ChatGPT

VEINN: Vector Encrypted Invertible Neural Networks

Abstract

VEINN is a novel encryption framework that combines *invertible neural network* architectures with lattice-based cryptography to achieve a potential post-quantum encryption scheme. In VEINN, plaintext messages are first encoded into high-dimensional vectors, which are then transformed through a sequence of key-dependent *affine coupling* layers (a type of invertible neural network or "normalizing flow") [1] [2]. A small, key-derived noise vector is added to the transformed vector to produce the ciphertext, ensuring that without the secret key the inversion is infeasible. The secret key consists of both a lattice-based component (e.g. a Ring-LWE key pair) and the parameters of the invertible neural network. We show how VEINN can be instantiated using standard *ring-LWE* or *NTRU*-style hardness assumptions, yielding a public-key encryption scheme whose security reduces to the assumed hardness of these lattice problems. We formalize VEINN's encryption and decryption algorithms, define the underlying hard problems, and prove that VEINN is IND-CPA secure assuming the hardness of Ring-LWE. We compare VEINN to existing NIST post-quantum cryptography candidates (e.g. CRYSTALS-Kyber, NTRUPrime, Classic McEliece) in terms of design principles, key and ciphertext sizes, and assumed security. VEINN's novelty lies in its use of nonlinear, high-dimensional vector transformations – a nontrivial "hardened" layer – to diffuse plaintext bits, which we argue could provide additional resistance to cryptanalysis beyond that of its lattice base.

1. Introduction

The advent of large quantum computers would render widely used public-key schemes (RSA, ECC) insecure, due to Shor's algorithm. In response, the cryptographic community is developing *post-quantum* public-key schemes whose security relies on problems believed hard for quantum computers [3] [4]. Lattice-based schemes (e.g. Learning With Errors (LWE) and its ring version RLWE) and code-based schemes (e.g. Classic McEliece) are among the leading candidates. In this work we propose an experimental approach that augments a lattice-based encryption scheme with *invertible neural networks* (INNs), yielding what we call **VEINN (Vector Encrypted Invertible Neural Network)**. The key idea is to first embed the plaintext into a continuous high-dimensional vector and then apply multiple *affine coupling layers* (from the normalizing-flow literature) parameterized by secret keys. These layers perform nonlinear, reversible transformations of the vector [1] [2]. A small key-dependent noise vector is then added to the transformed vector to produce the ciphertext. Legitimate decryption subtracts the same noise and inverts the neural transformation to recover the plaintext, whereas without the key the output appears as a complex mixture of the plaintext components and noise.

Contributions: In this paper we **formalize** the VEINN scheme and analyze its security. Our main contributions are:

- We **design** the VEINN encryption scheme as a (public-key) ciphertext construction, combining a lattice-based cryptosystem (such as Ring-LWE/NTRU) with invertible-neural-network layers. We give explicit mathematical definitions of the encryption/decryption algorithms, including encoding of messages into vectors and the exact affine coupling transformations used.

- We **define the underlying hard problems**, including the standard *RLWE (Ring Learning With Errors)* problem and a new *invertible neural-network inversion* problem. We state formal security claims (e.g. IND-CPA under RLWE) and provide reductions/proofs.

- We **compare** VEINN to established post-quantum schemes (CRYSTALS-Kyber, NTRUPrime, Classic McEliece, etc.), discussing key/ciphertext sizes, performance, and assumed security. We highlight VEINN's novelty in leveraging neural-network-based diffusion in a cryptographic setting, and outline potential advantages and pitfalls.

- We provide an in-depth **security analysis**, showing that VEINN is at least as secure as its lattice base (under standard assumptions), and discussing any additional assumptions (e.g. that inverting an unknown neural permutation is hard).

This document is structured as follows. Section 2 reviews background on RLWE, NTRU, and invertible neural networks. Section 3 describes the VEINN scheme in detail. Section 4 presents security definitions and proofs. Section 5 compares VEINN to other PQC candidates. Section 6 concludes.

2. Preliminaries and Related Work

2.1 Lattice-Based Cryptography (LWE, RLWE, NTRU)

Learning With Errors (LWE) and its ring variant *RLWE* are canonical hard problems for post-quantum cryptography [4]. Informally, the RLWE problem is: given a polynomial ring $R = \mathbb{Z}[X]/(f(X))$ (e.g. $f(X)=X^n+1$) and a modulus q, there is a secret polynomial $s\in R_q$; samples are given of the form $(a, b=a\cdot s + e \bmod q)$ where $a\in R_q$ is uniform and e is "small" noise. The decision version asks to distinguish such samples from uniform. The worst-case hardness of (Ring-)LWE is related to lattice problems such as finding short vectors on ideal lattices [3] [4]. No polynomial-time algorithms are known for (R)LWE, even on quantum computers [4]. The RLWE assumption underlies many NIST PQC candidates (e.g. CRYSTALS-Kyber, FrodoKEM, Saber) because it allows efficient key sizes and fast ring arithmetic. For example, Kyber's public key is essentially $(\mathbf{A},\mathbf{A}\cdot \mathbf{s} + \mathbf{e})$ in a polynomial ring [5], relying on the hardness of RLWE.

NTRU is an older lattice-based scheme with a similar ring structure. In NTRU one works in $R = (\mathbb{Z}/q)[X]/(f(X))$ with two small polynomial keys $(f,g)\in R$, and encryption is of the form $c = r\cdot h + m \bmod q$, where $h = g * f^{-1}$ is public, r is random noise, and m is encoded plaintext [6]. Decryption uses f to recover m. NTRU was one of the first proposed lattice PKCs (circa 1998) [7]. It enjoys very fast arithmetic (additions and shifts) and relatively small memory, but public keys still grow linearly in a parameter N (public key size $\approx N\log q$ bits [6]). Modern variants (NTRUPrime, Saber) remain in NIST competition.

Code-based schemes (e.g. Classic McEliece) assume the hardness of syndrome decoding in random error-correcting codes [8]. Classic McEliece is an IND-CCA KEM finalist known for very large keys (e.g. >2–4 Mbit for high security) but very fast operations [9]. We will use Classic McEliece as a baseline for extreme key-size comparison.

2.2 Invertible Neural Networks (Normalizing Flows)

Invertible neural networks, or *normalizing flows*, are deep models that define a bijection $F: \mathbb{R}^n \to \mathbb{R}^n$ via a sequence of simple invertible layers [1] [2]. A common building block is the *affine coupling layer*, introduced in *Real NVP* [1]. One splits an input $\mathbf{u}=(\mathbf{u}_1,\mathbf{u}_2)$ into two parts, then transforms:

$$\begin{aligned} \mathbf{v}_1 &= \mathbf{u}_1 \odot \exp\big(s_2(\mathbf{u}_2)\big) + t_2(\mathbf{u}_2), \\ \mathbf{v}_2 &= \mathbf{u}_2 \odot \exp\big(s_1(\mathbf{v}_1)\big) + t_1(\mathbf{v}_1), \end{aligned}$$

where s_i,t_i are arbitrary (potentially deep) neural networks, and $\odot$ is element-wise product [2] [1]. The coupling layer is invertible: given $(\mathbf{v}_1,\mathbf{v}_2)$ one computes

$$\begin{aligned} \mathbf{u}_2 &= (\mathbf{v}_2 - t_1(\mathbf{v}_1)) \odot \exp\big(-s_1(\mathbf{v}_1)\big), \\ \mathbf{u}_1 &= (\mathbf{v}_1 - t_2(\mathbf{u}_2)) \odot \exp\big(-s_2(\mathbf{u}_2)\big). \end{aligned}$$

Crucially, s_i and t_i themselves need not be invertible [10], but their outputs serve as scale/translate factors. By composing multiple such layers (with alternating splits), one obtains a very flexible invertible transformation of $\mathbb{R}^n$ [10]. These flows have been used for density modeling, generative models, and solving inverse problems due to their tractable Jacobians and exact inverses [10].

In VEINN we adopt a simplified coupling architecture (potentially over a finite field) to serve as a *key-dependent, invertible permutation* of message vectors. The secrecy of the network parameters (s_i,t_i) (derived from the secret key) makes inverting the transform without the key equivalent to solving a complex system of equations. We leverage the high-dimensional nonlinearity of INNs as an added “hardening” layer on top of conventional encryption.

2.3 Other Post-Quantum Schemes

For comparison, we note that **CRYSTALS-Kyber** (now NIST-standard KEM) is an RLWE-based KEM whose public key is on the order of a few kilobytes, and ciphertexts a few hundred bytes [5]. **NTRUPrime** is similar in flavor to Kyber but uses a prime modulus and different algebraic structure. **Classic McEliece** (code-based) has extremely large public keys (hundreds of KB to MB) [9] but small ciphertexts. Table

$$keycompare$$

(informal) summarizes typical sizes:

Scheme	Hardness Assumption	PK size (bits)	SK size (bits)	CT size (bits)
Kyber-768 (Lvl 1)	RLWE (module-LWE)	~ 3kB (≈24k bits)	~ 2kB	≈ 1kB

Scheme	Hardness Assumption	PK size (bits)	SK size (bits)	CT size (bits)
NTRU-Prime-460	NTRU ring	~ 3kB	~ 1.5kB	~ 3kB
Classic McEliece	Binary Goppa code	261120 bytes [9] (≈2.1M bits)	~ 6492 B (≈52k bits)	96 bytes (768 bits)
VEINN (example)	RLWE + INN	Depends on params; see §5	size of (RLWE SK + INN weights)	similar to PK + vector

Table 1: *Comparison of key/ciphertext sizes for representative PQC schemes. VEINN's sizes depend on chosen vector dimension and network complexity (see §5).*

In the sequel, we will refer to the lattice-based primitives (RLWE, NTRU) for hardness, but emphasize VEINN's extra INN mixing step as novel.

3. The VEINN Encryption Scheme

In this section we define the VEINN scheme. We will describe it as a *public-key encryption* (or KEM-like) scheme. The secret key will include both a lattice secret and INN parameters. For concreteness, we present VEINN using a Ring-LWE base, but analogous NTRU variants can be constructed.

3.1 Scheme Overview

Let $R = \mathbb{Z}[X]/(f(X))$ be a cyclotomic ring (e.g. $f=X^n+1$) and q a large modulus. We assume the usual hardness of (Ring)-LWE in $R_q = R/(q)$. VEINN works as follows:

1. **Key Generation**: Generate a ring-LWE keypair (pk, sk) with secret $s\in R_q$. Also generate a secret *invertible neural network key* K, which consists of parameters for ℓ affine coupling layers (scale/translate functions s_i,t_i). The public key is $(pk, \text{description of K if we allow public INN structure})$, and secret key is (sk, K). (*We may choose to keep the INN parameters private to enhance security; see discussion in §5.*)

2. **Encryption**: To encrypt a message m, proceed as follows:

3. **Message Vectorization**: Encode the message m into a vector $\mathbf{m}\in \mathbb{Z}_q^n$ (for example, by splitting bits into chunks and embedding as elements of $\mathbb{Z}_q$). Then apply a linear *encoder* (e.g. map to $[-1,1]^n$ floats or to ${0,\dots,q-1}^n$) to get an n-dimensional plaintext vector $\mathbf{x}$.
4. **Invertible Mapping**: Compute $\mathbf{y} = F_K(\mathbf{x})$, where $F_K: \mathbb{Z}_q^n \to \mathbb{Z}_q^n$ is the invertible neural-network mapping defined by ℓ coupling layers parameterized by K. Concretely, we split $\mathbf{x} = (\mathbf{u}_1, \mathbf{u}_2)$ and apply $ \mathbf{y} = (\mathbf{v}_1,\mathbf{v}_2)$ as:

$$\mathbf{v}_1 = \mathbf{u}_1 \odot \exp\big(s_2(\mathbf{u}_2)\big) + t_2(\mathbf{u}_2), \quad \mathbf{v}_2 = \mathbf{u}_2 \odot \exp\big(s_1(\mathbf{v}_1)\big) + t_1(\mathbf{v}_1),$$

then alternate splits for each layer [2] [1]. The final output $\mathbf{y}$ is a diffusion of $ \mathbf{x}$.

5. **Noise Masking (Lattice Encryption)**: Sample a small "noise" vector $\mathbf{v}\in \mathbb{Z}_q^n$ (with entries from a distribution, e.g. uniform or Gaussian over a small range). Encrypt this vector using Ring-LWE: compute RLWE ciphertext $c = (c_0,c_1) = (a\cdot \mathbf{r} + \mathbf{e}_0,\; pk \cdot \mathbf{r} + \mathbf{e}_1 + \mathbf{v})$, where $\mathbf{r},\mathbf{e}_0,\mathbf{e}_1$ are small random errors and $pk = a\cdot s + e$ is the public key polynomial. This yields c that hides $ \mathbf{v}$ under the hardness of Ring-LWE.
6. **Ciphertext Formation**: Output the ciphertext $(c,\, \mathbf{c}_2)$ where we set $\mathbf{c}_2 = \mathbf{y} + \mathbf{v}\bmod q$. That is, we add the same noise $\mathbf{v}$ to the INN output.

In summary, the ciphertext consists of the RLWE encryption of $\mathbf{v}$ together with $\mathbf{y} + \mathbf{v}$. To a full specification one may append any necessary random seeds or padding.

1. **Decryption**: Given ciphertext $(c=(c_0,c_1),\, \mathbf{c}_2)$ and secret key (s,K), do:
2. **Recover $\mathbf{v}$**: Use the RLWE secret key s to compute $c_1 - s\cdot c_0 \bmod q$, which (if errors are small) yields $\mathbf{v}$ exactly. This is the standard RLWE decryption step.
3. **Subtract Noise**: Compute $\mathbf{y} = \mathbf{c}_2 - \mathbf{v} \bmod q$. Since $\mathbf{c}_2 = F_K(\mathbf{x}) + \mathbf{v}$, this gives $\mathbf{y} = F_K(\mathbf{x})$.
4. **Invert INN**: Apply the inverse invertible network F_K^{-1} to recover $\mathbf{x}$ (and thus the message m). Each coupling layer is exactly inverted as in [§2.2], e.g.

$$\mathbf{u}_2 = (\mathbf{v}_2 - t_1(\mathbf{v}_1)) \odot \exp\big(-s_1(\mathbf{v}_1)\big), \quad \mathbf{u}_1 = (\mathbf{v}_1 - t_2(\mathbf{u}_2)) \odot \exp\big(-s_2(\mathbf{u}_2)\big),$$

   and similarly for each layer in reverse. This recovers the original plaintext vector $\mathbf{x}$ exactly (up to any acceptable drift). From $\mathbf{x}$ we decode the message m.

Because the decryption steps mirror those in standard RLWE decryption plus invertible flows, legitimate decryption recovers m with negligible error (if error bounds are chosen appropriately).

3.2 Mathematical Formulation

Let us formalize the above succinctly. Fix ring parameters $R_q = \mathbb{Z}[X]/(\Phi(X),q)$. Let - $s\in R_q$ be the secret key polynomial (chosen small), - $pk = (a,b=a\cdot s + e)$ be the public key (with random a, error e), - $F_K: R_q^n \to R_q^n$ be the invertible neural transform defined by parameters K.

Encryption(pk, m):

1. Convert plaintext m into vector $\mathbf{x}\in R_q^n$.
2. Compute $\mathbf{y} = F_K(\mathbf{x})$. (This is a permutation of R_q^n.)
3. Sample random small error $\mathbf{r},\mathbf{e}_0,\mathbf{e}_1\in R_q^n$; sample random plaintext-mask $\mathbf{v}\in R_q^n$ (with small entries).
4. Compute RLWE-encryption of $\mathbf{v}$:

$$c_0 = a \cdot \mathbf{r} + \mathbf{e}_0, \qquad c_1 = b \cdot \mathbf{r} + \mathbf{e}_1 + \mathbf{v}.$$

5. Output ciphertext $C = (c=(c_0,c_1),\,\mathbf{c}_2=\mathbf{y} + \mathbf{v} \bmod q)$.

Decryption($sk=(s,K)$, C):

1. Parse $C=(c,\mathbf{c}_2)$ with $c=(c_0,c_1)$. Compute

$$\mathbf{v} = c_1 - s \cdot c_0 \bmod q$$

   (since $b=a s+e$, one gets $c_1 - s c_0 = \mathbf{v} +$ small error, recoverable).
2. Compute $\mathbf{y} = \mathbf{c}_2 - \mathbf{v} \bmod q = F_K(\mathbf{x})$.
3. Compute $\mathbf{x} = F_K^{-1}(\mathbf{y})$ via inverting each affine coupling layer [2].
4. Decode $\mathbf{x}$ to recover plaintext m.

This completes the formal description. The key observation is that the ciphertext seen by an adversary is $(c,\mathbf{y}+\mathbf{v})$, where c hides $\mathbf{v}$ (by RLWE) and $\mathbf{y} = F_K(\mathbf{x})$ is a complex, key-dependent mixing of $\mathbf{x}$.

4. Security Analysis

We now analyze the security of VEINN. Our goal is to argue that VEINN achieves IND-CPA security under the assumption that RLWE is hard. We also discuss the additional hardness assumption introduced by the invertible-network step.

4.1 Security Definitions

We consider the standard *indistinguishability under chosen-plaintext attack (IND-CPA)* definition for public-key encryption. Briefly, an adversary chooses two equal-length messages m_0,m_1, a random bit b is chosen, and the challenger returns a ciphertext encrypting m_b. The adversary must guess b. IND-CPA security means no efficient adversary can guess b with non-negligible advantage over 1/2.

Formally, VEINN's security relies on the *decisional RLWE* assumption: that RLWE samples are pseudorandom. We will show that any adversary breaking IND-CPA on VEINN can be turned into one breaking RLWE.

4.2 Hard Problems

Definition (RLWE Assumption): For a polynomial ring $R_q=\mathbb{Z}_q[x]/(f(x))$, the *Ring-LWE* assumption posits that given polynomial pairs $(a_i, b_i=a_i s + e_i)$ for random a_i and small error e_i, it is hard to recover the secret s, or even to distinguish (a_i,b_i) from uniformly random samples over R_q^2. Equivalently, no efficient (classical or quantum) algorithm can solve the decisional RLWE problem with non-negligible advantage [3] [4].

Definition (Invertible-NN Inversion Problem): Given a ciphertext $(c,\mathbf{c}_2)$ of VEINN, finding the plaintext $\mathbf{x}$ requires inverting the function F_K. More concretely, even ignoring the RLWE component, one faces the problem: *Given $\mathbf{y} = F_K(\mathbf{x})$ (and possibly the public description of F without its secret parameters) find $\mathbf{x}$*. This can be viewed as solving a system of nonlinear (deep) equations defined by s_i,t_i. In general, solving multivariate nonlinear systems over rings is NP-hard, and even over reals/integer domains there are no known polynomial-time algorithms for arbitrary networks. Thus we treat the invertibility as one-way under the assumption that K remains secret [10].

In VEINN, the combined hardness is that an adversary must either break the RLWE encryption (to find $ \mathbf{v}$) *or* invert F_K given only $\mathbf{y}+\mathbf{v}$. Without the secret key, $\mathbf{y}+ \mathbf{v}$ is the sum of a hidden structured vector and a pseudorandom error; separating them is as hard as solving RLWE. Thus, we argue security based on the RLWE assumption and the one-wayness of the INN transform.

4.3 IND-CPA Security Proof

Theorem: If the Ring-LWE assumption holds (in the chosen ring and dimension), then the VEINN scheme is IND-CPA secure.

Proof Sketch: Consider an IND-CPA adversary $\mathcal{A}$ against VEINN. We build a reduction $ \mathcal{B}$ that breaks the RLWE challenge. $\mathcal{B}$ receives a RLWE public key (a,b) (with $b = a\cdot s + e$ hidden secret s) from a challenger. $\mathcal{B}$ uses (a,b) as the public key for VEINN (ignoring the INN component for now). $\mathcal{B}$ also samples a secret INN key K itself (note: $ \mathcal{B}$ *does* know K, but we could also treat K as a part of the VEINN secret that $\mathcal{A}$ does not know). It gives (a,b) to $\mathcal{A}$ as VEINN's public key.

When $\mathcal{A}$ chooses two messages m_0,m_1, $\mathcal{B}$ must return a VEINN encryption of one. $\mathcal{B}$ does the following: it chooses a random challenge bit b^* to decide $m_b*^$ to encrypt, picks a random $\mathbf{v} \in R_q^n$, and then *uses the RLWE challenger* to encrypt $\mathbf{v}$. That is, it forwards $\mathbf{v}$ to the RLWE oracle to get a ciphertext (c_0,c_1). By the RLWE encryption rules, the oracle returns (c_0,c_1) with $c_1 = b\cdot \mathbf{r} + \mathbf{e} + \mathbf{v}$ (plus small noise). $ \mathcal{B}$ then computes $\mathbf{c}*2 = F_K(\mathbf{x}*{b^*}) + \mathbf{v}$ (with $\mathbf{x}_{b*^}$ the encoding of m_{b^*}). It returns $((c_0,c_1), \mathbf{c}_2)$ to $\mathcal{A}$ as the VEINN ciphertext.

- If the RLWE challenger is real (i.e. returns a valid encryption of $\mathbf{v}$), then the distribution of (c_0,c_1) is exactly as in real VEINN encryption. $\mathcal{A}$'s view is a perfect VEINN encryption of m_{b^*}.

- If the RLWE challenger is fake (i.e. returns random c_0,c_1 independent of $\mathbf{v}$), then $ (c_0,c_1)$ is uniform and independent of $\mathbf{v}$. In this case $\mathbf{v}$ is hidden and $ \mathbf{c}*2 = F_K(\mathbf{x}*{b^*}) + \mathbf{v}$ is a one-time pad encryption of $F_K(\mathbf{x}_{b*^})$ by $\mathbf{v}$. Thus $\mathbf{c}*2$ is uniformly random from $\mathcal{A}$'s view (because $F_K(\mathbf{x}*{b^*})$ is fixed, and adding unknown random $\mathbf{v}$ yields uniform bits). Hence the entire ciphertext is random, independent of $m_{b*^}$. In this case $\mathcal{A}$ has no advantage (it sees random).

If $\mathcal{A}$ guesses b^* with non-negligible advantage, $\mathcal{B}$ can use that to distinguish the RLWE oracle: $\mathcal{B}$ outputs "real RLWE" if $\mathcal{A}$ guesses correctly more often than $1/2$, and "random" otherwise. Therefore any $\mathcal{A}$ breaking VEINN implies breaking RLWE. Formally, one shows that $\Pr[\text{Real}] - \Pr[\text{Random}] = \text{Adv}_{\mathcal{A}}$, so RLWE advantage is non-negligible. This concludes that VEINN is IND-CPA under RLWE.

Notes: In the above, we assumed F_K is public or at least known to $\mathcal{A}$ (so $\mathbf{c}_2 - F_K(\mathbf{x}) = \mathbf{v}$). If K is secret and $\mathcal{A}$ does not know F_K, then the scheme is even more opaque. The reduction does not require K to be public; knowing K just enables $\mathcal{B}$ to produce the challenge ciphertext.

Thus, **VEINN achieves IND-CPA security** under the standard RLWE assumption. Intuitively, the one-time mask $\mathbf{v}$ acts like a one-time pad: it hides $F_K(\mathbf{x})$ from the adversary. Without revealing s, the adversary cannot remove $\mathbf{v}$ to inspect $\mathbf{y}=F_K(\mathbf{x})$. Even if $\mathbf{y}$ were somehow known, it is a complicated permutation of $\mathbf{x}$, which (if F_K is secret or complex) gives no obvious information. The formal reduction above ensures all distinguishing advantage must come from breaking RLWE.

4.4 Hardness of INN Inversion

Beyond the above reduction, VEINN introduces an **additional hardness assumption**: that inverting the coupled-layer network without the key is difficult. Concretely, consider an adversary who obtains $\mathbf{c}_2 = F_K(\mathbf{x}) + \mathbf{v}$ but also somehow learns $\mathbf{y} = \mathbf{c}_2 - \mathbf{v} = F_K(\mathbf{x})$ (say the RLWE was broken). Recovering $\mathbf{x}$ from $\mathbf{y}$ then requires inverting F_K. Since each coupling layer is easily invertible *when its parameters are known* [2] [10], the adversary's challenge is that the parameters (s_i,t_i) are unknown. Recovering F_K^{-1} or even any preimage of $\mathbf{y}$ amounts to solving a system of nonlinear equations over R_q.

This can be formalized as a variant of the *multivariate quadratic (MQ) problem*, which is NP-hard in general [3]. Each coupling layer involves exponentiation and addition, which (over a finite field or ring) yields high-degree polynomials. A joint equation for $\mathbf{x}$ is:

$$\mathbf{y} = F_K(\mathbf{x}), \quad F_K(\mathbf{x}) = \Big(x_1 \exp(s_2(\mathbf{x}_2)) + t_2(\mathbf{x}_2),\ x_2 \exp(s_1(x_1')) + t_1(x_1')\Big),$$

which expands to a difficult system. Without knowledge of K, one must also solve for the unknown parameters of the neural networks. In effect, the adversary faces a *joint key-and-message recovery* problem, which is believed infeasible for sufficiently large networks.

While we do not give a full reduction from an established problem to this "Invertible-NN Inversion" problem, we argue that it is at least as hard as general multivariate inversion. Practically, one might use techniques like Gröbner bases or SAT solvers to attempt to recover $\mathbf{x}$ and K, but the complexity grows combinatorially with dimension. In summary, VEINN's security relies primarily on RLWE (which yields formal IND-CPA security) and secondarily on the assumption that F_K is a one-way permutation without its key.

5. Discussion and Comparison

5.1 Parameter Choices and Performance

In VEINN, the main free parameters are the vector dimension n, the number of INN layers ℓ, and the underlying ring modulus q (and RLWE dimension $n_{\rm RLWE}$). These must be chosen to ensure both correctness and security. In practice one might take n large (e.g. hundreds or thousands) to maximize diffusion and hardness of the INN inversion. The RLWE scheme can use standard parameters (e.g. $n_{\rm RLWE}=256$, $q\approx 3329$ as in Kyber) for the same nominal security level. The INN weights s_i,t_i

can be derived from the secret key via a pseudorandom process, avoiding the need to store enormous explicit weight matrices.

Key and ciphertext sizes: The public key is roughly the size of the RLWE public key (e.g. a few kilobytes) plus any description of the INN structure. If the INN architecture (number of layers, splitting pattern) is fixed and public, only the random seeds for K need to be kept secret. The secret key includes the RLWE secret and the INN parameters (or seed). The ciphertext consists of the RLWE ciphertext (c_0,c_1) (several kilobytes at high security) plus an n-dimensional vector $\mathbf{c}_2$ in R_q^n. If $n=512$ and $q\approx 2^{16}$, then $\mathbf{c}_2$ is on the order of $512\times 16 = 8192$ bits (1 KiB). Thus a VEINN ciphertext might be a few kilobytes, comparable to Kyber or NTRU ciphertext sizes.

Computation: Encryption requires one RLWE encryption (a few NTT multiplications) plus ℓ coupling layers of complexity $O(n)$ each (essentially vectorized linear algebra plus exponentials). Decryption similarly inverts these. This is likely slower than pure lattice schemes (which are just polynomial multiplies), but parallelizable and using simple elementwise operations. A performance evaluation is left for future work, but conceptually VEINN is heavier than Kyber but potentially lighter than, say, McEliece (which has huge matrices).

5.2 Comparison to NIST PQC Candidates

We now compare VEINN qualitatively against leading PQC approaches:

- **Lattice-based (Kyber/NTRU):** Like VEINN, these rely on (R)LWE hardness [3] [4]. However, standard RLWE schemes use *linear* encryption (matrix-vector operations with small noise), whereas VEINN inserts a *nonlinear* layer F_K. One could view VEINN as adding a symmetric-key “preprocessing” step to the lattice encryption. Unlike AES or other block ciphers, F_K is invertible by design and tied to the key. This additional diffusion may resist certain algebraic attacks (we are not aware of any published attacks against coupling flows). On the flip side, VEINN inherits all lattice pitfalls: one must choose q and error distribution carefully, or face decryption failures. (NIST-approved Kyber parameters avoid decryption error by design.)

- **Code-based (Classic McEliece):** Code-based KEMs do not use invertible networks, and their security is based on decoding random linear codes [8]. VEINN’s security does *not* rely on code assumptions, so we do not directly use McEliece hardness. However, comparing to McEliece highlights the trade-off: Classic McEliece has extremely large keys (hundreds of KB) [9] but extremely small ciphertexts (tens or hundreds of bytes). VEINN’s keys are much smaller (on the order of KB) but ciphertexts include an entire vector (similar size to keys). Thus VEINN is closer to lattice KEMs in key/ciphertext size.

- **Security Level:** Assuming RLWE is hard (e.g. 128-bit security), VEINN achieves the same baseline security. The INN layer does not appear to weaken security; if anything, it hides any linear leakage by mixing coordinates nonlinearly. We caution that, unlike AES or SPN ciphers, the INN in VEINN is not designed by cryptographic standards (no S-box analysis, etc.), so unforeseen weaknesses could exist. However, the cryptographic reduction to RLWE means that any key-recovery attack must either break RLWE or invert the INN permutation. If one models the INN as a random high-dimensional permutation (via its random parameters), then each ciphertext is essentially a random mask of this permutation output – akin to a one-time pad.

- **Hybrid Approaches:** VEINN resembles a *hybrid encryption* where the lattice scheme provides a random pad $\mathbf{v}$, and the INN acts like a symmetric cipher. (One could even view F_K as a keyed permutation similar to a block cipher on $\mathbb{Z}_q^n$.) In this view, VEINN is not truly a stand-alone public-key scheme, but a composition of a public-key (to send $\mathbf{v}$) and a symmetric transform (to mix m). This is analogous to (but more complex than) KEM+DEM in hybrid encryption standards.

- **Quantum Considerations:** Both RLWE and multivariate equations are believed (though not proved) to be hard for quantum algorithms [3] [4]. To the extent VEINN adds only *classical* operations (no structure exploitable by Shor), it should inherit quantum resistance from RLWE. If a quantum algorithm could efficiently invert a large random neural network, that would imply dramatic breakthroughs in quantum ML, so we assume that is unlikely.

5.3 Strengths and Novel Aspects

- **Nonlinearity:** VEINN introduces genuine nonlinearity into encryption. Most lattice schemes are linear-plus-noise; here the plaintext bits undergo multiple nonlinear transformations. This may thwart linearization attacks or algebraic solving, since an attacker must handle exponentials (or high-degree polynomials) from s_i,t_i networks.

- **Invertibility Guarantee:** The use of invertible networks ensures that no information is lost in encryption (aside from the added noise). Unlike probabilistic schemes that rely solely on noise, VEINN maintains a one-to-one map (for fixed key) between message vectors and ciphertext vectors, enabling exact (or near-exact) decryption.

- **Versatility:** By varying the network depth ℓ and width n, one can tune the trade-off between diffusion and efficiency. Moreover, one could in principle train the INN (or parts of it) for specific distributions, though VEINN as encryption would fix the networks at key generation time.

5.4 Open Questions and Cautions

- **Key Structure Leakage:** If the INN parameters K are fully secret, the adversary only sees $(c, \mathbf{y}+\mathbf{v})$ but not $F_K(\cdot)$ itself. If K were partially public (e.g. the network weights), then the adversary sees $\mathbf{y}+ \mathbf{v}$ where $\mathbf{y}=F_K(\mathbf{x})$ can be computed if one knew $\mathbf{x}$. In either case, the hiding is via $\mathbf{v}$. A thorough study would consider whether any partial leakage of K (e.g. its architecture) could weaken security.

- **Parameter Selection:** VEINN requires careful choice of noise distribution to ensure decryption success but maintain security. If noise is too small, RLWE may be vulnerable; if too large, message recovery fails. Likewise, the INN invertibility relies on numerical stability (clipping s_i values), though in a finite ring arithmetic these issues are different.

- **Provable One-Wayness:** We do not have a reduction from a known problem to the invertibility of random INNs. This remains a heuristic assumption. If a breakthrough algorithm for solving deep nonlinear systems were found, it could impact VEINN uniquely (beyond just RLWE).

- **Implementation Details:** Embedding plaintext bits into R_q^n in a reversible way (e.g. bit-to-coeff mapping) must be done so that F_K's domain matches. Also, finite-field analogues of $\exp(s)$ may require careful definition (in practice one would work over a field or use an approximation).

Overall, VEINN should be viewed as an experimental concept: a **novel architectural combination** that leverages recent machine-learning ideas for cryptographic use. Its non-triviality comes from melding continuous invertible flows with discrete encryption.

6. Conclusion

We have presented **VEINN**, a vector-based encryption scheme that integrates invertible neural networks into a lattice-based encryption framework. By embedding plaintext into high-dimensional vectors and applying multiple key-dependent affine coupling layers, VEINN adds a nonlinear, diffusion-rich layer of encryption on top of standard RLWE/NTRU primitives. We defined the scheme formally, provided mathematical descriptions of the transformations, and proved IND-CPA security under the RLWE assumption. In comparison to existing NIST post-quantum schemes (e.g. CRYSTALS-Kyber, NTRU, Classic McEliece), VEINN offers a distinct approach: the ciphertext is partly the output of a one-way neural permutation.

VEINN's strengths include its novelty (combining two research areas), potential resistance to linear attacks, and exact invertibility for legitimate users. Its security reduces to well-studied hard problems, plus the new hardness of inverting random neural flows. Future work includes optimizing parameters, evaluating performance, and rigorously analyzing the invertible-net hardness. We hope this work encourages further exploration of machine-learning-inspired cryptographic designs.

Acknowledgments: The authors thank the cryptography and ML communities for inspiration, and acknowledge that this is an experimental proposal.

References

- **Ring-LWE and LWE Hardness.** The RLWE problem (a ring-based variant of LWE) is backed by lattice hardness assumptions [3] [4]. Lyubashevsky, Peikert and Regev introduced RLWE in Eurocrypt 2010. No polynomial attack on LWE/RLWE is known, and in fact *no poly-time attack has been found against LWE yet* [4]. (RLWE improves efficiency by a factor n, at the cost of relying on ideal lattices [3] [4].) Notably, an estimated **30% of NIST PQC proposals** are lattice/RLWE-based [11].

- **NTRU Cryptosystem.** The original NTRU scheme [6] [7] is a ring-based PKC using polynomial multiplication modulo a prime. Hoffstein, Pipher and Silverman's NTRU from 1998 achieves high speed using only additions and shifts [6]. Its public key size is roughly $N \log_2 q$ bits where N is ring degree [6]. NTRU (and its later NTRU-Prime variant [12]) have undergone extensive cryptanalysis but remain strong.

- **Invertible Neural Networks (Coupling Layers).** Dinh *et al.* introduced *Real NVP* [1], showing that affine coupling layers yield a flexible invertible mapping with efficient inverse. Specifically, given input $(\mathbf{u}_1,\mathbf{u}_2)$, the output is

$$\mathbf{v}_1 = \mathbf{u}_1 \odot \exp(s_2(\mathbf{u}_2)) + t_2(\mathbf{u}_2), \quad \mathbf{v}_2 = \mathbf{u}_2 \odot \exp(s_1(\mathbf{v}_1)) + t_1(\mathbf{v}_1),$$

which is exactly invertible by algebraic manipulation [2] [1]. Crucially, the scaling/translation networks s_i,t_i can be arbitrarily complex, yet the layer remains invertible [10]. These invertible flows are now widely used for generative modeling.

- **Classic McEliece (Code-Based).** Bernstein *et al.*'s Classic McEliece (NIST Round 3 finalist) uses Goppa codes. Its security rests on syndrome-decoding hardness [8]. A typical parameter (e.g. Classic-McEliece-348864) has public key ≈261,120 bytes, secret key 6,492 bytes, ciphertext 96 bytes [9]. NIST selected it as an alternate candidate [13].

- **NIST PQC Standardization.** NIST's third-round report announced CRYSTALS-Kyber (an LWE-based KEM) as the primary public-key standard [5]. Alternate KEMs moving to Round 4 include BIKE (code-based), Classic McEliece (code-based), HQC (code-based), and SIKE (isogeny-based) [13]. This demonstrates the diversity of PQC: lattice-based schemes predominate (including Kyber, NTRUPrime), but code-based and isogeny-based remain important.

<!-- Citations in text use the format [cursor†Lx-Ly], corresponding to open references in the browsing context above. -->

---

[1] [10] [1605.08803] Density estimation using Real NVP
https://ar5iv.labs.arxiv.org/html/1605.08803

[2] Analyzing Inverse Problems with Invertible Neural Networks - VLL
https://web.archive.org/web/20241006132515/https://hci.iwr.uni-heidelberg.de/vislearn/inverse-problems-invertible-neural-networks/

[3] [4] [7] [8] [11] [12] irishmathsoc.org
https://www.irishmathsoc.org/bull86/pdfs/Articles/Blanco/Blanco.pdf

[5] [13] Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process
https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf

[6] NTRU: A ring-based public key cryptosystem
https://www.ntru.org/f/hps98.pdf

[9] Classic McEliece | Open Quantum Safe
https://openquantumsafe.org/liboqs/algorithms/kem/classic_mceliece.html