} \in \mathbb{Z}_b^n$ we just set $\mathbf{c} := \mathsf{VecEnc}_{\mathsf{sk}}(\mathbf{v}) = \mathsf{Enc}^*_{\mathsf{sk}}(\delta \cdot \mathbf{v})$. To decrypt we set $\mathbf{u} := S \times \mathbf{c} = \delta \cdot \mathbf{v} + \mathbf{e} \pmod q$, then decode each entry of $\mathbf{u}$ to the nearest multiple of δ. Namely, we decrypt as

$$\mathbf{v} := \mathsf{VecDec}_{\mathsf{sk}}(\mathbf{c}) = \left\lceil \frac{b \cdot (S \times \mathbf{c} \bmod q)}{q} \right\rfloor.$$

Matrix encryption and decryption. Matrix encryption is similar, except that instead of just multiplying by a large scalar, we use the GSW technique of redundant encoding using G.

The "native plaintext space" consists of square matrices $M \in \mathbb{Z}_q^{n\times n}$. To encrypt M we first compute $M' = M \times G \bmod q$ and let $\mathbf{m}'_j$ be the j'th column of M' ($j = 1, \ldots, m$). Then we set

$$\mathbf{c}_j := \mathsf{Enc}^*_{\mathsf{sk}}(\mathbf{m}'_j), \text{ and } C := \mathsf{MatEnc}_{\mathsf{sk}}(M) = [\mathbf{c}_1|\mathbf{c}_2|\ldots|\mathbf{c}_m].$$

Note that the ciphertext C has the form $C = S^{-1} \times (MG + E')$, where E' is the low-norm matrix consisting of all the noise vectors that were drawn inside of $\mathsf{Enc}^*_{\mathsf{sk}}$. In other words, the property that this ciphertext satisfies is $S \times C \approx M \times G$, up to the low-norm error matrix E'.

In our application we never need to decrypt matrices, but note that we could compute $U := S \times C = MG + E' \pmod q$, and then recover M from U (since E' is low norm and G is the gadget matrix that has a known trapdoor).

3.3 A Leveled NFA-Homomorphic Scheme

Computing a single product chain. To enable homomorphic computation of a product of k matrices by a vector, $(\prod_{i=0}^{k-1} M_i) \times \mathbf{v}$, we choose $k+1$ secret keys as above, $\mathsf{sk}_i = (S_i, E_i)$, $i = 0, 1, \ldots, k$. We then encrypt the vector $\mathbf{v}$ under the last key sk_k, and for $i < k$ we use sk_i to encrypt the matrix $M'_i = M_i \times S_{i+1}$. In other words, we prepare the ciphertexts

$$\mathbf{c} = S_k^{-1} \times (\delta\mathbf{v} + \mathbf{e}) \bmod q, \text{ and } C_i = S_i^{-1} \times (M_i S_{i+1} G + E_i) \bmod q, \ i = 0, 1, \ldots, k-1,$$

where the noise vectors/matrices are all low-norm. To perform the homomorphic computation, we initialize $\mathbf{c}_k := \mathbf{c}$, and then repeatedly set

$$\mathbf{c}_{i-1} := C_{i-1} \times G^{-1}(\mathbf{c}_i) \bmod q,$$

outputting the final vector ciphertext $\mathbf{c}_0$. We now show (by induction) that for every i, the vector ciphertext $\mathbf{c}_i$ is a valid encryption of the plaintext vector $\mathbf{v}_i = (\prod_{j=i}^{k-1} M_j) \times \mathbf{v}$ under the key sk_i. This holds by definition for $\mathbf{v}_k = \mathbf{v}$, so we now assume that it holds for i and show for $i-1$. By assumption we have

$$\mathbf{c}_i = S_i^{-1} \times (\delta\mathbf{v}_i + \mathbf{e}_i),$$

for some low-norm noise vector $\mathbf{e}_i$. Hence we get

$$\begin{aligned}
\mathbf{c}_{i-1} = C_{i-1} \times G^{-1}(\mathbf{c}_i) &= S_{i-1}^{-1} \times (M_{i-1} S_i G + E_{i-1}) \times G^{-1}(\mathbf{c}_i) \\
&= S_{i-1}^{-1} \times \big(M_{i-1} S_i \times \mathbf{c}_i + E_{i-1} \times G^{-1}(\mathbf{c}_i)\big) \\
&= S_{i-1}^{-1} \times \big(M_{i-1} S_i \times S_i^{-1} \times (\delta\mathbf{v}_i + \mathbf{e}_i) + E_{i-1} \times G^{-1}(\mathbf{c}_i)\big) \\
&= S_{i-1}^{-1} \times \big(M_{i-1}(\delta\mathbf{v}_i + \mathbf{e}_i) + E_{i-1} \times G^{-1}(\mathbf{c}_i)\big) \\
&= S_{i-1}^{-1} \times \big(\delta \underbrace{M_{i-1}\mathbf{v}_i}_{\mathbf{v}_{i-1}} + \underbrace{M_{i-1}\mathbf{e}_i + E_{i-1} \times G^{-1}(\mathbf{c}_i)}_{\mathbf{e}_{i-1}}\big).
\end{aligned}$$

Since $\mathbf{e}_i, E_{i-1}$, and $G^{-1}(\mathbf{c}_i)$ are all low norm, then $\mathbf{e}_{i-1}$ will be low norm as long as M_{i-1} is. We conclude that $\mathbf{c}_0 = S_0^{-1}(\delta\mathbf{v}_0 + \mathbf{e}_0) \pmod q$, where the noise term is

$$\mathbf{e}_0 = \big(\prod_{j=0}^{k-1} M_j\big)\mathbf{e} \ + \ \sum_{i=0}^{k-2} \big(\prod_{j=0}^{i} M_j\big) E_{i+1} G^{-1}(\mathbf{c}_{i+2}) \ + \ E_0 G^{-1}(\mathbf{c}) \pmod q. \tag{1}$$

Hence as long as all the products $\prod_{j=0}^{i} M_j$ have low norm, the final noise term $\mathbf{e}_0$ will also have low norm.

Encrypting and evaluating an NFA. To be able to evaluate this NFA on strings of up to k symbols, we set the parameters so that $\delta = \lfloor q/b \rfloor$ is sufficiently larger than $\max_{w \in \Sigma^{\leq k}} \| \prod_{i=0}^{|w|} M_{w_i} \|_\infty$, then choose $k+1$ secret keys sk_i, $i = 0, \ldots, k$. We encrypt the vectors $\mathbf{v}$ under sk_k, and encrypt each of the matrices M_σ under all the other keys. Namely we set

$$\mathbf{c} = \mathsf{VecEnc}_{\mathsf{sk}_k}(\mathbf{v}), \text{ and } C_{\sigma,i} = \mathsf{MatEnc}_{\mathsf{sk}_i}(M_\sigma S_{i+1}), i = 0, 1, \ldots, k-1.$$

Clearly this method provides semantic security for the NFA, so long as the basic "encryption" transformation from above is semantically secure.

To evaluate the encrypted NFA on a k-symbol string $\sigma_1 \sigma_2 \ldots \sigma_k$, we apply the chain-product procedure from above to evaluate homomorphically the product $(\prod_{i=k}^{1} M_{\sigma_i}) \times \mathbf{v}$. Namely we set $\mathbf{c}'_k = \mathbf{c}$ and then $\mathbf{c}'_{i-1} = C_{\sigma_{k-i+1},i-1} \times G^{-1}(\mathbf{c}'_i)$. At the end of the evaluation, we decrypt the final ciphertext $\mathbf{c}'_0$ to $\mathbf{u} = \mathsf{VecDec}_{sk_0}(\mathbf{c}'_0)$ and check if the computation is accepting.

Circular Security for Better Efficiency. As usual, we can improve efficiency by assuming circular security of the encryption. Namely, instead of choosing all the secret keys independently, we choose just a single secret key and use it everywhere. This means that we only need the ciphertexts

$$\mathbf{c} = S^{-1} \times (\delta \mathbf{v} + \mathbf{e}), \text{ and } C_\sigma = S^{-1} \times (M_\sigma S G + E_\sigma) \text{ for each } \sigma \in \Sigma.$$

3.4 The Parameters

To determine the parameters that are needed for certain NFA (or a class of NFAs) on k-symbol strings, we first need an upper bound on the size of the plaintext, specifically

$$B_{\mathsf{ptxt}} \geq \max_{w \in \Sigma^{\leq k}} \| \prod_{i=0}^{|w|} M_{w_i} \|_\infty.$$

(See Section 5 for methods of converting regular expressions to NFAs while keeping this bound small.) Once we have the bound B_{ptxt}, we use it to compute a high probability bound on the expression

$$B^* \geq \| B_{\mathsf{ptxt}} \cdot \mathbf{e} + k \cdot B_{\mathsf{ptxt}} \cdot E \times G^{-1}(\mathbf{c}) \|,$$

where $\mathbf{e}, E$ are noise terms that are output by the $\mathsf{NoiseSamp}$ procedure. This value B^* bounds with high probability the size of the noise that we can get when evaluating the NFA, and so we need to choose $q > B^* \cdot B_{\mathsf{ptxt}}$ (since our plaintext can be as large as B_{ptxt}).

At the same time, we need to set n large enough relative to q to ensure the required security level (say $q < 2^{n/\lambda}$), and $m > O(n \log q)$ (since we rely on the leftover hash lemma). As usual with lattice-based systems, there is a weak circular dependence between these constraints, but it is not hard to find values that satisfy them all.

4 Security Analysis

Below we define (two variants of) the inhomogeneous NTRU problem, one over a ring and one over integer matrices. We describe some properties of this problem, and show that hardness of the matrix variant implies the security of our encryption scheme.

4.1 Inhomogeneous NTRU

We begin with the ring variant of our hardness assumption. Fix a ring R, a modulus q, and an error distribution χ over R, producing with overwhelming probability elements with norm $\ll q$ and $-\chi = \chi$. Denoting $\ell = \lceil \log q \rceil$, the iNTRU distribution with these parameters is defined as follows:

$$\mathsf{iNTRU} = \left\{ \begin{array}{l} \text{draw } s \leftarrow R/qR, \text{ and } e_i \leftarrow \chi, \text{ for } i = 0, \ldots, \ell, \\ \text{set } a_0 := e_0/s \bmod q, \\ \text{and } a_i := (2^{i-1} - e_i)/s \bmod q \text{ for } i = 1, \ldots, \ell, \\ \qquad\qquad \text{output } (a_0, \ldots, a_{\ell-1}) \end{array} \right\}. \tag{2}$$

The inhomogeneous NTRU problem is to distinguish between this distribution and the uniform distribution over $(R/qR)^\ell$.

In the matrix variant of this assumption, the ring elements s, e_i are replaced by n-by-n integer matrices, and the a_i's are similarly replaced with matrices $A_0 := S^{-1} \times E_0$, $A_i := S^{-1} \times (2^i I - E_i)$. In matrix notation, let $m' = n(\ell+1)$ and G' be the gadget matrix[8] $G' = [0|I|2I|4I|, \ldots |2^{\ell-1}I] \in \mathbb{Z}^{n \times m'}$, and let χ be a distribution over $\mathbb{Z}$, producing with overwhelming probability integers of magnitude $\ll q$. The matrix-iNTRU distribution (MiNTRU) with these parameters is defined as follows:

$$\mathsf{MiNTRU} = \left\{ \begin{array}{l} \text{draw } S \leftarrow \mathbb{Z}_q^{n \times n}, \text{ and } E' \leftarrow \chi^{n \times m'}, \\ \quad \text{output } A' := S^{-1} \times (G' - E') \bmod q \end{array} \right\}. \tag{3}$$

As before, the hardness assumption says that MiNTRU is pseudorandom, namely that the matrix A' is indistinguishable from a matrix uniform in $\mathbb{Z}_q^{n \times m'}$.

4.1.1 Small-Secret Inhomogeneous NTRU

Similarly to LWE, here too we can prove that the Inhomogeneous NTRU problem remains hard even when the secret is chosen from the error distribution. We lose a little on parameters in the conversion, specifically the extra block at the beginning of G'. With the parameters n, m', q, χ as above, let $m = n \lceil \log q \rceil = m' - n$, and $G = [I|2I|4I|, \ldots |2^{\ell-1}I] \in \mathbb{Z}^{n \times m}$. The matrix iNTRU distribution with small secret is as follows:

$$\mathsf{MiNTRU}^s = \left\{ \begin{array}{l} \text{draw } S \leftarrow \chi^{n \times n}, \text{ and } E \leftarrow \chi^{n \times m}, \\ \quad \text{output } A := S^{-1} \times (G - E) \bmod q \end{array} \right\}. \tag{4}$$

Lemma 4.1. *For the parameters n, m, m', q, χ as above, if MiNTRU is pseudorandom in $\mathbb{Z}_q^{n \times m'}$, then MiNTRU^s is pseudorandom in $\mathbb{Z}_q^{n \times m}$.*

Proof. (sketch) We show that if we could distinguish MiNTRU^s from random then we could also distinguish MiNTRU from random. Given a MiNTRU instance that we want to distinguish, $A' = [A'_0|A'_1|\ldots|A'_\ell]$ (with $A'_i \in \mathbb{Z}_q^{n \times n}$), we set

$$A_i = {A'_0}^{-1} \times A'_i \bmod q \text{ for } i = 1, \ldots, \ell,$$

(aborting if A'_0 is not invertible), then run the MiNTRU^s distinguisher on $A = [A_1|A_2|\ldots|A_\ell]$. Observe that if A' is uniform then so is A, and if A' is chosen from the MiNTRU distribution then

$$A_i = {A'_0}^{-1} \times A'_i = -{E'_0}^{-1} \times S \times S^{-1} \times (2^{i-1}I - E'_i) = -{E'_0}^{-1} \times (2^{i-1}I - E'_i),$$

for $i = 1, \ldots, \ell$ as needed. □

[8] We use a slightly larger gadget matrix than usual, with an extra first block. The reason will become clear when we prove Lemma 4.1 below.

4.2 Security Reduction

We next show that pseudorandomness of MiNTRU^s (or equivalently MiNTRU) with some error distribution χ, implies the semantic security of our scheme with a related error distribution (but not quite the same). Specifically, let n, m, q, χ be the parameters of the MiNTRU^s distribution above. For a fixed pair of matrices $E, Y \in \mathbb{Z}_q^{n\times m}$, consider the distribution

$$\psi[E, Y] = \{R \leftarrow G^{-1}(Y),\ \text{output } E \times R \bmod q\}.$$

In the provable version of our scheme, the secret key includes the square invertible matrix $S \leftarrow \chi^{n\times n}$, and in addition a fixed error matrix $E \leftarrow \chi^{n\times m}$, and we use the error distribution $\psi[E, M \times G]$ when encrypting a matrix $M \in \mathbb{Z}_q^{n\times n}$. Namely we draw a sample $R \leftarrow G^{-1}(MG) \in \mathbb{Z}_q^{m\times m}$, then output the ciphertext $C := S^{-1} \times (MG - ER) \bmod q$.

Proposition 4.2. *If MiNTRU^s is pseudorandom, then our encryption scheme using the error distribution $\psi[E, M \times G]$ is semantically secure.*

Proof. (sketch) We use the "real-or-random" formulation of semantic security for secret-key encryption [6]. Namely, we have a challenger that chooses a secret key $S \leftarrow \chi^{n\times n}, E \leftarrow \chi^{n\times m}$ and a bit σ, then the adversary repeatedly chooses messages $M_i \in \mathbb{Z}_q^{n\times n}$ and sends to the challenger, who replies either with a uniformly random $C_i \in \mathbb{Z}_q^{n\times m}$ if $\sigma = 0$, or with $C_i := S^{-1} \times (M_i G + E_i)$ if $\sigma = 1$, where $E_i \leftarrow \psi[E, M_i G]$. The adversary eventually outputs a guess σ' for σ, and is considered successful if $\sigma' = \sigma$ with probability significantly larger than $1/2$.

We show that an adversary Adv with a noticeable advantage ϵ can be transformed into a distinguisher between MiNTRU^s and uniform, with advantage close to ϵ. The distinguisher D receives as input an instance of MiNTRU^s, $A \in \mathbb{Z}_q^{n\times m}$, and it interacts with the adversary Adv as follows:

When receiving a matrix M_i from Adv, the distinguisher D draws a sample $R \leftarrow G^{-1}(M_i G)$, and replies with the "ciphertext" $C := AR \bmod q$. When Adv eventually outputs a guess σ', the distinguisher D outputs the same guess. We next show that the distinguishing advantage of D is very close to ϵ.

If A was a uniform matrix in $\mathbb{Z}_q^{n\times m}$ then each $C_i = A \times G^{-1}(\mathsf{something}) \bmod q$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ and independent of A, by the leftover hash lemma. On the other hand, if $A = S^{-1} \times (G - E)$, then we have

$$\begin{aligned} C_i = A \times G^{-1}(M_i G) &= S^{-1} \times \left(G \times G^{-1}(M_i G) - E \times G^{-1}(M_i G)\right) \\ &= S^{-1} \times \left(M_i G - E \times G^{-1}(M_i G)\right), \end{aligned}$$

which is identical to the distribution of the encryption scheme. □

5 Converting Regular Expressions to Automata

In real world applications, regular languages or finite automata are often represented by regular expressions, which have a very compact form and are convenient to store. So it is important for our scheme to be useful when NFAs are specified using regular expressions. In this section we present an efficient method to convert regular expressions to NFAs of relatively small sizes, and we discuss how to find a suitable NFA to bound the noise growth. We assume the reader has some familiarity with regular languages, regular expressions, and finite automata. See Appendix A for basic notation and definitions.

Partial derivatives and NFAs Let Σ be a finite alphabet, and RE be the set of all regular expressions over Σ. We consider the basic operations such as union ("+"), concatenation ("$\cdot$"), and Kleene star ("$*$") on regular expressions. To convert a regular expression to a NFA, we start with Antimirov's partial derivative construction [3], which is an elegant extension of Brzozowski's derivative construction [12] to NFAs. For any symbol $a \in \Sigma$, the *partial derivative* of e w.r.t. a, denoted as $\partial_a(e)$, is a set of regular expressions defined inductively as

$$\partial_a(\epsilon) = \emptyset, \qquad \partial_a(e_0 + e_1) = \partial_a(e_0) \cup \partial_a(e_1), \qquad \partial_a(e^*) = \partial_a(e)e^*$$

$$\partial_a(a_i) = \begin{cases} \{\epsilon\} & \text{if } a_i = a \\ \emptyset & \text{otherwise} \end{cases} \qquad \partial_a(e_0 \cdot e_1) = \begin{cases} \partial_a(e_0)e_1 \cup \partial_a(e_1) & \text{if } \epsilon \in \mathcal{L}(e_0) \\ \partial_a(e_0)e_1 & \text{otherwise} \end{cases}$$

where e, e_0, e_1 range over RE. The partial derivative of e w.r.t. any string is $\partial_\epsilon(e) = \{e\}$ and $\partial_{ua}(e) = \bigcup\{\partial_a(f) \mid f \in \partial_u(e)\}$ where $u \in \Sigma^*$ and $a \in \Sigma$. A regular expression e' is a *partial derivative term* of e if e' is an element of $\partial_w(e)$ for some $w \in \Sigma^*$.

Definition 1 (Partial derivative NFA)**.** For any regular expression e, the *partial derivative NFA* of e is $M_{\mathcal{PD}}(e) = (Q, \Sigma, \delta, Q_I, Q_F)$, where $Q = \partial(e)$, $Q_I = \{e\}$, $Q_F = \{e' \in \partial(e) \mid \epsilon \in \mathcal{L}(e')\}$, and for any $e' \in Q$ and $a \in \Sigma$, $\delta(e', a) = \partial_a(e')$.

Remark. It was shown in [3] that, the set $\partial(e)$ of all partial derivative terms of e is a finite set (with respect to syntactic equality on regular expressions). In fact, $|\partial(e)| \leq n + 1$ where n is the number of occurrences of alphabet symbols in e.

The language of e satisfies $\mathcal{L}(e) = \bigcup_{a\in\Sigma} a \cdot \partial_a(e)$. It follows that the language accepted by $M_{\mathcal{PD}}(e)$ is exactly $\mathcal{L}(e)$.

Ambiguity measure As will be shown later, when evaluating an encrypted NFA on a string, the noise growth is closely related to the amount of nondeterministic choices of the NFA. Here we describe some notions that characterize this quantity. Let $M = (Q, \Sigma, \delta, Q_I, Q_F)$ be a NFA. For any string $w = w_1 \cdots w_k$ where $w_1, \ldots, w_k \in \Sigma$, a *path of w from state s to state t* is a finite sequence of states $s = s_{i_0}, s_{i_1}, \ldots, s_{i_k} = t$ such that $s_{i_j} \in \delta(s_{i_{j-1}}, w_j)$ for all $1 \leq j \leq k$. A path is accepting if $s \in Q_I$ and $t \in Q_F$. The *degree of ambiguity of M*, denoted as $\mathrm{da}(M, k)$, is the maximal number of accepting paths for a string of length k. If $\mathrm{da}(M, k) \leq 1$ for all $k > 0$, then we say M is *unambiguous*.[9] Clearly $\mathrm{da}(M, k) \leq |Q|^{k+1}$. We say that M is *finitely ambiguous* if $\sup\{\mathrm{da}(M, k) \mid k \geq 0\} < \infty$, and M is infinitely ambiguous otherwise. To upper bound the quantity $\mathrm{da}(M, k)$ using a function of k, we can define the *degree of growth of ambiguity of M*, denoted as $\deg(M)$, to be the minimal *degree* of a polynomial $h(\cdot)$ such that $\mathrm{da}(M, k) \leq h(k)$ for all $k \geq 0$. If no such polynomial exists, we simply set $\deg(M) = \infty$. Note that M is finitely ambiguous if and only if $\deg(M) = 0$. It was shown in [44] that $\deg(M)$ can be computed in time $O(n^6|\Sigma|)$ for any NFA M with n states.

On optimizing NFA For our application of evaluating encrypted NFA, an optimal NFA should be such that its encryption can be correctly evaluated on as many strings as possible. Concretely, we want to find a NFA such that the noise term at the end of evaluation is small enough for a

[9] Notice that a DFA M has $\mathrm{da}(M, k) \leq 1$ for all $k \geq 0$, but the converse is not necessarily true. An NFA can have multiple nondeterministic choices at every state but still satisfies $\mathrm{da}(M, k) \leq 1$, in such cases at most one of these choices could lead to a final state.

successful decryption. As we assume the first state will be the only initial state in all our NFAs, we can encrypt the initial state vector with no noise. As a result, we obtain the following bounds on the noise due to homomorphic evaluation of NFAs, which can be bounded using the ambiguity measures of M.

Proposition 5.1. *For any $n \geq 1$, if M is an NFA with n states $s_1, \ldots, s_n$, and w a string of length k, the noise vector $\mathbf{e}$ at the end of homomorphic evaluation of encrypted M on w satisfies the following bounds:*

- *If M is finitely ambiguous, then $\|\mathbf{e}^{(k)}\|_\infty \leq bn^2k\chi \log_b q$.*
- *If M is infinitely ambiguous, then $\|\mathbf{e}^{(k)}\|_\infty \leq bnk^{\deg(M)+1}\chi \log_b q$.*

Notice that both the number of states and the degree of ambiguity contribute to the bound on the noise growth. To find a small noise growth for the general case of processing an arbitrary long input string, we can try to solve the following optimization problem on NFA minimization with bounded ambiguity.

Definition 2 (NFA Minimization with Bounded Ambiguity Problem)**.** For a given NFA of n states and a function $N : \mathbb{N} \to \mathbb{N}$, find an equivalent NFA M with a minimal number of states such that $\mathrm{da}(M, k) \leq N(k)$ for all $k \geq 1$.

A closely related problem is to find a minimal NFA M with a given bound on $\deg(M)$. Conversely, we can consider a similar minimization problem of finding an NFA M with minimal $\deg(M)$ when given a regular expression and a bound on the number of states. These problems seem to be hard in general as evidenced by several exponential separation results in automata theory, and we briefly mention a few. It was shown in [29] that, for each $n > 0$, there exists a NFA of n states such that the minimal equivalent NFA M' of bounded $\deg(M')$[10] have $2^n - 1$ states. With a more strict bound on the ambiguity, it was known [27] that there exist NFAs of n states such that the equivalent finitely ambiguous NFAs have at least $2^{\Omega(n^{1/3})}$ states. A more tractable problem of finding a minimal unambiguous NFA is NP-complete [28, 7].

On the other hand, unambiguous NFAs can have much smaller size than equivalent DFAs. A well-known example is the language $L_n = (a + b)^*a(a + b)^{n-2}$ for any $n \geq 2$: its partial derivative NFA has n states and is unambiguous, but its minimal equivalent DFA requires 2^{n-1} states [33]. The exponential upper bound 2^n can actually be met: it was shown in [30] that there exists a series $\{M_n\}_{n\geq 1}$ of unambiguous NFAs such that M_n has n states but the minimal equivalent DFA of M_n has 2^n states. Notice that, if the size of the given regular expression is small, the bound on the size of the noise is dominated by the degree of ambiguity, which is same for unambiguous NFAs and DFAs. So we can exploit the fact that our scheme supports homomorphic encryption of NFAs and try to find a small unambiguous NFA, which can be more efficient than encrypting DFAs.

A particular useful class of NFA's is the pattern matching languages $L = \Sigma^*K\Sigma^*$ where K is a finite set of strings, called the *pattern*. One can check using the criterion in [44] that the partial derivative NFA for such a language is unambiguous, but its minimal equivalent DFA may have 2^n states and such upper bound is tight, where n is the length of the regular expression for K. Even if K can be specificied using a DFA of m states, the minimal equivalent DFA of L may still have $2^{m-2} + 1$ states. As our scheme supports encryption of NFAs, pattern matching on encrypted patterns can be much more efficient than previous approaches via DFAs.

[10]Note that $\deg(M')$ is bounded if and only if $\mathrm{da}(M', k)$ is at most a polynomial in k for all $k > 0$.

6 Implementation and Performance

Description This section describes a proof of concept implementation of our scheme on short inputs in C++ using the NTL library (version 10.5.0) for a power of two modulus, q, on an Intel i7-2600 3.4 GHz CPU. The implementation is naive in that it only uses NTL's native functionality with no further optimizations. It can be done in a few hundred lines of code and a few days' programming effort. There are many opportunities for optimization since the code was written for simplicity and not efficiency. Despite this, we noticed exceptionally fast evaluation times as listed in Table 1

We conducted tests on N-state NFA's accepting the string-matching languages $(a+b)^*a(a+b)^{N-1}$ with low ambiguity, for some N smaller than the lattice dimention n. Notice that the equivalent minimal DFA's have 2^{N-1} states. We set lattice parameters to $n = 1024$ and $q = 2^{42}$. We kept the modulus both as a power of two and as a power of b in order to take advantage of bit-shifting instead of multiplications and divisions modulo q. In the experiments, we pad the transition matrices to n-dimensional matrices by adding transitions from nonreachable states to final states to increase ambiguity, and hence we effectively obtain an n-state NFA. The strings scanned were randomly generated. At the end of each scan, our code checked for any decryption errors. We observed no decryption errors nor noise overflow. The experiment results for $N = 11$ are listed in Table 1, where time was measured using C++'s "time.h" library.

The noise matrices $E_i \in \mathbb{Z}_q^{n\times n}$ and the secret keys $S \in \mathbb{Z}_q^{n\times n}$ were chosen as uniformly random binary matrices with the latter being invertible modulo q. We used NTL's pseudorandom number generator "Random_ZZ" for all random matrices.

Program Size and the NFA Minimization with Bounded Ambiguity Problem Consider the case where the NFA has infinite ambiguity, but bounded degree of growth of ambiguity. Then, $\|\mathbf{e}^{(m)}\|_\infty \leq bnm^{\deg(M)+1}\chi \log_b q$ as discussed in the previous section. By setting the modulus just above the error growth, we see that the bit length of the modulus is linear in $\deg(M)+1$. Now as we view total memory for the encrypted NFA, $n^2|\Sigma|\log_2(q)\log_b(q)$ bits, we see that efficiency is quadratic in NFA's number of states and quadratic in the degree of growth of ambiguity (though we have some control over $\log_b(q)$ by choosing a large base b). This gives us an exact relation between the number of states, the NFA's ambiguity, and performance.

MiNTRU^s can be cryptanalyzed by NTRU attacks like dimension reduction [32] and the hybrid attack [26] for key recovery. Therefore, we use the uSVP attack to estimate the time for a key recovery attack as in [1] and set the LWE noise parameter as $\alpha = \sqrt{2n}/q$ in the on-line LWE bit security estimator[11]. Rough estimates for the bit-securities in the 750, and 1024-state settings are 80 bits of security, and 100 bits of security respectively.

Potential Optimizations One potential optimization is parallelization through the unused states. Say we must evaluate a long string (10000 bits) but only use a 100 state NFA. Then, we can evaluate ten such NFAs in parallel by setting the transition matrix for character $a \in \Sigma$ as the block diagonal matrix with the blocks as the smaller transition matrices in the small parameter setting. The total number of states must stay above a few hundred for this corresponds to the lattice dimension of the underlying lattice problem.

[11] https://bitbucket.org/malb/lwe-estimator

States (n) and Input Length ($4N$)	NFA Enc. Time	Matching	Enc. NFA	RAM used
1024 states and 256 bit S.L.	16.35 sec	1.53 sec	66Mb	172Mb
1024 states and 512 bit S.L.	16.66 sec	3.34 sec	66Mb	172Mb
1024 states and 1024 bit S.L.	16.53 sec	6.63 sec	66Mb	172Mb
1024 states and 16384 bit S.L.	16.76 sec	98.97 sec	66Mb	172Mb
1024 states and 65536 bit S.L.	16.42 sec	394.47 sec	66Mb	172Mb

Table 1: Running times for each function along with memory. "NFA Enc. Time" is the time to encrypt an NFA with n states, "Matching" is the time to evaluate an encrypted NFA on an input of N characters, "Enc. NFA" is the memory storage for the encrypted NFA, and the last column measures the total RAM used during encryption, evaluation, and decryption. Total RAM usage was measured with the "sys/resource.h" library in unix.

Let $G = I_n \otimes \mathbf{g}^t$ for $\mathbf{g}^t = (1, b, \cdots, b^{\log_b(q)-1})$ as in [35]. We expect to see smaller noise growth via a randomized bit decomposition for the decomposition of the encrypted state vector, as used in [2]. This can be done with a simple tweak to Babai's nearest plane algorithm [4] on the G-matrix's null lattice $\Lambda_q^{\perp}(G) = \{x \in \mathbb{Z}^m : Gx = 0 \mod q\}$ and its cosets.

References

[1] M. R. Albrecht, B. R. Curtis, A. Deo, A. Davidson, R. Player, E. W. Postlethwaite, F. Virdia, and T. Wunderer. Estimate all the {LWE, NTRU} schemes! *IACR Cryptology ePrint Archive*, 2018:331, 2018.

[2] J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014, Part I*, pages 297–314. Springer, 2014.

[3] V. M. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theor. Comput. Sci.*, 155(2):291–319, 1996.

[4] L. Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[5] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

[6] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 394–403. IEEE Press, 1997.

[7] H. Björklund and W. Martens. The tractability frontier for NFA minimization. *J. Comput. Syst. Sci.*, 78(1):198–210, 2012.

[8] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3):13, 2014.

[10] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584. ACM, 2013.

[11] Z. Brakerski and V. Vaikuntanathan. Lattice-based FHE as secure as PKE. In M. Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14*, pages 1–12. ACM, 2014.

[12] J. A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964.

[13] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.

[14] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 3–33. Springer, 2016.

[15] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pages 377–408. Springer, 2017.

[16] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[17] N. Gama, M. Izabachène, P. Q. Nguyen, and X. Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 528–558. Springer, 2016.

[18] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st ACM Symposium on Theory of Computing – STOC 2009*, pages 169–178. ACM, 2009.

[19] C. Gentry, S. Gorbunov, and S. Halevi. Graph-induced multilinear maps from lattices. In Y. Dodis and J. B. Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527. Springer, 2015.

[20] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In C. Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008.

[21] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology - CRYPTO 2013, Part I*, pages 75–92. Springer, 2013.

[22] S. Halevi and V. Shoup. Faster homomorphic linear transformations in helib. *IACR Cryptology ePrint Archive*, 2018:244, 2018.

[23] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[24] R. Hiromasa, M. Abe, and T. Okamoto. Packing messages and optimizing bootstrapping in GSW-FHE. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 699–715, 2015.

[25] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In J. Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[26] N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007.

[27] J. Hromkovic and G. Schnitger. Ambiguity and communication. *Theory Comput. Syst.*, 48(3):517–534, 2011.

[28] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993.

[29] H. Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM J. Comput.*, 27(4):1073–1082, 1998.

[30] H. Leung. Descriptional complexity of nfa of different ambiguity. *Int. J. Found. Comput. Sci.*, 16(5):975–984, 2005.

[31] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

[32] A. May and J. H. Silverman. Dimension reduction methods for convolution modular lattices. In Silverman [40], pages 110–125.

[33] A. R. Meyer and M. J. Fischer. Economy of description by automata, grammars, and formal systems. In *12th Annual Symposium on Switching and Automata Theory, East Lansing, Michigan, USA, October 13-15, 1971*, pages 188–191, 1971.

[34] D. Micciancio. Improving lattice based cryptosystems using the hermite normal form. In Silverman [40], pages 126–145.

[35] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[36] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 372–381. IEEE Computer Society, 2004.

[37] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166. Springer, 2006.

[38] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[39] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.

[40] J. H. Silverman, editor. *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[41] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014. Early verion at `http://eprint.iacr.org/2011/133`.

[42] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[43] B. Wang, X. Wang, R. Xue, and X. Huang. Matrix fhe and its application in optimizing bootstrapping. *The Computer Journal*, page bxy088, 2018.

[44] A. Weber and H. Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991.

[45] S. Yu. Handbook of formal languages, vol. 1. chapter Regular Languages, pages 41–110. Springer-Verlag, Berlin, Heidelberg, 1997.

A Definitions on Regular Expressions and NFA

We recall some standard definitions about regular languages and finite automata [45]. Let Σ be a finite alphabet, and Σ^* the free monoid generated by Σ. A *string* w is an element of Σ^*, which can be written as a finite sequence of symbols $w = w_1 w_2 \cdots w_k$ where $w_1, \ldots, w_k \in \Sigma$, and its *length* is

$|w| = k$. The *empty string* is denoted by ϵ, which is the neutral element of Σ^*. The *concatenation* of two strings $u = u_1 \cdots u_m$ and $v = v_1 \cdots v_n$ is a string $uv = u_1 \cdots u_m v_1 \cdots v_n$. A *language* over Σ is a subset of Σ^*. For any languages L and K, we consider the following regular operations: (union) $L \cup K$, (product) $LK = \{uv \mid u \in L, v \in K\}$, and (Kleene star) $L^* = \cup_{i \geq 0} L^i$, where $L^0 = \{\epsilon\}$, and $L^i = LL^{i-1}$ for $i > 0$. *Regular languages* are the smallest class of languages containing the basic languages $\emptyset$, $\{\epsilon\}$, and $\{a_i\}$ for all $a_i \in \Sigma$ that are closed under regular operations.

A *nondeterministic finite automaton (NFA)* over Σ is a quintuple $M = (Q, \Sigma, \delta, Q_I, Q_F)$, where $Q = \{s_1, \ldots, s_n\}$ is a finite set of states, $\delta : Q \times \Sigma \to \wp(Q)$ is a transition function, $Q_I \subseteq Q$ is the set of initial states, and $Q_F \subseteq Q$ is the set of final states. We can extend δ to a function $\delta : Q \times \Sigma^* \to \wp(Q)$ over strings in the natural way. Without loss of generality, we assume that all our NFAs have a single initial state s_1. A string $w \in \Sigma^*$ is *accepted* by a NFA M if $\delta(s_1, w) \cap Q_F \neq \emptyset$. The set of all the strings accepted by a NFA M is called the *language of* M, and it is denoted by $\mathcal{L}(M)$. A *deterministic finite automaton (DFA)* is a NFA such that $\delta(s, a_i)$ is a singleton set for all $s \in Q$ and $a_i \in \Sigma$, and $|Q_I| = 1$.

A *regular expression* over Σ is a formal expression generated by the following grammar rules:

$$\mathsf{RE} \to \epsilon \mid a_i \mid (\mathsf{RE} + \mathsf{RE}) \mid (\mathsf{RE} \cdot \mathsf{RE}) \mid (\mathsf{RE})^*,$$

where a_i ranges over Σ. The operator $*$ takes the highest precedence, followed by $\cdot$, and then by $+$. The parentheses can be omitted when there is no ambiguity. The operator $\cdot$ is usually omitted as well, and concatenations can be written as juxtapositions of regular expressions. For a regular expression e, its *language* $\mathcal{L}(e)$ can be defined inductively as follows:

$$\begin{aligned} &\mathcal{L}(\epsilon) = \{\epsilon\}, && \mathcal{L}(a_i) = \{a_i\}, \\ &\mathcal{L}(e_0 + e_1) = \mathcal{L}(e_0) \cup \mathcal{L}(e_1), && \mathcal{L}(e_0 \cdot e_1) = \{uv \mid u \in \mathcal{L}(e_0), v \in \mathcal{L}(e_1)\}, \\ &\mathcal{L}(e^*) = \cup_{i \geq 0} \mathcal{L}(e)^i, \end{aligned}$$

where a_i ranges over Σ, and e_0, e_1 are regular expressions. For any set R of regular expressions, let $\mathcal{L}(R) = \cup_{e \in R} \mathcal{L}(e)$. It is well known that the languages defined by regular expressions are exactly the regular languages, which are exactly the languages accepted by finite automata.

For any sets R, T of regular expressions, we write RT for the set of regular expressions

$$RT = \{e \cdot f \mid e \in R, f \in T\},$$

and we write $Re = \{f \cdot e \mid f \in R\}$ and $eR = \{e \cdot f \mid f \in R\}$; in particular, $\emptyset T = R\emptyset = \emptyset e = e\emptyset = \emptyset$.

B Proofs

In this section we present proofs that are omitted in the main paper.

Proof of Proposition 5.1. Let $M = (Q, \Sigma, \delta, \{s_1\}, Q_F)$ be an NFA with n states $s_1, \ldots, s_n$. For any $t \in Q$ let $M_t = (Q, \Sigma, \delta, Q, \{t\})$ be the NFA obtained from M by setting all states to be initial and t the only final state. Notice that $\mathrm{da}(M_t, \ell)$ is an upper bound on the total number of paths in M on a string of length ℓ from any state to t.

Let $Q = \{s_1, \ldots, s_n\}$ be the set of states of M, and let $w = w_1 \cdots w_k$. For all $1 \leq i \leq k$, the encrypted state vector $\mathbf{q}^{(i)}$ after reading w_i is:

$$\mathbf{q}^{(i)} = \sum_{j=0}^{\log_b q} C_{w_i,j}\mathbf{q}_j^{(i-1)} = \beta S^{-1} V_{w_i} \cdots V_{w_1}\mathbf{v} + S^{-1}(V_{w_i}\mathbf{e}^{(i-1)} + \sum_{j=0}^{\log_b q} E_{w_i,j}\mathbf{q}_j^{(i-1)}),$$

where $\mathbf{e}^{(i-1)}$ is the noise term after reading the previous symbol w_{i-1}. As in our assumption, s_1 is always the sole initial state in M, we can set the initial noise $\mathbf{e}^{(0)} = \mathbf{0}$ without leaking any additional information about the NFA M. By expanding all the noise terms, we get

$$\mathbf{e}^{(k)} = \sum_{\ell=2}^{k} V_{w_k} \cdots V_{w_\ell} \sum_{j=0}^{\log_b q} E_{w_{\ell-1},j}\mathbf{q}_j^{(\ell-2)} + \sum_{j=0}^{\log_b q} E_{w_k,j}\mathbf{q}_j^{(k-1)}. \tag{5}$$

Notice that, for any symbol $a \in \Sigma$, the (t,s)-th entry of V_a is 1 if $t \in \delta(s,a)$ and it is 0 otherwise. So the (t,s)-th entry of the product $V_{w_i} \cdots V_{w_\ell}$ counts the number of paths from s to t on the string $w_\ell \cdots w_i$, where $1 \leq \ell \leq i$. Let $\mathbf{1}$ be the vector whose entries are all 1. Then the t-th entry of the vector $V_{w_i} \cdots V_{w_\ell}\mathbf{1}$ counts the total number of paths from an arbitrary state to t on this string, which is at most $\mathrm{da}(M_t, i-\ell+1)$. Thus we have

$$\|V_{w_k} \cdots V_{w_\ell} \sum_{j=0}^{\log_b q} E_{w_{\ell-1},j}\mathbf{q}_j^{(\ell-2)}\|_\infty \leq bn\chi \log_b q \cdot \max_{t\in Q}\{\mathrm{da}(M_t, k-\ell+1)\}.$$

It follows that the final noise vector $\mathbf{e}^{(k)}$ can be bounded by

$$\|\mathbf{e}^{(k)}\|_\infty \leq bn\chi \log_b q \cdot \sum_{\ell=1}^{k-1} \max_{t\in Q}\{\mathrm{da}(M_t, \ell)\} + bn\chi \log_b q \tag{6}$$

If M is finitely ambiguous, then for all $s, t \in Q$, the number of paths of w from s to t is at most 1 [44]. So $\mathrm{da}(M_t, \ell) \leq n$ for all $t \in Q$ and $\ell \geq 0$, and $\mathbf{e}^{(k)}$ can be bounded by

$$\|\mathbf{e}^{(k)}\|_\infty \leq bkn^2\chi \log_b q.$$

For the case where M is infinitely ambiguous, notice that $\mathrm{da}(M_t, \ell) \leq \ell^{\deg(M)}$ for all $\ell \geq 1$, and we have

$$\begin{aligned}\|\mathbf{e}^{(k)}\|_\infty &\leq b\chi \log_b q \sum_{\ell=1}^{k-1} \ell^{\deg(M)} + b\chi \log_b q \\ &\leq bnk^{\deg(M)+1}\chi \log_b q\end{aligned}$$

□

C Performance comparisons with HAO15

In this section we present a brief analysis of applying the matrix-FHE scheme of HAO15 [24] to the case of homomorphic evaluation of NFA.

Fix a NFA $\mathcal{M}$ of r states and with an alphabet Σ, and let $M_\sigma \in \{0,1\}^{r\times r}$ for $\sigma \in \Sigma$ be its transition matrices on symbol σ. Recall the "leveled version" of the HAO15 scheme as described in Section 3.1. To encrypt $\mathcal{M}$ for homomorphic evaluation on any string of length at most k, we let the key generation procedure sample $k+1$ random matrices $S'_i \leftarrow \chi^{r\times n}$ and sets the secret key for level i to be $S_i = [I_r | -S'_i]$ for $0 \le i \le k$. For each $\sigma \in \Sigma$, encrypt M_σ with all keys S_i for $i \in [k]$:

$$C_{\sigma,i} = \begin{pmatrix} S'_i A_{\sigma,i} + E_{\sigma,i} \\ \hline A_{\sigma,i} \end{pmatrix} + \begin{pmatrix} M S_{i-1} \\ \hline 0 \end{pmatrix} \times G \bmod q,$$

where $N = (n+r)\lceil \log q \rceil$, $A_{\sigma,i} \leftarrow \mathbb{Z}_q^{n\times N}$, $E_{\sigma,i} \leftarrow \chi^{r\times N}$. We also encrypt the initial state vector $\mathbf{v} = (1,0,\ldots,0)^t$ in a ciphertext $\mathbf{c} \in \mathbb{Z}_q^{n+r}$:

$$\mathbf{c} = \begin{pmatrix} S'_0 \mathbf{a} + \mathbf{e} \\ \hline \mathbf{a} \end{pmatrix} + \begin{pmatrix} \mathbf{v} \\ \hline 0 \end{pmatrix} \bmod q,$$

for a uniformly random vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and a noise vector $\mathbf{e} \leftarrow \chi^r$.

To evaluate the NFA, let $\mathbf{c}_0 = \mathbf{c}$, and let $\mathbf{c}_i = C_{w_i,i} \odot \mathbf{c}_{i-1} = C_{w_i,i} \times G^{-1}(\mathbf{c}_{i-1})$. Then each ciphertext $\mathbf{c}_i$ satisfies $S_i \mathbf{c}_i = (\prod_{j=i}^{1} M_{w_j}) \times \mathbf{v} + \mathbf{e}_i$ for some noise vector $\mathbf{e}_i$. The final ciphertext is $\mathbf{c}_k$ with a noise vector $\mathbf{e}_k$ as follows:

$$\mathbf{e}_k = E_{w_k,k} G^{-1}(\mathbf{c}_{k-1}) + \sum_{i=2}^{k} (\prod_{j=k}^{i} M_{w_j}) E_{w_{i-1},i-1} G^{-1}(\mathbf{c}_{i-2}) + (\prod_{j=k}^{1} M_{w_j})\mathbf{e}.$$

The l_∞ norm of $\mathbf{e}_k$ can be bounded by

$$\|\mathbf{e}_k\|_\infty \le \chi N + \chi N \sum_{\ell=2}^{k} \mathrm{da}(M,\ell) + \chi \mathrm{da}(M,k),$$

which must be bounded away from $q/4$.

For performance comparison, consider two cases of the ambiguity measures of M:

- M is finitely ambiguous: We have $\mathrm{da}(M,\ell) \le r$ for all $1 \le \ell \le k$, so w.h.p.

  $$\|E\|_\infty \le \alpha q (n+r)(kr+1)\log q,$$

  where $\alpha = \sqrt{2n}/q$ is the LWE noise parameter. Thus, in the HAO15 scheme we can homomorphically evaluate M on strings of length $k \le \frac{1}{\alpha(n+r)r\log q}$. For example, assuming at least 100 bit of security is needed, for a NFA of 10 states on strings of length up to 1000, we must choose $n = 750$ and $q = 2^{34}$, and for a NFA of 1000 states on strings of length up to 1000, we need $n = 1024$ and $q = 2^{42}$. On the other hand, using our scheme we can evaluate M on strings of length $k \le \frac{q}{b^2 n\chi r \log_b q}$. So, using our scheme with the above sets of parameters, we can homomorphically evaluate a NFA of 10 states on strings of length up to 8000 in the former case, and we can evaluate a NFA of 1000 states on strings of length up to 12000 in the latter case.

- M is infinitely ambiguous: We have $\mathrm{da}(M,\ell) \le \ell^{\deg(M)}$, so w.h.p.

  $$\|E\|_\infty \le \alpha q (n+r)\log q \cdot (\sum_{\ell=1}^{k} \ell^{\deg(M)} + 1) \le \alpha q (n+r)\log q k^{\deg(M)+1}$$

Using the same parameters as the above to achieve at least 100 bit of security, and assuming that $\deg(M) = 2$ for the NFA M, we can homomorphically evaluate strings of length up to 25 in the HAO15 scheme, whereas we can homomorphically evaluate strings of length up to 55 in our scheme.

Moreover, the computational complexity of k homomorphic matrix multiplications, assuming naive matrix-vector multiplication of complexity $O(n^2)$, is $O(k(r+n)^2 \log q)$. On the other hand, the complexity of our homomorphic evaluation procedure is $O(kn^2 \log q)$.

D Hardness of MiNTRU from LWE with a Trapdoor

Here we prove the reduction alluded to in Section 1.2. We define a trapdoor oracle for an arbitrary matrix $B \in \mathbb{Z}_q^{n \times m}$ as an oracle which takes as input B, a vector $\mathbf{v} \in \mathbb{Z}_q^n$, and outputs a discrete Gaussian integer vector $\mathbf{x} \in \mathbb{Z}^m$ conditioned on $B\mathbf{x} \mod q = \mathbf{v}$. Repeated calls to the oracle are assumed to use independent random coins. Further, we assume the oracle's distribution samples above the smoothing parameter of

$$\Lambda_q^{\perp}(B) = \{\mathbf{x} \in \mathbb{Z}^m : B\mathbf{x} = \mathbf{0} \mod q\}$$

for a uniformly random B, for some negligible functon $\epsilon(n)$. In general, the smoothing parameter of $\Lambda_q^{\perp}(B)$ is just above the smoothing parameter of $\mathbb{Z}^m$, for some negiligible $\epsilon(n)$, when $m > n \log q$, [35, Lemma 2.4].

Let n-secret LWE define the distribution

$$\{(A, B = SA + E) : A \leftarrow \mathbb{Z}_q^{n \times m}, S \leftarrow \mathbb{Z}_q^{n \times n}, E \leftarrow \chi^{n \times m}\}$$

for some distribution χ. Next, we show the pseudorandomness of MiNTRU follows from the n-secret LWE distribution with a trapdoor oracle for B. Let $G \in \mathbb{Z}_q^{n \times m'}$ be any formulation of the gadget matrix. ($G = [0|2I| \cdots |2^{\log q - 1} I] \in \mathbb{Z}_q^{n \times n(\log q + 1)}$ in the MiNTRU definition.)

Proposition D.1. *Let $n \in \mathbb{N}$, $q < 2^{poly(n)}$, χ be a distribution over $\mathbb{Z}_q$, $m \geq n \log q$, and m' be the number of columns in the G-matrix. Further, let $q = \omega(\sqrt{m})$. Then, the pseudorandomness* MiNTRU *with error distribution $\chi^{n \times m} \cdot B^{-1}(G)$ follows from the pseudorandomness n-secret LWE with a oracle for B.*

Proof. Given (A, B), we call the orcale m' times to get $X \leftarrow B^{-1}(G)$. Then we return $AX \mod q$. Notice when (A, B) are generated uniformly and independently, then $AX \mod q$ is negligibly close to uniformly random by the LHL, along with Lemmas 2.3 and 2.4. Conversely, we have $S^{-1} \in \mathbb{Z}_q^{n \times n}$ exists with high probability and $A = S^{-1}(B - E) \mod q$ when (A, B) are sampled as the n-secret LWE distribution. Therefore,

$$AB^{-1}(G) = S^{-1}(G - EB^{-1}(G)) = S^{-1}(G - E') \mod q.$$

□

Remark. There is an identical reduction reducing n-secret LWE with a trapdoor with small secrets to MiNTRU^s.