

A Cryptographic Perspective on Fingerprinting Machine Learning Model Weights

Huijia Lin^a and Kameron Shahabi^a

^aPaul G. Allen School of Computer Science & Engineering, University of Washington
{rachel,kshahabi}@cs.washington.edu

October 5, 2026

Abstract

We introduce a cryptographic framework for *fingerprinting* machine learning models, enabling model providers to train models that can later be attributed to them. Our framework captures an honest setting in which providers distribute base models that users may subsequently modify to adapt them to their own applications (e.g., through finetuning). We formalize three properties for fingerprinting schemes: (1) *quality preservation* or *undetectability*: fingerprinted models must remain computationally indistinguishable from models produced by an underlying training algorithm, (2) *robustness*: the fingerprint must remain detectable even after the model weights are subject to permitted modification, and (3) *unforgeability*: no computationally bounded trainer can forge the fingerprint except by applying a permitted modification to an existing fingerprinted model. Under the continuous learning with errors assumption [BRST21], we construct a robust and unforgeable fingerprinting scheme for modifications in ℓ_2 . The scheme achieves undetectability with respect to any given *noisy* training algorithm, meaning any algorithm whose output weights can be decomposed into independent Gaussian and non-Gaussian components.

Along the way, we introduce two new technical tools. The first is a form of steganography; we develop a method for undetectably modifying a noisy training algorithm so that its output weights secretly encode a message. The embedded message remains decodable even after bounded ℓ_2 perturbations to the weights. Second, we introduce the *pancake alignment* problem, a “semi-search” variant of homogeneous continuous learning with errors (hCLWE). Given hCLWE samples generated from a random secret, the goal is to recover a direction that is sufficiently aligned with the secret. We provide evidence for the hardness of pancake alignment in certain parameter regimes via reductions from continuous learning with errors.

Contents

1	Introduction	3
1.1	Our results	5
1.2	Related work	10
2	Technical Overview	11
3	Preliminaries	14
3.1	Lattices and discrete Gaussians	15
3.2	Cryptography	17
3.3	Continuous Learning with Errors	17
4	New Definitions	19
4.1	Fingerprinting	19
4.2	KDS Gaussian pancake and Pancake Alignment	21
5	Robust Fingerprinting	23
5.1	Noisy training algorithms	23
5.2	Embedding messages into weights	25
5.3	Fingerprinting construction	27
5.3.1	Construction guarantees	28
5.3.2	Proofs	30
5.4	Discussion	32
6	Unforgeable Fingerprinting	33
6.1	Unforgeable fingerprinting construction	34
6.2	Hybrids	36
6.3	Security proofs	38
7	Pancake Alignment Cryptanalysis	42
7.1	An efficient attack for $t < 1$	43
7.2	Reduction from KDS Gaussian Pancake for $t = \tilde{\Omega}(\sqrt{\lambda})$	44
7.2.1	The reduction for a single secret	45
7.2.2	Reducing low-dimensional PA to high-dimensional PA	47
7.2.3	Reducing single-secret PA to multi-secret PA	48
7.2.4	Facts about the uniform distribution on the sphere restricted to a subspace	49
7.3	Reduction from KDS Gaussian Pancake for $t = \frac{1}{\text{advantage}}$	50
7.3.1	Covariance estimation	52
8	Hardness of Key-Dependent Shifted Gaussian Pancake	53
8.1	CLWE with key-dependent shift	53
8.2	Reducing normalized binary secret CLWE to CLWE with KDS	54
8.2.1	Making secrets uniform over the unit sphere	55
8.3	Reducing CLWE with KDS to KDS Gaussian Pancake	55
8.3.1	Separating the parameters and the dimension	56
8.3.2	Extension to multiple secrets	56
A	Acknowledgments and AI Usage	61
B	Noise Level of Natural Training Algorithms	61
B.1	Accumulating noise during training	61
B.2	Noisy training algorithms	63

1 Introduction

In recent years, AI models have begun to pass through complex *supply chains*, where they may be processed, modified, or hosted by multiple parties [CVS23; WN23; NS26]. This makes it difficult to reliably attribute a model to the party or parties responsible for producing it, which is important both for protecting intellectual property (IP) and for holding model providers accountable for their models’ behavior [MMML+25]. To address this, *fingerprinting*, or watermarking the weights of a machine learning model, has emerged as a prominent tool for certifying model ownership [UNSS17; WWZY20; XWMK+24; PKA25; NHBS+26]. The goal is to train a base model $\mathbf{w} \in \mathbb{R}^n$ whose origin can later be verified, even after the model has been adapted to downstream tasks by other parties. For example, a model provider may distribute a general-purpose, pretrained model to a customer, who subsequently fine-tunes it. An auditor may then wish to verify that the resulting model originated from the provider, without requiring the customer to preserve the original model or training procedure.

Fingerprinting schemes should satisfy several desirable properties [XWMK+24; NHBS+26; RCS26]. The fingerprinted models should (1) remain high in quality, (2) remain fingerprinted even after undergoing permitted modifications (e.g., fine-tuning or compression), and (3) be distinguishable from independently trained, un-fingerprinted models¹. Existing approaches provide empirical evidence for these properties, but are largely heuristic and lack formal guarantees [XWMK+24; NHBS+26; RCS26; ZWHX+24; UNSS17; WWZY20; PKA25; WXLX+25]. This limits the extent to which they can be relied upon, particularly in sensitive settings such as legal applications.

In this work, we provide a *cryptographic* treatment of fingerprinting. Cryptography provides a natural language for formally capturing the desired properties listed above. For example, rather than informally requiring that fingerprinted models “remain high in quality,” we can require they be *computationally indistinguishable* from models produced by an ordinary training algorithm. Cryptography also provides powerful tools for reasoning and providing formal guarantees about “permitted model modifications”. Rather than attempting to enumerate every transformation that an honest downstream user might perform, we define a broad class of permissible modifications and require the fingerprint persist even when these modifications are chosen adversarially.

Our model. We consider a setting in which a model provider trains multiple models and distributes their weights to multiple customers. We assume that customers are honest: they do not intentionally attempt to remove the provider’s fingerprint, but may nevertheless modify the model for their own purposes. In order to certify ownership over these models, the model provider trains them using a *fingerprinting scheme*, consisting of efficient algorithms ($\text{Gen}, \text{FP}, \text{Ver}$). Before any training, they first sample a (secret) fingerprinting key $\text{sk} \leftarrow \text{Gen}(1^\lambda)$, *independent* from any trained model. They then produce weights using the fingerprinting algorithm, which takes in a training data distribution D and training specification spec (e.g., the training hyperparameters), and outputs fingerprinted weights $\mathbf{w}_{\text{FP}} \leftarrow \text{FP}^D(\text{sk}, \text{spec})$. Finally, the verification algorithm $\text{Ver}(\text{sk}, \mathbf{w}')$ ² is able to distinguish fingerprinted weights (correctness) from weights that are *independent* of the key sk (soundness). In addition, we ask for the following,

- *Undetectability*: To computationally bounded users without the secret key, models produced by the fingerprinting algorithm are indistinguishable (given white-box access to the weights) from models produced by an underlying training algorithm. To the best of our knowledge, this is the first work to introduce undetectability to fingerprinting. In fact, the spirit of our approach, which modifies the training procedure to hide a detectable signal in the model parameters, has previously been described as *invasive* [WXLX+25], as prior attempts to do so resulted in a detectable modification to the model’s output distribution [WK19]. However, undetectability allows us to take such an approach while providing a strong, cryptographic guarantee of quality preservation (against all efficient benchmarks).
- *Robustness*: Following related work on watermarking open-source models [CGMR24], we provide a formal treatment of robustness: even if a customer receives a fingerprinted model $\mathbf{w}_{\text{FP}}$ and modifies it in a *permissible manner* to obtain weights $\mathbf{w}'$, the verifier $\text{Ver}(\text{sk}, \mathbf{w}')$ will still accept. To formalize

¹In the fingerprinting literature, these properties are often referred to as *harmlessness*, *persistence*, and *reliability* or *uniqueness*, respectively.

²We consider white-box verification, where the verifier sees the candidate weights in the full.

the class of permitted modifications, we use a two-input predicate $\Phi : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \{0, 1\}$, where $\Phi(\mathbf{w}_{\text{FP}}, \mathbf{w}') = 1$ indicates that $\mathbf{w}'$ is within the permitted modification of $\mathbf{w}_{\text{FP}}$ ³.

- *Unforgeability*: If a separate model trainer produces a model $\mathbf{w}'$ that is sufficiently *far*, according to some two-input predicate Ψ , from every fingerprinted model they have observed, then the verifier will reject $\mathbf{w}'$ ⁴. To the best of our knowledge, this is the first work in fingerprinting to formalize this form of unforgeability, which considers forgeries only on far weights. It significantly strengthens soundness, because it protects honest trainers that unintentionally produce models correlated with the fingerprinting key $\mathbf{sk}$ —for example, by training on data scraped from the internet that happened to be produced by a fingerprinted model. While soundness no longer guarantees that the verifier rejects such a model, unforgeability guarantees rejection so long as the new model is far from the fingerprinted model whose outputs were used for training.

Readers familiar with the literature on *watermarking* AI-generated outputs may already recognize formal treatments of these properties [CGZ24; CG24; LSS26]. While watermarking and fingerprinting are related, their settings and applications are different. In watermarking, the goal is to generate certified *outputs* using an already trained model; to later answer the question, *were these outputs generated by an AI model?* In contrast, fingerprinting aims to certifiably train a *model* such that, later, given any set of weights, one can answer questions such as: *did these weights originate from a particular provider?* or *are these weights a modification of a particular model?* Formalizing these properties for fingerprinting is new and, as we will discuss shortly (Sections 1.1 and 2), achieving them requires completely different techniques.

Applications One may wonder why we need a secret key to determine whether a candidate model is within a permitted modification of a fingerprinted model. Why would a model provider not simply retain all of its trained models and, during verification, check whether the candidate weights are sufficiently close to one of the models in its database? There are several benefits to instead using a fingerprinting scheme. First is *efficiency*: the secret key $\mathbf{sk}$ is more *succinct* for storage and enables *faster verification*, eliminating the need to maintain a potentially large database of models. Second, the secret key can be *distributed* without revealing information about the provider’s models: although the fingerprinted models depend on $\mathbf{sk}$, the key itself is sampled independently before any training, so observing $\mathbf{sk}$ alone reveals no information about the models. Finally, a secret key provides a *strong claim of ownership*. In some scenarios, providers may discard model weights after training them—for instance, due to copyright agreements⁵ or for versioning purposes. With a fingerprinting scheme, so long as the provider’s key is retained, recovered copies of the old weights can still be attributed back to them. We describe two general settings that benefit from these properties.

Third-party auditing. One or many model providers may outsource fingerprint verification to an auditor (e.g., the EU AI Office [Eur24]). Without fingerprinting, the providers would need to give the auditor access to all of their models, which is both inefficient and undesirable. Although the auditor is trusted to perform verification, they may not be trusted with the providers’ proprietary weights. A fingerprinting scheme instead allows the auditor to store only a succinct secret key $\mathbf{sk}$ per-model or per-provider. This enables a natural workflow in which the auditor is given access to a model *only when* it is flagged downstream—for example, because it is exhibiting problematic behavior. Since the flagged model is already the subject of a dispute, disclosing it to the auditor may be acceptable; crucially, the auditor need not receive any of the providers’ other, undisputed models. Moreover, if the application requires that even disputed models remain private, the model holder and auditor can instead perform verification using secure multiparty computation.

IP protection via strong claims of ownership. Consider a scenario in which a model provider’s weights are illegally posted online, for example, by a disgruntled employee or by a competitor posing as an honest customer. With our fingerprinting setup, the provider may have previously distributed or committed to a

³For example, for our construction Φ outputs 1 when the two models are sufficiently close in ℓ_2 distance.

⁴Ideally the predicates $\Psi = \Phi$, guaranteeing rejection when weights fall *exactly* outside the permitted modification threshold. However, our construction achieves a *gap* variant, where $\Psi(\mathbf{w}, \mathbf{w}') = 0 \Rightarrow \Phi(\mathbf{w}, \mathbf{w}') = 0$. This is still meaningful; rejection is at least guaranteed for weights *far enough* outside the permitted threshold.

⁵For instance, a customer may commission model training on sensitive data, under the agreement that the provider discards the weights upon delivery.

particular secret key sk . The provider can then open the commitment and demonstrate that the published weights carry its fingerprint. Coupled with unforgeability, this establishes that the published model is close to a model produced using the provider’s secret key, providing evidence of the provider’s prior possession of the model.

1.1 Our results

Our main conceptual contribution is in providing formal definitions of undetectability, robustness, and unforgeability for fingerprinting (Section 4). On the technical side, we construct a fingerprinting scheme for a broad class of training algorithms that we call *noisy training algorithms* (Definition 5.1). In short, these are training algorithms that produce weights which are subject to additive Gaussian noise. Under the *continuous learning with errors* (CLWE) assumption [BRST21], we construct an ℓ_2 -robust and unforgeable fingerprinting scheme (Sections 5 and 6). Robustness guarantees that verification accepts on weights that are ℓ_2 -close to fingerprinted weights, while unforgeability guarantees that verification rejects on weights that are ℓ_2 -far from all previously fingerprinted weights.

Along the way, we develop two new technical tools. The first is a new computational problem, the *pancake alignment* problem (Definition 4.8), which we show is hard in certain parameter regimes via a reduction from CLWE. We believe this problem may be of independent interest, and our preliminary cryptanalysis suggests that it may remain hard over a broader range of parameters than those covered by our reductions (Section 7). The second is a method for *embedding* messages into naturally trained weights such that the weights are indistinguishable from standard training. Assuming the hardness of the pancake alignment problem, the message remains decodable even in the presence of bounded adversarial perturbations in ℓ_2 (Section 5.2).

Defining fingerprinting. We define a fingerprinting scheme $(\text{Gen}, \text{FP}, \text{Ver})$ (Definition 4.2) with respect to an underlying training algorithm Train , which takes as input a distribution D and training specification spec and outputs trained weights $\mathbf{w} \leftarrow \text{Train}^D(1^\lambda, \text{spec})$. As noted earlier, we require that, with the secret key $\text{sk} \leftarrow \text{Gen}(1^\lambda)$, a verifier can distinguish fingerprinted weights from weights independent of sk . That is, we require correctness: for any D and spec , $\text{Ver}(\text{sk}, \text{FP}^D(\text{sk}, \text{spec})) = 1$, and soundness: for any $\mathbf{w}'$ independent of sk , $\text{Ver}(\text{sk}, \mathbf{w}') = 0$.

Moreover, we require undetectability, which says *without the secret key* sk , no customer should be able to distinguish fingerprinted weights from weights produced by the original training algorithm. This subsumes any efficiently computable test of quality, such as performance on a test dataset.

Definition 1.1 (Undetectability, informal). *No PPT adversary $\mathcal{A}$ can distinguish between an oracle $\mathcal{O}_{\text{FP}}(\text{sk}, \cdot, \cdot)$, for a randomly generated $\text{sk} \leftarrow \text{Gen}(1^\lambda)$, and an oracle $\mathcal{O}_{\text{Train}}(\cdot, \cdot)$. The oracles take in as input a training distribution D and specification spec , and return weights from $\text{FP}^D(\text{sk}, \text{spec})$ and $\text{Train}^D(1^\lambda, \text{spec})$ respectively.*

We also want to allow an honest user to make permissible modifications to fingerprinted weights while maintaining verification. In the simplest case, a user receives fingerprinted weights $\mathbf{w}$ and produces a modification $\mathbf{w}'$; if $\Phi(\mathbf{w}, \mathbf{w}') = 1$, meaning $\mathbf{w}'$ is a permitted modification of $\mathbf{w}$, verification should still accept. We strengthen this by allowing the user access to a fingerprinting oracle, so they may interact with *multiple* fingerprinted models. Verification then accepts so long as $\mathbf{w}'$ is Φ -close to *any* of these models. This subsumes the simple case above, and also captures scenarios where a user’s model $\mathbf{w}'$ is correlated with multiple fingerprinted models, such as when a user finetunes fingerprinted weights $\mathbf{w}^{(1)}$ using outputs from a second fingerprinted model $\mathbf{w}^{(2)}$.

Definition 1.2 (Robustness, informal version of Definition 4.3). *For any two-input predicate $\Phi : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \{0, 1\}$ indicating permissible distance, a fingerprinting scheme $(\text{Gen}, \text{FP}, \text{Ver})$ is Φ -robust if for any PPT adversary outputting $\mathbf{w}' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{FP}}(\text{sk}, \cdot, \cdot)}$,*

$$\Phi(\mathbf{w}, \mathbf{w}') = 1 \text{ for some } \mathbf{w} \in \mathcal{W} \implies \text{Ver}(\text{sk}, \mathbf{w}') = 1$$

with overwhelming probability over $\text{sk} \leftarrow \text{Gen}(1^\lambda)$ and the randomness of the adversary and its oracle. Here, $\mathcal{W}$ is the set of fingerprinted weights observed by $\mathcal{A}$ through its queries to the oracle $\mathcal{O}_{\text{FP}}$.

Permitting modifications motivates a unique form of “robust” unforgeability, first observed in the watermarking literature [LSS26]. Typically, in cryptography, unforgeability requires verification to reject under *any* modification to the object of interest. In contrast, in many ML settings, we want to distinguish *slightly modified* objects from *significantly different* ones. For example in fingerprinting, we want to allow robust verification, but we do not want to accidentally attribute weights that are *outside* the modification threshold to the secret key owner. That is, an honest trainer who can observe weights via a fingerprinting oracle $\mathcal{O}_{\text{FP}}(\text{sk}, \cdot, \cdot)$ shouldn’t be able to produce new weights that verify despite being *far* from all the fingerprinted weights. This threat model subsumes users who see fingerprinted weights indirectly through their outputs (e.g., by scraping data from the internet).

Definition 1.3 (Unforgeability, informal version of Definition 4.4). *For any two-input predicate $\Psi : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \{0, 1\}$ indicating permissible distance, a fingerprinting scheme $(\text{Gen}, \text{FP}, \text{Ver})$ is Ψ -unforgeable if for any PPT adversary outputting $\mathbf{w}' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{FP}}(\text{sk}, \cdot, \cdot)}$,*

$$\Psi(\mathbf{w}, \mathbf{w}') = 0 \text{ for all } \mathbf{w} \in \mathcal{W} \implies \text{Ver}(\text{sk}, \mathbf{w}') = 0$$

with overwhelming probability over $\text{sk} \leftarrow \text{Gen}(1^\lambda)$ and the randomness of $\mathcal{A}$ and $\mathcal{O}_{\text{FP}}$.

Next, we show that the above definitions can be satisfied under standard assumptions. Before that, we diverge momentarily into sharing a perspective on the use of cryptography in machine learning; a technically-minded reader can skip the following paragraph.

A perspective on the cryptographic treatment of honest behavior. In traditional cryptography, correctness is usually easy to achieve since “honest behavior” is fully prescribed by the cryptographic schemes and protocols. In contrast, many challenging problems in human-AI systems arise because permissible behavior *is not* fully prescribed. Often, we want to allow *honest parties* to modify the object of interest with some degree of freedom—in fingerprinting, a general-purpose base model may be fine-tuned; in watermarking, a generated text template may be edited; and in generalization, deployment inputs may be different from the training data. Whether cryptography can help in all such settings remains open, but it at least offers a methodology for providing guarantees against broad classes of behaviors, no matter if those behaviors are malicious or honest. That said, it is worth noting that there is a slight shift of emphasis. In traditional cryptography, it is often justified to consider a threat model that is stronger than one might initially expect, at the cost of other desiderata such as functionality or efficiency. In ML settings, it is instead often valuable to capture reasonable, honest behaviors while being minimally intrusive to the surrounding system.

Coming back to our definitions of robustness and unforgeability, we believe it is reasonable for honest users to produce new weights that have dependency on the weights of multiple fingerprinted models, hence we provide the adversary with access to the fingerprinting oracle. One could also consider a stronger definition in which the adversary additionally has access to a verification oracle. However, from the perspective of honest modification, this is arguably less natural: an honest user should not be adapting a model with the intent of influencing verification. We work with the current definitions and leave the exploration of alternatives to future work.

A fingerprinting scheme for noisy training algorithms. We construct an ℓ_2 robust and unforgeable fingerprinting scheme with respect to any *noisy training algorithm* (Definition 5.1). For a noise level σ and dimension n , a training algorithm Train is in the class (σ, n) -NoisyTrain if training can equivalently be decomposed as first sampling some n -dimensional underlying weights $\mathbf{w} \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$, then sampling independent Gaussian noise $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$, and outputting $\mathbf{w} + \mathbf{g}$ as the final weights⁶. At first glance, a training algorithm that adds noise to the final parameters may seem counterintuitive. However, prior work has shown that injecting noise during and after training can help achieve many useful guarantees, including differential privacy [ACGM+16], generalization [Bis95], adversarial robustness [HRF19; CRK19], and exploration in reinforcement learning [PHDS+17]. In Section 5.1, we provide more detailed examples, and in Section B, we show that, for certain training algorithms, noise added *during* training can propagate to the final parameters, allowing us to represent such algorithms as noisy training algorithms.

⁶The actual class NoisyTrain is slightly more general than this, and only requires a *subset* of coordinates be decomposable into a weight component and a Gaussian component; for this section we will consider this simplification.

Theorem 1.1 (Robust fingerprinting, informal version of Corollary 5.5.1). *Let Train be any training algorithm in the class (σ, n) -NoisyTrain. Assume CLWE with binary secrets is hard (Definition 3.7) in dimension λ . Then for some $\Delta = \tilde{\Theta}(\frac{\sigma\sqrt{n}}{\lambda^{3/4}})$, there exists an undetectable fingerprinting scheme w.r.t Train that is $\text{L2}(\Delta)$ -robust, where the predicate $\text{L2}(\Delta)$ outputs 1 iff its inputs are within Δ in ℓ_2 -distance.*

The security of our fingerprinting scheme relies on (normalized) binary-secret CLWE⁷, meaning the secrets are drawn uniformly from $\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda$. Prior work [GVV22] has shown that this variant is hard assuming LWE with binary secrets is hard, which in turn follows from the hardness of standard LWE [Mic18].

We now interpret Theorem 1.1 quantitatively. Recall that σ is the standard deviation of the Gaussian noise $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$ that the training algorithm Train adds to its base weights $\mathbf{w}$, producing final weights $\mathbf{w} + \mathbf{g}$. Undetectability then guarantees that fingerprinted weights are indistinguishable from $\mathbf{w} + \mathbf{g}$. Note that by standard concentration, $\|\mathbf{g}\|_2 \approx \sigma\sqrt{n}$. In typical applications, the noise level σ is therefore set small enough so that the noise does not dominate the weights, $\sigma\sqrt{n} = o(\|\mathbf{w}\|_2)$, while still large enough to obtain the benefits of noisy training.

Robustness then guarantees that removing the fingerprint requires an ℓ_2 perturbation of magnitude roughly $\|\mathbf{g}\|_2/\lambda^{3/4}$, where λ is the dimension that CLWE is hard in. It is worth noting that we can instantiate our construction with an instance of CLWE that has relatively small dimension λ (e.g., $\lambda = 512$, see Remark 5.4 for details). In comparison, the number of weights n in a modern machine learning model may be on the order of millions or billions [GJJP+24]. Therefore, the robustness threshold $\|\mathbf{g}\|_2/\lambda^{3/4}$ may still be substantial in typical settings, where $\|\mathbf{g}\|_2 \approx \sigma\sqrt{n} \gg \lambda^{3/4}$. Indeed, many natural model modifications either affect only a subset of weights (e.g., fine-tuning the last $k \ll n$ weights) or perturb all weights only slightly (e.g., quantization for resource-constrained devices [LGH24]). In the former case, concentrating an $\|\mathbf{g}\|_2/\lambda^{3/4}$ magnitude perturbation on only $k \ll n$ weights permits substantially larger changes per weight. In the latter, the total perturbation should remain smaller than the training noise, since larger changes may begin to degrade model quality. We refer the reader to Section 5.4 for further discussion.

We also show that, under additional assumptions, one can simultaneously guarantee ℓ_2 -unforgeability.

Theorem 1.2 (Unforgeable fingerprinting, informal version of Theorem 6.1). *Let Train be any training algorithm in the class (σ, n) -NoisyTrain. Assume 1) CLWE with binary secrets is hard in dimension λ , 2) symmetric key encryption exists, and 3) pseudorandom functions exist. Then for some $\Delta = \tilde{\Theta}(\frac{\sigma\sqrt{n}}{\lambda^{3/4}})$, there exists an undetectable fingerprinting scheme w.r.t Train that is $\text{L2}(\Delta)$ -robust and $\text{L2}(\tilde{\Theta}(\Delta + \sigma\sqrt{\lambda}))$ -unforgeable.*

Ideally, the robustness and unforgeability thresholds would be the same, to ensure that the verifier accepts if and only if the candidate weights are Δ -close in ℓ_2 to some fingerprinted weights. Our construction, however, incurs a small gap, guaranteeing that the verifier accepts weights that are Δ -close in ℓ_2 , but rejects weights that are more than roughly $(\Delta + \sigma\sqrt{\lambda})$ -far from all fingerprinted weights. We make no guarantees for weights whose distance falls between these thresholds. However, as we discussed earlier, because the dimension of the weights $n \gg \lambda$, this $\sigma\sqrt{\lambda}$ gap is relatively small.

Heuristic potential. Our guarantees apply to any model produced by a noisy training algorithm, whose weights can be represented as a base model with additive Gaussian noise. It may be the case, however, that the weights are not exactly of this form, but may still be sufficiently random due to randomness in the data collection or training process. Our construction admits a heuristic approach for fingerprinting such weights: treat the algorithm Train as a noisy training algorithm by using its output $\mathbf{w}_{\text{random}} \leftarrow \text{Train}^D(1^\lambda, \text{spec})$ as the Gaussian component $\mathbf{g}$ and let the base weights $\mathbf{w} = 0$. Under the assumption that the distribution of $\text{Train}^D(1^\lambda, \text{spec})$ is Gaussian with noise level σ , the fingerprinted weights would then inherit all our formal guarantees; heuristically, one might expect them to hold approximately if the training algorithm Train 's output distribution is “close enough” to Gaussian.

New tool: pancake alignment. As an interface between the hardness of CLWE and robust fingerprinting, we introduce the pancake alignment problem. Before defining it, we must first discuss the (multi-secret) Gaussian pancake distribution (Definition 4.6), also known as the homogeneous CLWE distribution [BRST21]. For parameters $\gamma \in \mathbb{R}$ and $\beta \ll 1$, a shift $\mathbf{z} \in [0, 1)^\ell$, and orthonormal secrets $\mathbf{S} = (\mathbf{S}[1], \dots, \mathbf{S}[\ell]) \in \mathbb{R}^{n \times \ell}$,

⁷The original formulation of CLWE [BRST21] draws secrets uniformly from the unit sphere.

the Gaussian pancake distribution generates samples that look standard Gaussian in most directions, but lie near a discrete “pancake” in each secret direction:

$$\mathbf{y} \leftarrow D_{\mathbb{R}^n, 1} \quad \text{conditioned on} \quad \gamma \mathbf{S}^\top \mathbf{y} + \mathbf{e} = \mathbf{z} \pmod{1}, \quad \text{where } \mathbf{e} \leftarrow D_{\mathbb{R}^\ell, \beta}.$$

Prior work [BRST21; BNHR22] has shown that assuming CLWE is hard, then for any fixed $\mathbf{z} \in [0, 1]^\ell$, no PPT adversary can distinguish between samples from the Gaussian pancake distribution for a uniformly random $\mathbf{S}$ and the standard Gaussian distribution. In our work, we show a slightly stronger statement that even Gaussian pancake samples where the shift $\mathbf{z} = \mathbf{S}^\top \mathbf{u}$ can depend on the secret are computationally indistinguishable from standard Gaussian samples (Proposition 4.1).

This brings us to the *pancake alignment problem*, which can be viewed as a “semi-search” version of the Gaussian pancake distinguishing problem. The problem asks an adversary with access to Gaussian pancake samples to find a direction $\mathbf{v}$ that is *aligned* with the secret $\mathbf{S}$.

Definition 1.4 (Pancake Alignment Problem, informal version of Definition 4.8). *For an approximation parameter $t \in [1, n]$, the pancake alignment problem asks, given samples from the ℓ -secret Gaussian pancake distribution for a random secret $\mathbf{S} \in \mathbb{R}^{n \times \ell}$, find a unit vector $\mathbf{v} \in \mathbb{R}^n$ such that*

$$\|\mathbf{S}^\top \mathbf{v}\|_2^2 > t \frac{\ell}{n}.$$

Put differently, the pancake alignment problem asks that an adversary cannot achieve a significantly better alignment with the secret subspace than it could without access to Gaussian pancake samples. In particular, for any *fixed* $\mathbf{v}$ independent of $\mathbf{S}$, $\mathbb{E}_{\mathbf{S}}[\|\mathbf{S}^\top \mathbf{v}\|_2^2] = \frac{\ell}{n}$, and standard concentration bounds tell us $\|\mathbf{S}^\top \mathbf{v}\|_2^2 \leq O(\ell/n)$ with high probability. The problem therefore asserts that observing Gaussian pancake samples does not help an adversary improve this alignment by a multiplicative factor of t .

One might wonder why CLWE does not immediately imply the hardness of pancake alignment. If Gaussian pancake samples are computationally indistinguishable from standard Gaussian samples, independent of the secret $\mathbf{S}$, then it is tempting to treat $\mathbf{S}^\top \mathbf{v}$ as if the adversary’s output $\mathbf{v}$ were independent of $\mathbf{S}$. Unfortunately, this is not necessarily the case. Although the distribution of $\mathbf{v}$ is computationally indistinguishable whether the adversary is given Gaussian pancake samples or standard Gaussian samples, this does not imply that the secret-dependent quantity $\mathbf{S}^\top \mathbf{v}$ is computationally indistinguishable between the two experiments. In particular, a distinguisher for $\mathbf{S}^\top \mathbf{v}$ does not necessarily imply a distinguisher for $\mathbf{v}$, since without knowing $\mathbf{S}$, an adversary cannot evaluate the distinguisher for this secret-dependent quantity.

Nevertheless, we show that for certain approximations t , there *is* a reduction from CLWE. Concretely, both when $t = \tilde{\Omega}(\sqrt{\lambda})$, as well as when $t \approx 1/\epsilon$, where ϵ denotes the pancake alignment adversary’s success probability.

Theorem 1.3 (Informal version of Theorems 4.1 and 4.2). *Assume CLWE with binary secrets is hard. Then, for any number of secrets ℓ such that $n - \ell \geq \lambda$, there is some $t = \tilde{\Theta}(\sqrt{\lambda})$ such that the t -pancake alignment problem is hard to solve with inverse polynomial success probability. Moreover, for any $\epsilon \in (0, 1)$, there is some $t = \Theta(1/\epsilon)$ such that t -pancake alignment problem is hard to solve with success probability more than ϵ .*

Despite only having reductions from CLWE for approximations t given above, we conjecture that the problem remains hard to solve with inverse-polynomial success probability even when $t = \Theta(1)$ is a constant (Assumption 4.3), because we currently know of no attacks that outperform the trivial strategy of outputting a fixed vector independent of the Gaussian pancake samples.

New tool: embedding messages into noisy weights. Using the pancake alignment assumption, we develop a method for (robustly) embedding messages into model weights. One can view this as a form of steganography, which has proven to be a powerful tool in the watermarking literature [Zam24; CG24]. The main differences in our setting are that (1) we embed messages into real-valued model weights rather than binary token strings, and (2) we seek robustness with respect to ℓ_2 rather than Hamming perturbations.

Specifically, in Section 5.2, we construct PPT algorithms (Gen, Embed, Dec) that produce weights which embed a decodable message $\mathbf{m}$ whilst appearing computationally indistinguishable from weights that came

from a noisy training algorithm. Given a secret key $\text{sk} \leftarrow \text{Gen}(1^\lambda)$, weights $\mathbf{w} \in \mathbb{R}^n$, and noise $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$, we embed a message $\mathbf{m} \in \{0, 1\}^\lambda$ by computing

$$\mathbf{g}_{\text{emb}} \leftarrow \text{Embed}(\text{sk}, \mathbf{m}, \mathbf{w}, \mathbf{g})$$

and outputting the embedded weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$. We obtain the following guarantees.

Lemma 1.4 (Informal version of the Lemmas in Section 5.2). *There exist PPT algorithms $(\text{Gen}, \text{Embed}, \text{Dec})$ such that, for any message $\mathbf{m} \in \{0, 1\}^\ell$ and base weights $\mathbf{w} \in \mathbb{R}^n$, the following experiment*

$$\text{Sample } \mathbf{g}_{\text{emb}} \leftarrow \text{Embed}(\text{sk}, \mathbf{m}, \mathbf{w}, \mathbf{g}), \quad \text{where } \text{sk} \leftarrow \text{Gen}(1^\lambda) \text{ and } \mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$$

satisfies:

1. **Undetectability.** *Assuming CLWE with binary secrets is hard, the embedded weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$ are computationally indistinguishable from the ordinary noisy weights $\mathbf{w} + \mathbf{g}$ to any user who does not possess sk .*
2. **Decodability.** *The message can be decoded from the embedded weights:*

$$\text{Dec}(\text{sk}, \mathbf{w} + \mathbf{g}_{\text{emb}}) = \mathbf{m}.$$

with overwhelming probability.

3. **Robustness.** *Assuming t -pancake alignment is hard, if a PPT adversary $\mathcal{A}$ chooses a bounded perturbation $\mathbf{v} \leftarrow \mathcal{A}(\mathbf{w} + \mathbf{g}_{\text{emb}})$ with*

$$\|\mathbf{v}\|_2 \leq \tilde{O}\left(\frac{\sigma\sqrt{n}}{\sqrt{\lambda t}}\right),$$

then $\text{Dec}(\text{sk}, \mathbf{w} + \mathbf{g}_{\text{emb}} + \mathbf{v}) = \mathbf{m}$ with overwhelming probability.

4. **Small perturbation.** *The embedding procedure modifies the noise vector by only a small amount:*

$$\|\mathbf{g} - \mathbf{g}_{\text{emb}}\|_2 \leq O(\sigma\sqrt{\lambda}).$$

with overwhelming probability.

While undetectability relies on the hardness of CLWE, robustness relies on our new pancake alignment assumption. Thus, for $t = \tilde{\Omega}(\sqrt{\lambda})$, our entire construction can be based solely on CLWE via the reduction from Theorem 4.1. However, assuming the hardness of pancake alignment for smaller t , such as constant $t = \Theta(1)$, yields a larger robustness threshold.

Given this tool, a model provider can embed a message into weights from a noisy training algorithm by first obtaining a decomposition $(\mathbf{w}, \mathbf{g})$, and then replacing the noise $\mathbf{g}$ with the embedded noise $\mathbf{g}_{\text{emb}}$. The first property ensures that the resulting weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$ look as though they were produced by the original noisy training algorithm. The second and third properties ensure that the decoder can recover the message $\mathbf{m}$ from the resulting weights, even after an adversary has slightly perturbed them. Finally, the fourth property ensures that the embedded weights remain ℓ_2 -close to the original noisy weights: since $\lambda \ll n$, the embedding perturbation $\sigma\sqrt{\lambda}$ is much smaller than the typical ℓ_2 norm $\sigma\sqrt{n}$ of the noise. This property will be useful for obtaining unforgeability (Section 6).

It is worth noting that the decoding algorithm presented in Section 5.2 only *approximately* recovers the message $\mathbf{m}$. For exact recovery, one can instead embed the encoding of the message using an error-correcting code. The decoder can then approximately recover this encoding and use error-correction to recover the original message, as we do in Section 6.

1.2 Related work

Backdooring. A related line of work, initiated by [GKVZ22], studies how to undetectably *backdoor* machine learning models. The goal is to produce a model that, without a backdoor key, is computationally indistinguishable from one obtained through ordinary training, while with the key, one can identify inputs on which the model exhibits special behavior. Follow-up work considers backdoors for which it is hard to generate triggers without the backdoor key [BRV26]. Such hidden behaviors could potentially be used for fingerprinting: a model provider could associate a model with a secret trigger and later verify the fingerprint by observing the model’s response to that trigger. However, existing backdoor constructions do not provide a formal notion of robustness to model modification, and recent work shows that they can, in some settings, be removed [GSVV25].

Fingerprinting. There is a large body of work on heuristic approaches to fingerprinting machine learning models, which broadly fall into two categories. The first introduces a backdoor into the model, typically through additional training [ABCP+18; XWMK+24; PKA25; NHBS+26; RCS26] or by using the model parameters to identify a natural backdoor [ZWHX+24]. The second embeds a detectable signal directly into the model’s parameters [UNSS17; WWZY20]. Although these works refer to themselves as watermarking, we view them as fingerprinting in our terminology, since they certify the model itself. Our approach is closer to the latter. It is unique in the sense that, while prior methods have been shown to detectably alter the model distribution [WK19], our construction achieves undetectability, ruling out such degradation.

Watermarking. The term watermarking has been used in a broad range of applications. Most closely related to our work is the open-source watermarking scheme of [CGMR24], which plants an undetectable signal into model weights by post-processing them with Gaussian noise. Their setting is nevertheless quite different. The watermark is designed to be detected from the model’s *outputs*. While detection is also possible given candidate weights, it requires access to the original, unwatermarked model. In contrast, our fingerprint can be certified directly from candidate weights using only the secret key.

In addition, a line of work studies undetectably watermarking the *outputs* of language models when users have API access to responses [KGWK+23; CGZ24; FGJM+23; CG24; GM24; CHS25; LSS26; MMP26], with similar approaches extending to image models [GZS25]. These approaches modify how outputs are sampled from an *already trained* model. Our setting differs in that we modify the training procedure to implant a signal directly into the model parameters. We refer the reader to [ZGCF+25] for a comprehensive survey on watermarking model outputs.

Another line of work studies cryptographically watermarking *circuits*, where the goal is to produce a circuit that is computationally indistinguishable from an unwatermarked one, yet can be verified as originating from a secret key. Following the impossibility results of [BGIR+12], [CHNV+16] initiated the study of circuit watermarking, and along with subsequent work [KW17; QWZ18; KW19] showed how to watermark circuits implementing pseudorandom functions (PRFs). However, the input-output behavior of a machine learning model is generally far from that of a PRF, so these techniques do not directly apply.

Certifying model ownership. An adjacent line of theoretical work studies certifying model ownership using backdoor-like techniques. [CGZ26] introduce a framework in which a marker takes a model and produces a similar marked model together with a proof of ownership. Given the proof and *black-box* queries to any model with similar input-output behavior to the marked model, one can verify ownership. They show that they can mark a model if and only if it is not *self-correctable*, meaning that its input-output behavior cannot be recovered from an approximation to the model. [GTNP26] study a setting in which ownership is verified directly from the marked model, without a proof. They show that, against time-bounded adversaries, either a method for marking models, a method for making models adversarially robust, or a transferable attack must exist.

Our work is complementary to these settings. First, our verifier has white-box access to the model parameters, which is less general than these frameworks, but allows us to establish qualitatively different guarantees; namely, our guarantees are geometric rather than based directly on input-output behavior. This allows us to capture robustness against physical model transformations such as quantization without assumptions about their effect on behavior, while also capturing robustness against behavioral changes when

appropriate assumptions *do* hold—for example, when large ℓ_2 perturbations to model parameters induce significant change in input-output behavior. Second, we modify the training algorithm itself rather than marking an already-trained model. This provides a well-defined notion of undetectability.

2 Technical Overview

How to embed messages into weights? Recall from Section 1.1, our goal is to take an embedding key $\mathbf{sk}$, a message $\mathbf{m} \in \{0, 1\}^\lambda$, and a decomposition of noisy weights, $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$, and produce embedding noise $\mathbf{g}_{\text{emb}}$. With the secret key, we should be able to recover $\mathbf{m}$ from the embedded weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$. At the same time, without the secret key, $\mathbf{g}_{\text{emb}}$ should remain computationally indistinguishable from $\mathbf{g}$, thereby preserving undetectability.

Shifted Gaussian pancake samples, where the shift *depends on the secret-key*, give us exactly this. Recall the λ -secret Gaussian pancake distribution $\text{GP}_{\beta, \gamma}(\mathbf{S}, \mathbf{u}, \mathbf{z})$, where $\mathbf{S} \in \mathbb{R}^{n \times \lambda}$ consists of λ orthonormal secrets, and $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{z} \in [0, 1]^\lambda$ are inputs that describe the shift. It generates samples as

$$\mathbf{y} \leftarrow D_{\mathbb{R}^n, 1} \quad \text{conditioned on} \quad \gamma \mathbf{S}^\top \mathbf{y} + \mathbf{e} = \mathbf{S}^\top \mathbf{u} + \mathbf{z} \pmod{1}, \quad \text{where } \mathbf{e} \leftarrow D_{\mathbb{R}^\lambda, \beta}.$$

Proposition 4.1 states that, assuming CLWE, samples from $\text{GP}(\mathbf{S}, \cdot, \cdot)$ for a random $\mathbf{S}$ (where the shifts can be chosen adaptively) are computationally indistinguishable from standard Gaussian samples. This is analogous to standard learning with errors (LWE), where LWE samples remain indistinguishable from null samples even when they contain a secret-dependent linear shift.

This suggests a natural first approach: we use $\mathbf{z}$ to embed the message and $\mathbf{u}$ to cancel the contribution from $\mathbf{w}$. Specifically, the embedding key $\mathbf{S} \in \mathbb{R}^{n \times \lambda}$ is a Gaussian pancake secret, and the embedding algorithm $\text{Embed}(\mathbf{S}, \mathbf{m}, \mathbf{w}, \mathbf{g})$ ignores $\mathbf{g}$ and outputs $\mathbf{g}_{\text{emb}} = \sigma \mathbf{y}$, where

$$\mathbf{y} \leftarrow \text{GP}_{\beta, \gamma}(\mathbf{S}, -\frac{\gamma}{\sigma} \mathbf{w}, \mathbf{m}), \quad \text{with } \mathbf{m} \text{ represented in } \{1/4, 3/4\}^\ell.$$

This preserves undetectability, since $\sigma \mathbf{y}$ is computationally indistinguishable from a sample from $D_{\mathbb{R}^n, \sigma}$. Furthermore, we can define a decoder $\text{Dec}(\mathbf{S}, \mathbf{w} + \sigma \mathbf{y})$ that approximately recovers $\mathbf{m}$ by projecting the weights onto the secret directions and scaling appropriately. A simple calculation gives

$$\begin{aligned} \frac{\gamma}{\sigma} \mathbf{S}^\top (\mathbf{w} + \sigma \mathbf{y}) &= \frac{\gamma}{\sigma} \mathbf{S}^\top \mathbf{w} + \gamma \mathbf{S}^\top \mathbf{y} \pmod{1} \\ &= \frac{\gamma}{\sigma} \mathbf{S}^\top \mathbf{w} - \frac{\gamma}{\sigma} \mathbf{S}^\top \mathbf{w} + \mathbf{m} - \mathbf{e} \pmod{1} \\ &= \mathbf{m} - \mathbf{e} \pmod{1} \end{aligned}$$

where $\mathbf{e} \leftarrow D_{\mathbb{R}^\lambda, \beta}$ is small noise. For exact recovery, when the noise level β is sufficiently small, one can round each coordinate of $\mathbf{m} - \mathbf{e}$ to whichever of $1/4$ or $3/4$ is closer.

This construction satisfies undetectability and decodability, but it simply discards the input noise $\mathbf{g}$ and samples a fresh embedding vector $\mathbf{g}_{\text{emb}}$. This incurs an ℓ_2 -distance $\|\mathbf{g} - \mathbf{g}_{\text{emb}}\|_2$ of roughly $2\sigma\sqrt{n}$, rather than the $\sigma\sqrt{\lambda}$ bound claimed in Section 1.1. To achieve the latter, we instead start from Gaussian $\mathbf{g}$ and *round* it to the desired Gaussian pancake vector $\mathbf{g}_{\text{emb}}$, using the randomized rounding procedure of [Pei10]. This has the additional benefit that $\|\mathbf{g} - \mathbf{g}_{\text{emb}}\|_2 \leq O(\sigma\sqrt{\lambda})$. Figure 1 provides a visual representation of the embedding.

Robustness from pancake alignment. To see robustness, suppose an adversary observes an embedded sample $\mathbf{w} + \mathbf{g}_{\text{emb}}$ and attempts to break decoding by adding a bounded perturbation $\mathbf{v} \in \mathbb{R}^n$. The decoder $\text{Dec}(\mathbf{S}, \mathbf{w} + \mathbf{g}_{\text{emb}} + \mathbf{v})$ then recovers

$$\mathbf{m} - \mathbf{e} + \frac{\gamma}{\sigma} \mathbf{S}^\top \mathbf{v} \pmod{1}.$$

Therefore, to cause the decoder to fail, the adversary must induce a sufficiently large perturbation in the secret subspace; in particular, $\mathbf{v}$ must satisfy $\|\frac{\gamma}{\sigma} \mathbf{S}^\top \mathbf{v}\|_2^2 > \alpha \lambda$ for some constant $\alpha \in (0, 1/2)$ ⁸. Equivalently,

⁸The adversary needs to induce a sufficiently large ℓ_2 contribution (rather than an ℓ_∞ one), since it must prevent the decoder from recovering *most* coordinates of $\mathbf{m}$, which suffices for exact recovery when $\mathbf{m}$ is encoded using an error-correcting code.

writing $\mathbf{v}^* = \mathbf{v}/\|\mathbf{v}\|_2$, the adversary must find a direction $\mathbf{v}^*$ that is sufficiently aligned with $\mathbf{S}$:

$$\|\mathbf{S}^\top \mathbf{v}^*\|_2^2 > \frac{\sigma^2 \alpha}{\gamma^2 \|\mathbf{v}\|_2^2} \lambda.$$

If $\mathbf{v}$ were chosen independently of $\mathbf{S}$, such alignment would be extremely unlikely when $n \gg \lambda$, since a high-dimensional vector is almost always nearly orthogonal to a random low dimensional subspace. The problem is that $\mathbf{v}$ is not necessarily independent of $\mathbf{S}$, since the adversary sees the Gaussian pancake sample $\mathbf{g}_{\text{emb}}$. Therefore, to argue that such an attack can't occur, we must assume that the pancake alignment problem is hard for approximation $t = n \cdot \frac{\sigma^2 \alpha}{\gamma^2 \|\mathbf{v}\|_2^2}$. Under this assumption, $\mathcal{A}$ cannot remove the fingerprint so long as the perturbation is bounded by $\|\mathbf{v}\|_2 \leq \frac{\sigma \sqrt{\alpha n}}{\gamma \sqrt{t}}$. Using our reduction from CLWE (Theorem 4.1), we can set $t = \tilde{\Theta}(\sqrt{\lambda})$ and $\gamma = \tilde{\Theta}(\sqrt{\lambda})$ to see that the message can still be decoded after $\tilde{\Theta}(\frac{\sigma \sqrt{n}}{\lambda^{3/4}})$ perturbation.

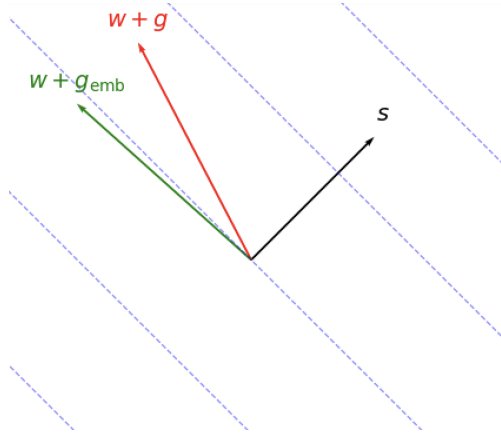


Figure 1: Visualization of embedding a single-bit message $m = 0$ in dimension $n = 2$. The red arrow represents trained weights $\mathbf{w} + \mathbf{g}$ produced by a noisy training algorithm. These weights are then *rounded* in the secret direction to $\mathbf{w} + \mathbf{g}_{\text{emb}}$, which lies within a discrete “pancake”. In higher dimensions, decoding is robust because an adversary cannot easily find a perturbation in the secret direction, i.e., a perturbation that moves the green arrow away from the pancake, back towards the red arrow.

Robust fingerprinting (Section 5). Given a way to embed messages into noisy weights, we can construct a robust fingerprinting scheme for any noisy training algorithm Train . The secret key consists of a Gaussian pancake secret $\mathbf{S}$ and a random message $\mathbf{m} \leftarrow \{0, 1\}^\lambda$. The fingerprinting algorithm computes noisy weights $(\mathbf{w}, \mathbf{g})$, embeds $\mathbf{m}$ into them, and outputs the embedded weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$. The verification algorithm $\text{Ver}((\mathbf{S}, \mathbf{m}), \mathbf{w}')$ decodes a message from $\mathbf{w}'$ and accepts iff it equals $\mathbf{m}$. Undetectability, correctness, and robustness follow directly from the corresponding properties of the embedding scheme. Soundness follows because, for fixed weights $\mathbf{w}'$ independent of the random message $\mathbf{m}$, the decoded value $\frac{\gamma}{\sigma} \mathbf{S}^\top \mathbf{w}' \pmod{1}$ is unlikely to be close to $\mathbf{m}$.

Unforgeable fingerprinting (Section 6). To additionally achieve unforgeability, we need to ensure that the verification procedure is not overly robust. To do so, rather than embedding the same random message $\mathbf{m}$ in every fingerprinted model, we instead embed a “robust” MAC of the noisy weights $\mathbf{w} + \mathbf{g}$ into the final output weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$. The verifier then decodes this MAC, and uses it to ensure that they only accept weights that are *close* to the original noisy weights.

To achieve this, we adapt the non-malleable transformation underlying the construction of ideal pseudo-random codes (PRCs) [AACD+25]. Their work gives a generic transformation that uses a PRF to convert a robust message-embedding scheme into an *authenticated* one, which prevents the decoder from decoding a message that was never embedded. That is, if an adversary $\mathcal{A}$ observes embeddings and outputs weights $\mathbf{w}'$, then we can guarantee:

1. *Robustness*: If $\mathbf{w}'$ is close to one of the embedded weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$ observed by the adversary, then the decoder recovers the corresponding embedded message.
2. *Authentication*: Otherwise, the decoder either outputs $\perp$ or recovers a message that was embedded into one of the previous weights observed by the adversary.

Given an authenticated embedding scheme, one idea for training fingerprinted weights $\text{FP}^D(\text{sk}, \text{spec})$ is to first produce noisy weights $(\mathbf{w}, \mathbf{g})$, and then embed the fingerprinting algorithm's randomness along with the training distribution D and training specification spec into the fingerprinted weights $\mathbf{w} + \mathbf{g}_{\text{emb}}$. During verification, $\text{Ver}(\text{sk}, \mathbf{w}')$ decodes (randomness, D , spec) from $\mathbf{w}'$ and uses it to reconstruct $\mathbf{w} + \mathbf{g}_{\text{emb}}$. The verifier can then check if $\mathbf{w}'$ is close to the reconstructed weights. If this were possible, we would obtain the following guarantees:

1. *Robustness*: If $\mathbf{w}'$ is close to some fingerprinted weights, the decoder recovers the information needed to reconstruct those weights, and the verifier accepts.
2. *Unforgeability*: Otherwise, if $\mathbf{w}'$ is far from all fingerprinted weights, authentication guarantees that the decoder either outputs $\perp$ or reconstructs some previously fingerprinted weights. In the latter case, those weights are far from $\mathbf{w}'$, so the verifier rejects.

However, there is an issue. This construction only works when the training distribution has a short description, so that it can be embedded into the n -dimensional vector $\mathbf{g}_{\text{emb}}$. Unfortunately, this does not capture many practical applications, such as when the training distribution is described by a dataset containing many datapoints $\mathbf{x} \in \mathbb{R}^n$. To overcome this, we still leverage authenticated embeddings, but instead of embedding all the information needed to *recover* $\mathbf{w} + \mathbf{g}_{\text{emb}}$, we only embed enough information for the verifier to *evaluate the closeness* between $\mathbf{w}'$ and $\mathbf{w} + \mathbf{g}_{\text{emb}}$. Specifically, we embed a random seed s and a seeded hash $h_s(\mathbf{w} + \mathbf{g})$, where h is an ℓ_2 locality-sensitive hash, which allows one to determine whether two inputs are close in ℓ_2 distance given their hashes, provided that the inputs are chosen by an adversary with *no knowledge of the seed* s ⁹. Since the seed and hash value form a short digest, they can be embedded into the n -dimensional embedding noise.

During verification, the verifier uses the authenticated decoder to recover either $\perp$ or a pair $(s, h_s(\mathbf{w} + \mathbf{g}))$. It can then use the seed s to compute $h_s(\mathbf{w}')$ and compare it with $h_s(\mathbf{w} + \mathbf{g})$ to determine whether $\mathbf{w}'$ is close to the weights $\mathbf{w} + \mathbf{g}$ before embedding. Since $\|\mathbf{g} - \mathbf{g}_{\text{emb}}\|_2 \leq O(\sigma\sqrt{\lambda})$, this essentially reduces to checking whether $\mathbf{w}'$ is close to $\mathbf{w} + \mathbf{g}_{\text{emb}}$.

One caveat is that if an adversary is able to learn the hash seed s by observing the fingerprinted weights, they may be able to construct some $\mathbf{w}'$ for which the closeness test between $h_s(\mathbf{w}')$ and $h_s(\mathbf{w} + \mathbf{g})$ gives an incorrect result. To prevent this, one could either use an adaptively robust locality-sensitive hash for ℓ_2 [BRVV26], or, in our setting where the verification key is secret, simply embed an *encryption* of $(s, h_s(\mathbf{w} + \mathbf{g}))$ instead.

Reducing CLWE to pancake alignment (Section 7). To show that our decoder is robust under the CLWE assumption, we give a reduction from CLWE to pancake alignment when $t = \tilde{\Omega}(\sqrt{\lambda})$. We work in λ dimensions. To start, in Section 8 we show that the hardness of CLWE implies the hardness of the (single-secret) shifted Gaussian pancake problem: namely, for a uniformly random unit vector $\mathbf{s} \leftarrow \mathbb{S}^{\lambda-1}$, the oracle $\text{GP}(\mathbf{s}, \cdot, \cdot)$ is computationally indistinguishable from an oracle that ignores its inputs and instead outputs standard Gaussian vectors $\mathbf{g} \leftarrow D_{\mathbb{R}^\lambda, 1}$.

Now, suppose there is an adversary $\mathcal{A}$ that breaks the t -pancake alignment problem for a single secret. With probability $1/\text{poly}(\lambda)$ over $\mathbf{s} \leftarrow \mathbb{S}^{\lambda-1}$ and $\mathbf{v} \leftarrow \mathcal{A}^{\text{GP}(\mathbf{s}, \cdot, \cdot)}(1^\lambda)$, we have

$$(\mathbf{s}^\top \mathbf{v})^2 > t/\lambda.$$

We will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that distinguishes between Gaussian pancake and standard Gaussian oracles. The key observation is that rotating Gaussian pancake samples preserves the distribution up to a corresponding rotation of the secret: if $\mathbf{y} \leftarrow \text{GP}(\mathbf{s}, \mathbf{R}^\top \mathbf{u}, \mathbf{z})$, then $\mathbf{R}\mathbf{y}$ is distributed as a sample from

⁹Such hashes can be constructed unconditionally, for example, by randomly projecting the inputs to a lower-dimensional space [JL+84].

$\text{GP}(\mathbf{R}\mathbf{s}, \mathbf{u}, \mathbf{z})$. The adversary $\mathcal{B}^{O(\cdot, \cdot)}$ exploits this by sampling two independent uniformly random rotations $\mathbf{R}_1, \mathbf{R}_2$ and running $\mathcal{A}$ twice, each time using its own oracle rotated with $\mathbf{R}_i$. This produces vectors $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}$, which $\mathcal{B}$ then rotates back, setting $\mathbf{u}^{(i)} = \mathbf{R}_i^\top \mathbf{v}^{(i)}$.

In the case where $O(\cdot, \cdot)$ outputs standard Gaussian vectors, the rotated oracles are both distributed exactly as standard Gaussian oracles, by rotational invariance of the Gaussian distribution. Thus, the vectors $\mathbf{v}^{(i)}$ are independent of the rotations $\mathbf{R}_i$. Rotating them back by the uniformly random rotations therefore makes the resulting vectors uniformly random, which means they are approximately orthogonal:

$$\left| (\mathbf{u}^{(1)})^\top \mathbf{u}^{(2)} \right| \approx \frac{1}{\sqrt{\lambda}}.$$

On the other hand, when $O(\cdot, \cdot)$ outputs Gaussian pancake samples, $\mathcal{B}$ runs $\mathcal{A}$ on two independent and uniform secrets $\mathbf{R}_1\mathbf{s}$ and $\mathbf{R}_2\mathbf{s}$. If $\mathcal{A}$ succeeds on both, which occurs with probability at least $1/(\text{poly}(\lambda))^2$, then each $\mathbf{v}^{(i)}$ is (t/λ) -aligned with $\mathbf{R}_i^\top \mathbf{s}$. Consequently, when rotated back, each $\mathbf{u}^{(i)}$ is (t/λ) -aligned with the same $\mathbf{s}$. We can then show that this implies the $\mathbf{u}^{(i)}$ themselves are aligned,

$$\left| (\mathbf{u}^{(1)})^\top \mathbf{u}^{(2)} \right| \geq \tilde{\Omega}\left(\frac{t}{\lambda}\right).$$

When $t = \tilde{\Omega}(\sqrt{\lambda})$, the alignment is noticeably larger than in the standard Gaussian case, allowing $\mathcal{B}$ to distinguish the two oracles. Figure 2 illustrates the two cases. The key technical step is showing that the alignment between $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$ is dominated by their alignment in the secret direction $\mathbf{s}$, and that it cannot be canceled by a large negative contribution from their alignment in the subspace $\mathbf{s}^\perp$ orthogonal to $\mathbf{s}$. Specifically, we show that the components of $\mathbf{u}^{(i)}$ in $\mathbf{s}^\perp$ are uniform, so their inner product is likely small.

The final steps are to extend the reduction from dimension λ to any dimension $n > \lambda$, while preserving essentially the same approximation t , and then to extend from a single secret to up to $n - \lambda$ secrets, again with essentially the same t .

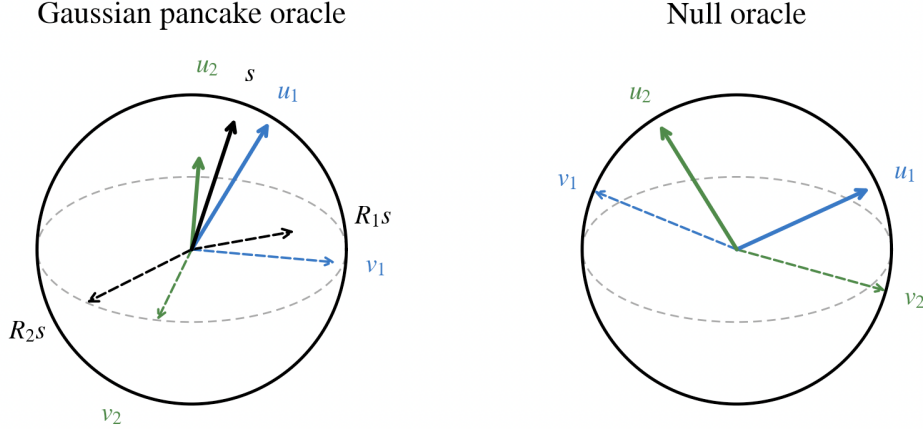


Figure 2: Visualization of our reduction in $\lambda = 3$ dimensions. Given a Gaussian pancake oracle (left), the adversary's outputs $\mathbf{v}^{(i)}$ are each aligned with a rotated secret $\mathbf{R}_i\mathbf{s}$, so when rotated back the $\mathbf{u}^{(i)}$'s are aligned. In contrast, with a null oracle (right), the rotated $\mathbf{u}^{(i)}$'s are uniform over the sphere and nearly orthogonal.

3 Preliminaries

Notation We write $D_{\mathbb{R}^n, \sigma}$ for the Gaussian distribution with mean $\mathbf{0}$ and covariance $\frac{\sigma^2}{2\pi} \mathbf{I}_n$. At times, we refer to the standard Gaussian, by which we mean $D_{\mathbb{R}^n, 1}$ despite the covariance being $\frac{1}{2\pi} \mathbf{I}_n$. For a

distribution $\mathcal{D}$, we write $s \leftarrow \mathcal{D}$ to denote sampling s from $\mathcal{D}$. For a set S , we write $U(S)$ for the uniform distribution over S , and use $s \leftarrow S$ as shorthand for $s \leftarrow U(S)$. For random variables X and Y , we write $X \stackrel{d}{=} Y$ to denote that X and Y are identically distributed.

For a matrix $\mathbf{S} \in \mathbb{R}^{n \times m}$ and $i \in [m]$, we write $\mathbf{S}[i] \in \mathbb{R}^n$ for the i th column of $\mathbf{S}$. For a vector $\mathbf{s} \in \mathbb{R}^n$ and index $j \in [n]$, we write $s_j \in \mathbb{R}$ for the j th entry of $\mathbf{s}$. For a set $I \subseteq [n]$, we write $\mathbf{s}_I$ for the vector consisting of the coordinates of $\mathbf{s}$ indexed by I , and $\mathbf{s}_{-I}$ for the vector consisting of the remaining coordinates, indexed by $[n] \setminus I$.

We write $\mathbb{S}^{n-1}$ for the $(n-1)$ -dimensional unit sphere, consisting of the set of unit vectors in $\mathbb{R}^n$. We write $\mathbb{Z}$ for the set of integers. For $\mathbf{a} \in \mathbb{R}^n$, we write $\mathbf{a} \bmod 1$ for the centered modulo operation, defined as the unique $\mathbf{z} \in [-1/2, 1/2)^n$ such that $\mathbf{a} - \mathbf{z} \in \mathbb{Z}^n$. We write $\mathbf{a} \bmod 1$ for the standard modulo operation, defined as the unique $\mathbf{z} \in [0, 1)^n$ such that $\mathbf{a} - \mathbf{z} \in \mathbb{Z}^n$. For $a, \eta \in \mathbb{R}$, we write $\lfloor a \rfloor_\eta$ for the largest element of $\eta\mathbb{Z}$ that is at most a . For vectors, we extend this notation entry-wise.

For $\ell, n \in \mathbb{N}$, let $V_{\ell, n} \subseteq \mathbb{R}^{n \times \ell}$ denote the Stiefel manifold of $n \times \ell$ matrices with orthonormal columns:

$$V_{\ell, n} = \{ \mathbf{W} \in \mathbb{R}^{n \times \ell} : \mathbf{W}^\top \mathbf{W} = \mathbf{I}_\ell \}.$$

For $n \in \mathbb{N}$, let $\text{SO}(n)$ denote the special orthogonal group, which consists of n -dimensional rotations about the origin. For $i \in [n]$, let $\mathbf{b}^{(i)} \in \mathbb{R}^n$ denote the i th standard basis vector, i.e.,

$$b_j^{(i)} = \begin{cases} 1, & \text{if } j = i, \\ 0, & \text{otherwise.} \end{cases}$$

We write $n = n(\lambda)$ to denote that $n : \mathbb{N} \rightarrow \mathbb{R}$ is a function of the security parameter. When the dependence on λ is clear from context, we overload notation and write n to refer to $n(\lambda)$. We write $n = \text{poly}(\lambda)$ when n is polynomially bounded and $\epsilon = \text{negl}(\lambda)$ when ϵ is bounded above by a negligible function.

3.1 Lattices and discrete Gaussians

Lattices A lattice is a discrete additive subgroup of $\mathbb{R}^n$. We assume throughout that all lattices are full-rank. The dual of a lattice L is defined as

$$L^* := \{ \mathbf{y} \in \mathbb{R}^n : \mathbf{x}^\top \mathbf{y} \in \mathbb{Z} \text{ for all } \mathbf{x} \in L \}.$$

For an n -dimensional lattice L and function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, we write $f(L) := \sum_{\mathbf{y} \in L} f(\mathbf{y})$. For $\mathbf{y} \in \mathbb{R}^n$, the shifted lattice (or coset) $L + \mathbf{y}$ is defined as $L + \mathbf{y} := \{ \mathbf{x} + \mathbf{y} : \mathbf{x} \in L \}$.

Discrete Gaussian For $s > 0$, we define the Gaussian function by $\rho_s(\mathbf{x}) = \exp(-\pi \|\mathbf{x}/s\|_2^2)$. For a shifted lattice $\mathbf{y} + L$, the *discrete* Gaussian $D_{\mathbf{y}+L, s}$ is the distribution supported on $\mathbf{y} + L$ that assigns each $\mathbf{x} \in \mathbf{y} + L$ probability proportional to a Gaussian with standard deviation $s/\sqrt{2\pi}$,

$$\Pr[\mathbf{x}] = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbf{y} + L)}.$$

Smoothing parameter We recall the smoothing parameter, which can be viewed intuitively as the standard deviation above which a discrete Gaussian behaves similarly to a continuous Gaussian.

Definition 3.1 (Smoothing parameter [MR07]). *For a lattice L and a parameter $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(L)$ is defined as*

$$\eta_\epsilon(L) := \inf \{ s : \rho_{1/s}(L^* \setminus \{ \mathbf{0} \}) \leq \epsilon \}.$$

Prior work provides various upper bounds on the smoothing parameter. Here, we only require bounds for the integer lattice.

Lemma 3.1 ([PRS17], Special case of Lemma 2.6). *For any $n \in \mathbb{N}$ and $c \geq 1$, $\eta_\epsilon(\mathbb{Z}^n) \leq c\sqrt{n}$, where $\epsilon = \exp(-c^2 n)$.*

Lemma 3.2 ([MR07], Special case of Lemma 3.3). *For any $n \in \mathbb{N}$ and $\epsilon > 0$,*

$$\eta_\epsilon(\mathbb{Z}^n) \leq \sqrt{\frac{\ln(2n(1 + 1/\epsilon))}{\pi}}.$$

Facts about various distributions. We recall some facts about the discrete Gaussian, Gaussian, and uniform distributions on the sphere. First, we see that a wide enough discrete Gaussian distribution over a shifted lattice has similar tails to a Gaussian distribution.

Lemma 3.3. *For any n -dimensional lattice L , shift $\mathbf{y} \in \mathbb{R}^n$, $\epsilon \in (0, 1)$, and $s > \eta_\epsilon(L)$, we have*

$$\Pr_{\mathbf{x} \leftarrow D_{L+\mathbf{y}, s}} [\|\mathbf{x}\|_2 > s\sqrt{n}] \leq \left(\frac{1+\epsilon}{1-\epsilon}\right) 4^{-n}.$$

Proof. From [DD18, Corollary 4]

$$\Pr_{\mathbf{x} \leftarrow D_{L+\mathbf{y}, s}} [\|\mathbf{x}\|_2 > s\sqrt{n}] \leq \frac{\rho_s(L)}{\rho_s(L+\mathbf{y})} 4^{-n}$$

From [Pei10, Lemma 2.4]

$$\frac{\rho_s(L)}{\rho_s(L+\mathbf{y})} \leq \frac{1+\epsilon}{1-\epsilon}$$

when $s \geq \eta_\epsilon(L)$. □

This is comparable to the following tail bound on the continuous Gaussian distribution.

Lemma 3.4 ([LM00], χ^2 tail bounds). *For any $n \in \mathbb{N}$, $\sigma > 0$, $t > 0$,*

$$\Pr_{\mathbf{e} \leftarrow D_{\mathbb{R}^n, \sigma}} [\|\mathbf{e}\|_2^2 \geq \frac{\sigma}{2\pi}(n + 2\sqrt{nt} + 2t)] \leq \exp(-t)$$

and

$$\Pr_{\mathbf{e} \leftarrow D_{\mathbb{R}^n, \sigma}} [\|\mathbf{e}\|_2^2 \leq \frac{\sigma}{2\pi}(n - 2\sqrt{nt})] \leq \exp(-t).$$

It is also comparable to the uniform distribution over the unit sphere, whose tails behave like a Gaussian with standard deviation $1/\sqrt{n}$.

Lemma 3.5 ([Ver18, Theorem 3.4.5]). *For any $n \in \mathbb{N}$, $\mathbf{v} \in \mathbb{S}^{n-1}$, and $t \geq 0$,*

$$\Pr_{\mathbf{s} \leftarrow \mathbb{S}^{n-1}} [\mathbf{s}^\top \mathbf{v} \geq t] \leq 2 \exp(-t^2 n/2).$$

Another useful fact will be that a wide enough continuous Gaussian is statistically close to a mixture of discrete Gaussians with equal weight.

Lemma 3.6 ([GVV22], in the proof of Lemma 17). *Let $t > 0$ be a parameter and let $\tau \geq \sqrt{\ln t + \omega(\log \lambda)}$ be the standard deviation. Then the distribution of y sampled as*

$$y \leftarrow D_{\mathbb{Z}+a, \tau}, \quad \text{where } a \leftarrow [0, 1)$$

is $\text{negl}(\lambda)/t$ close (in TV distance) to $D_{\mathbb{R}, \tau}$.

In addition, it will be useful to remember that the non-integral part of a wide enough Gaussian is statistically close to uniform.

Lemma 3.7 ([MR07], Corollary of Lemmas 3.3 and 4.1). *Let $t > 0$ be a parameter, let $\tau \geq \sqrt{\ln t + \omega(\log \lambda)}$ be the standard deviation, and let $a \leftarrow D_{\mathbb{R}, \tau}$. The distribution of $(a \bmod 1)$ is $\text{negl}(\lambda)/t$ close to uniform over $[0, 1)$.*

3.2 Cryptography

Pseudorandom functions A pseudorandom function (PRF) [GGM86] is a keyed function that is computationally indistinguishable from a uniformly random function to adversaries with black-box access.

Definition 3.2 (Pseudorandom function). *Let $\ell_{\text{in}} = \text{poly}(\lambda)$ and $\ell_{\text{out}} = \text{poly}(\lambda)$ denote the input and output lengths, respectively. A pseudorandom function family is a family of efficiently computable functions $\{\text{PRF}_k : \{0, 1\}^{\ell_{\text{in}}(\lambda)} \rightarrow \{0, 1\}^{\ell_{\text{out}}(\lambda)}\}_{k \in \{0, 1\}^\lambda}$ such that for every PPT adversary $\mathcal{A}$,*

$$\left| \Pr_{k \leftarrow \{0, 1\}^\lambda} [\mathcal{A}^{\text{PRF}_k(\cdot)}(1^\lambda) = 1] - \Pr [\mathcal{A}^{f(\cdot)}(1^\lambda) = 1] \right| \leq \text{negl}(\lambda),$$

where f is sampled uniformly at random from the set of all functions with the same domain and range as PRF_k .

Symmetric-key encryption We recall symmetric-key encryption with indistinguishability under chosen-plaintext attacks (IND-CPA) security. Informally, IND-CPA security guarantees that encryptions of any two adversarially chosen messages are computationally indistinguishable without knowledge of the secret key.

Definition 3.3 (Secret-key encryption). *Let $\ell_{\text{in}} = \text{poly}(\lambda)$ and $\ell_{\text{out}} = \text{poly}(\lambda)$ with $\ell_{\text{in}} \leq \ell_{\text{out}}$ denote the message and ciphertext lengths, respectively, and let $s = \text{poly}(\lambda)$ denote the number of random bits needed for each encryption. A (fixed-length) secret-key encryption scheme is a tuple of PPT algorithms $(\text{Gen}, \text{Enc}, \text{Dec})$ where*

- $\text{Gen}(1^\lambda)$ takes in a security parameter and outputs a secret key sk .
- $\text{Enc}(\text{sk}, \mathbf{m}; \mathbf{r})$ takes as input a message $\mathbf{m} \in \{0, 1\}^{\ell_{\text{in}}(\lambda)}$ and outputs a ciphertext $\mathbf{ct} \in \{0, 1\}^{\ell_{\text{out}}(\lambda)}$ using randomness $\mathbf{r} \leftarrow \{0, 1\}^{s(\lambda)}$.
- $\text{Dec}(\text{sk}, \mathbf{ct})$ takes as input a ciphertext $\mathbf{ct} \in \{0, 1\}^{\ell_{\text{out}}(\lambda)}$ and outputs a message $\mathbf{m} \in \{0, 1\}^{\ell_{\text{in}}(\lambda)}$ or failure $\perp$.

Additionally, these algorithms satisfy:

Perfect correctness.¹⁰ For every key $\text{sk} \in \text{supp}(\text{Gen}(1^\lambda))$ and message $\mathbf{m} \in \{0, 1\}^{\ell_{\text{in}}(\lambda)}$,

$$\Pr_{\mathbf{r} \leftarrow \{0, 1\}^{s(\lambda)}} [\text{Dec}(\text{sk}, \text{Enc}(\text{sk}, \mathbf{m}; \mathbf{r})) = \mathbf{m}] = 1.$$

IND-CPA security. For a PPT adversary $\mathcal{A}$ and a bit $b \in \{0, 1\}$, define the experiment $\text{Expt}_b(\lambda, \mathcal{A})$ as follows. A challenger samples $\text{sk} \leftarrow \text{Gen}(1^\lambda)$ and gives $\mathcal{A}$ access to an encryption oracle $\text{Enc}_{\text{sk}}(\cdot)$. The adversary submits two messages $(\mathbf{m}^{(0)}, \mathbf{m}^{(1)})$, each of length $\ell_{\text{in}}(\lambda)$. The challenger returns a challenge ciphertext $\mathbf{ct}^* = \text{Enc}(\text{sk}, \mathbf{m}^{(b)}; \mathbf{r})$, where $\mathbf{r} \leftarrow \{0, 1\}^{s(\lambda)}$ is fresh randomness. The adversary then outputs a bit b' , and the experiment outputs 1 if and only if $b' = b$. We require that, for every PPT adversary $\mathcal{A}$,

$$|\Pr[\text{Expt}_0(\lambda, \mathcal{A}) = 1] - \Pr[\text{Expt}_1(\lambda, \mathcal{A}) = 1]| \leq \text{negl}(\lambda).$$

3.3 Continuous Learning with Errors

We define the *advantage* $\delta = \delta(\lambda)$ of an adversary $\mathcal{A}$ solving an *average-case* distinguishing problem between a secret-dependent oracle $\mathcal{O}_{\lambda, \mathbf{s}}$, parameterized by λ and $\mathbf{s}$ drawn from a distribution W_λ , and a null oracle $\mathcal{O}'_\lambda$ as

$$\delta := \left| \Pr_{\mathbf{s} \leftarrow W_\lambda} [\mathcal{A}^{\mathcal{O}_{\lambda, \mathbf{s}}}(1^\lambda) = 1] - \Pr [\mathcal{A}^{\mathcal{O}'_\lambda}(1^\lambda) = 1] \right|$$

We say that a distinguishing task is *hard* if, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon = \text{negl}(\lambda)$ such that $\mathcal{A}$'s advantage on the task is at most ϵ .

The distinguishing tasks we will consider are the decisional versions of both discrete learning with errors (LWE) and continuous learning with errors (CLWE).

¹⁰We only require perfect correctness for ease of presentation; our arguments later on can be adapted to schemes with negligible correctness error.

Definition 3.4 (Learning with Errors Distribution [Reg09]). Let $n, q \in \mathbb{N}$ and $\sigma \in \mathbb{R}_{>0}$. For a secret $\mathbf{s} \in \mathbb{Z}_q^n$, we define the LWE distribution $\text{LWE}_{\mathbf{s}}(n, \sigma, q)$ as the distribution that generates samples

$$(\mathbf{a}, \mathbf{s}^\top \mathbf{a} + e \bmod q),$$

where $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and $e \leftarrow D_{\mathbb{Z}, \sigma}$.

Definition 3.5 (Decisional LWE). Let $n = n(\lambda)$, $q = q(\lambda)$, $\sigma = \sigma(\lambda)$ be parameter families, and let $W = \{W_\lambda\}$ be a family of distributions over $\mathbb{Z}_q^n$. The LWE problem $\text{LWE}(n, \sigma, q, W)$ asks to distinguish between an LWE oracle $\text{LWE}_{\mathbf{s}}$ for a random $\mathbf{s} \leftarrow W_\lambda$, where each query returns an independent sample from $\text{LWE}_{\mathbf{s}}(n, \sigma, q)$, and a null oracle Null , where each query returns an independent null sample, $(\mathbf{y}, r)$ for $\mathbf{y} \leftarrow \mathbb{Z}_q^n$, $r \leftarrow \mathbb{Z}_q$.

Definition 3.6 (Continuous Learning with Errors Distribution [BRST21]). Let $n \in \mathbb{N}$ and $\gamma, \beta \in \mathbb{R}_{>0}$. For a secret $\mathbf{s} \in \mathbb{R}^n$, we define the continuous LWE distribution $\text{CLWE}_{\mathbf{s}}(n, \beta, \gamma)$ as the distribution that generates samples

$$(\mathbf{y}, \gamma \mathbf{s}^\top \mathbf{y} + e \bmod 1),$$

where $\mathbf{y} \leftarrow D_{\mathbb{R}^n, 1}$ and $e \leftarrow D_{\mathbb{R}, \beta}$.

Definition 3.7 (Decisional CLWE). Let $n = n(\lambda)$, $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$ be parameter families, and let $W = \{W_\lambda\}$ be a family of distributions over $\mathbb{S}^{n-1}$. The CLWE problem $\text{CLWE}(n, \beta, \gamma, W)$ asks to distinguish between a CLWE oracle $\text{CLWE}_{\mathbf{s}}$ for a random $\mathbf{s} \leftarrow W_\lambda$, where each query returns an independent sample from $\text{CLWE}_{\mathbf{s}}(n, \beta, \gamma)$, and a null oracle Null , where each query returns an independent null sample, $(\mathbf{y}, r)$ for $\mathbf{y} \leftarrow D_{\mathbb{R}^n, 1}$, $r \leftarrow [0, 1)$.

Prior work [BRST21] has shown that when the secret distribution W_λ is uniform over the unit sphere $\mathbb{S}^{\lambda-1}$, CLWE is hard for parameters $\gamma \geq 2\sqrt{\lambda}$ and $\beta = 1/\text{poly}(\lambda)$, under the assumption that worst-case lattice problems are hard for efficient quantum algorithms. This is analogous to the classical LWE setting, where worst-case lattice problems admit quantum reductions to LWE for appropriate polynomial modulus-to-noise ratios [Reg09]. More recently, [GVV22] established a direct classical reduction from LWE with binary secrets to CLWE with normalized binary secrets.

Theorem 3.8 ([GVV22], Special case of Theorem 6). Let $\lambda \in \mathbb{N}$ be the security parameter (and dimension). Assume $\text{LWE}(\lambda, \sigma, q, U(\{\pm 1\}^\lambda))$ is hard for parameters $q = q(\lambda)$ and $\sigma = \sigma(\lambda)$ satisfying $\sigma \geq 3\sqrt{\lambda(\ln \lambda + 2\omega(\log \lambda))}$. Then for $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$ satisfying

$$\gamma = \sqrt{\lambda(\ln \lambda + 2\omega(\log \lambda))} \quad \text{and} \quad \beta = O\left(\frac{\sigma}{q}\right),$$

$\text{CLWE}(\lambda, \beta, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ is hard.

Theorem 3.8 gives us a reasonable range of parameters for which we can comfortably assume the hardness of CLWE with normalized binary secrets in λ dimensions. For example, setting $\gamma = \sqrt{3\lambda\omega(\log \lambda)}$ and $\beta = 1/\text{poly}(\lambda)$, any efficient algorithm for CLWE with normalized binary secrets would yield an efficient algorithm for LWE with a polynomial modulus-to-noise ratio. Using existing reductions from standard LWE to binary secret LWE [Mic18], this would in turn yield an efficient algorithm for standard LWE with polynomial modulus-to-noise ratio.

One may wonder whether we can decouple the CLWE parameters from the dimension while preserving security. In particular, when the dimension n is very large, can we set γ and β as functions of the security parameter alone, rather than requiring them to grow with n ? It turns out this is the case for uniform secrets, as seen in the proposition below. The key idea is to lift low-dimensional CLWE samples to a higher-dimensional space: we view the original samples as lying in a uniformly random low-dimensional subspace and add an independent standard Gaussian component in the orthogonal complement.

Proposition 3.1 (Separating CLWE gap and dimension). Let $\lambda \in \mathbb{N}$ be the security parameter and assume $\text{CLWE}(\lambda, \beta, \gamma, U(\mathbb{S}^{\lambda-1}))$ is hard for parameters $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$. Then for any $n = \text{poly}(\lambda) \geq \lambda$, $\text{CLWE}(n, \beta, \gamma, U(\mathbb{S}^{n-1}))$ is hard.

Proof. We prove the contrapositive. Say there is some $n \geq \lambda$ and some PPT adversary $\mathcal{A}$ that achieves advantage $\delta = 1/\text{poly}(\lambda)$ at distinguishing a $\text{CLWE}_{\mathbf{s}'}$ oracle where $\mathbf{s}' \leftarrow \mathbb{S}^{n-1}$ from a null oracle. Then we construct an adversary $\mathcal{B}$ against the lower dimensional problem. At the beginning $\mathcal{B}^O(1^\lambda)$ samples a uniformly random basis for a λ -dimensional subspace of $\mathbb{R}^n$, $\mathbf{U} \leftarrow V_{\lambda,n}$. Let $\mathbf{U}^\perp \in \mathbb{R}^{n \times n-\lambda}$ arbitrarily complete the basis. $\mathcal{B}$ then implements the oracle O' as follows. For each requested sample, $\mathcal{B}$ does as follows:

- Sample $(\mathbf{y}, b) \leftarrow O$
- Sample $\mathbf{g} \leftarrow D_{\mathbb{R}^{n-\lambda},1}$
- Output $(\mathbf{U}\mathbf{y} + \mathbf{U}^\perp\mathbf{g}, b)$

$\mathcal{B}$ then simulates $\mathcal{A}^{O'}(1^\lambda)$ and outputs the bit returned by $\mathcal{A}$. Clearly, when O is the null oracle, O' is a higher dimensional null oracle.

Now suppose O is a $\text{CLWE}_{\mathbf{s}}$ oracle for $\mathbf{s} \leftarrow \mathbb{S}^{\lambda-1}$. Then we know that the first component $\mathbf{U}\mathbf{y} + \mathbf{U}^\perp\mathbf{g}$ where $\mathbf{y} \leftarrow D_{\mathbb{R}^\lambda,1}$ and $\mathbf{g} \leftarrow D_{\mathbb{R}^{n-\lambda},1}$ is distributed as an n -dimensional standard Gaussian vector. We also have that

$$\begin{aligned} b &= \gamma \mathbf{s}^\top \mathbf{y} + e \pmod{1} \\ &= \gamma (\mathbf{U}\mathbf{s})^\top (\mathbf{U}\mathbf{y}) + e \pmod{1} \\ &= \gamma (\mathbf{U}\mathbf{s})^\top (\mathbf{U}\mathbf{y} + \mathbf{U}^\perp\mathbf{g}) + e \pmod{1} \end{aligned}$$

where the second line comes from $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_\lambda$ and the third from $\mathbf{U}^\top \mathbf{U}^\perp = \mathbf{0}$. Therefore, in this case, $(\mathbf{y}, b)$ is exactly a sample from $\text{CLWE}_{\mathbf{U}\mathbf{s}}$. Since $\mathbf{U} \leftarrow V_{\lambda,n}$ and $\mathbf{s} \leftarrow \mathbb{S}^{\lambda-1}$ are uniform, the vector $\mathbf{U}\mathbf{s}$ is uniformly distributed over $\mathbb{S}^{n-1}$. Hence O' is an oracle that outputs n -dimensional CLWE samples with uniformly random secret, and $\mathcal{B}$ inherits $\mathcal{A}$'s distinguishing advantage δ . $\square$

Remark 3.1. *The proposition above is intended primarily to provide intuition; it does not directly establish parameter separation for CLWE with normalized binary secrets. In Section 8, we will instead reduce CLWE with normalized binary secrets to a shifted variant of the Gaussian pancake distinguishing problem with secrets uniform on the sphere. We then use ideas similar to those in the proof above to separate the dimension from the parameters for that problem.*

4 New Definitions

4.1 Fingerprinting

We define a *fingerprinting scheme*, that applies to a fixed training algorithm Train . Intuitively, given a data distribution D and a training specification spec , the fingerprinting algorithm learns weights that, to users without the secret key sk , are *computationally indistinguishable* from those learned by Train with the same inputs. On the other hand, users with sk can detect a fingerprint on such weights.

Definition 4.1 (Training algorithm). *A training algorithm Train is a PPT algorithm that, given a security parameter 1^λ , oracle access to a training distribution D , and a training specification spec (e.g., hyperparameters), outputs a trained model $\mathbf{w} \leftarrow \text{Train}^D(1^\lambda, \text{spec})$.*

Definition 4.2 (Fingerprinting scheme). *A (secret-key) fingerprinting scheme with respect to a training algorithm Train is a tuple of PPT algorithms $(\text{Gen}, \text{FP}, \text{Ver})$ satisfying the following properties:*

- $\text{Gen}(1^\lambda)$ takes as input a security parameter λ and generates a fingerprinting key sk .
- $\text{FP}^D(1^\lambda, \text{sk}, \text{spec})$ takes as input a fingerprinting key sk , oracle access to a training distribution D , and a training specification spec and outputs fingerprinted weights $\mathbf{w}_{\text{FP}}$.
- $\text{Ver}(1^\lambda, \text{sk}, \mathbf{w}')$ takes as input candidate weights $\mathbf{w}'$ and outputs a bit in $\{0, 1\}$ indicating whether or not they contain a fingerprint.

Correctness. For every distribution D and specification spec that are valid inputs to Train ,

$$\Pr_{\substack{\text{sk} \leftarrow \text{Gen}(1^\lambda) \\ \mathbf{w}_{\text{FP}} \leftarrow \text{FP}^D(1^\lambda, \text{sk}, \text{spec})}} [\text{Ver}(1^\lambda, \text{sk}, \mathbf{w}_{\text{FP}}) = 1] \geq 1 - \text{negl}(\lambda).$$

Soundness. For every $n = \text{poly}(\lambda)$ and every fixed $\mathbf{w}' \in \mathbb{R}^n$,

$$\Pr_{\text{sk} \leftarrow \text{Gen}(1^\lambda)} [\text{Ver}(1^\lambda, \text{sk}, \mathbf{w}') = 1] \leq \text{negl}(\lambda).$$

Undetectability. For every PPT distinguisher $\mathcal{A}$,

$$\left| \Pr_{\text{sk} \leftarrow \text{Gen}(1^\lambda)} [\mathcal{A}^{\mathcal{O}_{\text{FP}}(1^\lambda, \text{sk}, \cdot, \cdot)}(1^\lambda) = 1] - \Pr [\mathcal{A}^{\mathcal{O}_{\text{Train}}(1^\lambda, \text{sk}, \cdot, \cdot)}(1^\lambda) = 1] \right| \leq \text{negl}(\lambda).$$

where $\mathcal{O}_{\text{FP}}(1^\lambda, \text{sk}, \cdot, \cdot)$ and $\mathcal{O}_{\text{Train}}(1^\lambda, \cdot, \cdot, \cdot)$ are oracles that take in the description of a distribution D and a specification spec and output a sample from $\text{FP}^D(\text{sk}, \text{spec})$ and $\text{Train}^D(1^\lambda, \text{spec})$ respectively.

Remark 4.1. Our definition of undetectability allows the adversary to adaptively choose the training distribution D and specification spec , and then observe fingerprinted weights $\mathbf{w} \leftarrow \text{FP}^D(1^\lambda, \text{spec})$. This captures the strongest setting we have in mind; for example, a customer may outsource training to a fingerprinter while specifying exactly how they want their weights to be trained.

In addition to correctness and soundness, we would like to capture a class of honest transformations that the model receiver may apply while preserving security. For example, the receiver may fine-tune part of the model to adapt it to their own task. Robustness formalizes this using a two-input predicate Φ : it requires that no adversary can produce weights that are within a Φ modification of fingerprinted weights for which the fingerprint no longer verifies.

Definition 4.3 (Robustness). Let $\Phi : \mathbb{R}^* \times \mathbb{R}^* \rightarrow \{0, 1\}$ be a predicate indicating whether two weight vectors are close and let $\delta : \mathbb{N} \rightarrow [0, 1]$ be the success probability. A fingerprinting scheme $(\text{Gen}, \text{FP}, \text{Ver})$ is (Φ, δ) -robust if for every PPT adversary $\mathcal{A}$,

$$\Pr \left[\begin{array}{l} \text{sk} \leftarrow \text{Gen}(1^\lambda) \\ \mathbf{w}' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{FP}}(1^\lambda, \text{sk}, \cdot, \cdot)}(1^\lambda) \end{array} : \begin{array}{l} \exists \mathbf{w} \in \mathcal{W} \text{ such that } \Phi(\mathbf{w}, \mathbf{w}') = 1 \\ \text{and } \text{Ver}(\text{sk}, \mathbf{w}') = 0 \end{array} \right] \leq \delta(\lambda),$$

where $\mathcal{W}$ denotes the set of fingerprinted weights observed by $\mathcal{A}$ through its queries to $\mathcal{O}_{\text{FP}}$. At times, when δ is a negligible function, we omit it and simply refer to this as Φ -robustness.

On the other hand, we want to ensure that honest model trainers who did not use the service are not falsely flagged. At first glance, soundness may seem sufficient to provide this guarantee. Indeed, if a model owner trains their model *independently* of any fingerprinted models, then soundness guarantees that they will not be falsely flagged. However, outputs from a fingerprinted model may themselves be correlated with the fingerprinting key, since they are functions of the fingerprinted weights. A fingerprinter therefore may want to allow even honest trainers to produce models that have some correlation with their secret key without claiming ownership of those models. Unforgeability addresses this issue by guaranteeing that even a model correlated with the secret key—namely, a model generated by an adversary that has access to fingerprinted models but not the secret key itself—is not falsely attributed to the fingerprinter, provided it is sufficiently far from every fingerprinted model.

Definition 4.4 (Unforgeability). Let $\Psi : \mathbb{R}^* \times \mathbb{R}^* \rightarrow \{0, 1\}$ be a predicate indicating whether two weight vectors are close and let $\delta : \mathbb{N} \rightarrow [0, 1]$ be the success probability. A fingerprinting scheme $(\text{Gen}, \text{FP}, \text{Ver})$ is (Ψ, δ) -unforgeable if for every PPT adversary $\mathcal{A}$,

$$\Pr \left[\begin{array}{l} \text{sk} \leftarrow \text{Gen}(1^\lambda) \\ \mathbf{w}' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{FP}}(1^\lambda, \text{sk}, \cdot, \cdot)}(1^\lambda) \end{array} : \begin{array}{l} \forall \mathbf{w} \in \mathcal{W}, \Psi(\mathbf{w}, \mathbf{w}') = 0 \\ \text{and } \text{Ver}(\text{sk}, \mathbf{w}') = 1 \end{array} \right] \leq \delta(\lambda),$$

where $\mathcal{W}$ denotes the set of fingerprinted weights observed by $\mathcal{A}$ through its queries to $\mathcal{O}_{\text{FP}}$. At times, when δ is a negligible function, we omit it and simply refer to this as Ψ -unforgeability.

As an example of a predicate describing permitted modifications, consider the ℓ_2 predicate, which outputs 1 when the inputs are close in Euclidean distance. In Section 5.4, we discuss transformations to ML model weights that may be captured by the ℓ_2 predicate.

Definition 4.5 (ℓ_2 predicate). *For a threshold $\Delta \in \mathbb{R}$, the predicate $\text{L2}(\Delta) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \{0, 1\}$ is defined as*

$$\text{L2}(\Delta)(\mathbf{w}, \mathbf{w}') = \begin{cases} 1 & \text{if } \|\mathbf{w} - \mathbf{w}'\|_2 \leq \Delta, \\ 0 & \text{otherwise.} \end{cases}$$

Remark 4.2. *Our definitions are strong in the sense that, in all security games, the adversary is given full white-box access to the fingerprinted weights, rather than being restricted to black-box queries to the model. This captures adversaries that directly obtain the model parameters and attempt to modify or distinguish them.*

Remark 4.3. *On the other hand, our definitions have two limitations that may be interesting to address in future work. First, neither the robustness nor the unforgeability adversary is given access to a verification oracle. We view this as reasonable in the semi-honest setting, where honest transformations are not influenced by verification decisions. Nonetheless, it remains an interesting technical question whether security can be achieved against such adversaries. Second, our verification algorithm requires white-box access to the model weights in the clear. This is natural in settings where verification is performed on disputed models whose weights must already be revealed to the verifier, or in the semi-honest setting, where verification can instead be performed via secure MPC. In less trusted settings, however, it would be interesting to study whether verification can be performed using only black-box queries to a model, without revealing to the model holder that the verifier is attempting to verify it.*

4.2 KDS Gaussian pancake and Pancake Alignment

KDS Gaussian pancake. We define the multi-secret, *key-dependent shifted* Gaussian pancake distribution. Its samples are standard Gaussian in most directions, while their projections onto the secret directions are discretized. Geometrically, the distribution consists of parallel “pancakes” that are narrow in the secret directions and wide in the orthogonal directions. In our variant, the center of these pancakes is determined by a key-dependent shift. Formally, samples are drawn as follows.

Definition 4.6 (KDS-Gaussian Pancake distribution). *For $n, \ell \in \mathbb{N}$ such that $1 \leq \ell < n$, real numbers $\beta, \gamma > 0$, a matrix $\mathbf{S} \in V_{\ell, n}$, and vectors $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{z} \in [0, 1)^\ell$, samples from the ℓ -secret key-dependent shifted (KDS) Gaussian pancake distribution $\text{GP}_{\beta, \gamma}^{(\ell)}(\mathbf{S}, \mathbf{u}, \mathbf{z})$ are generated as:*

- For $i \in [\ell]$, sample the pancake $k_i \leftarrow D_{\mathbb{Z} + t_i, \sqrt{\gamma^2 + \beta^2}}$, where the shift $t_i = -\mathbf{S}[i]^\top \mathbf{u} + z_i \bmod 1$.
- For $i \in [\ell]$, sample the noise $e_i \leftarrow D_{\mathbb{R}, \beta'}$ independently, where $\beta'^2 = \beta^2 / (\gamma^2 + \beta^2)$.
- Sample the orthogonal component $\mathbf{v}$ as $D_{\mathbb{R}^{n-\ell}, 1}$ from the subspace orthogonal to $\mathbf{S}$.
- Output $\mathbf{y} = \mathbf{v} + \sum_{i=1}^{\ell} (\frac{k_i}{\gamma'} + e_i) \mathbf{S}[i]$ where $\gamma' = (\gamma^2 + \beta^2) / \gamma$.

At times, we drop the superscript (ℓ) or subscripts β, γ from $\text{GP}_{\beta, \gamma}^{(\ell)}$ when they are clear from context.

Remark 4.4. *The component along each direction $\mathbf{S}[i]$ is distributed as*

$$\left(\frac{k_i}{\gamma'} + e_i \right) \stackrel{d}{=} \frac{\gamma}{\hat{\gamma}^2} \left(D_{\mathbb{Z} + (z_i - \mathbf{S}[i]^\top \mathbf{u} \bmod 1), \hat{\gamma}} + D_{\mathbb{R}, \frac{\gamma \beta}{\hat{\gamma}}} \right), \quad \text{where } \hat{\gamma} = \sqrt{\beta^2 + \gamma^2}.$$

The KDS Gaussian pancake distinguishing task asks an adversary to distinguish samples from the Gaussian pancake distribution from standard Gaussian samples, where the shift may be chosen adaptively.

Definition 4.7 (KDS-Gaussian pancake distinguishing task). *Let $n = n(\lambda)$, $\ell = \ell(\lambda)$, $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$ be parameter families. The Gaussian pancake distinguishing problem $\text{GP}_{\text{shift}}(n, \ell, \beta, \gamma)$ asks to distinguish between a KDS-Gaussian pancake oracle $\text{GP}_{\beta, \gamma}^{(\ell)}(\mathbf{S}, \cdot, \cdot)$ for a random $\mathbf{S} \leftarrow V_{\ell, n}$, and a null oracle $\text{Null}(\cdot, \cdot)$, which ignores its inputs and outputs fresh samples from $D_{\mathbb{R}^n, 1}$.*

The Gaussian pancake distribution without a shift $\text{GP}(\mathbf{S}, \mathbf{0}, \mathbf{0})$, also known as the homogeneous CLWE distribution, was introduced in [BRST21]. They showed a reduction from CLWE to the corresponding Gaussian pancake distinguishing problem. Subsequently, [BNHR22] introduced the shifted Gaussian pancake distribution $\text{GP}(\mathbf{S}, \mathbf{0}, \mathbf{z})$, where the shift has no dependence on the key, and likewise gave a reduction from CLWE. In contrast, we consider a setting in which the adversary may adaptively choose vectors $\mathbf{u}, \mathbf{z}$ and receive samples from $\text{GP}(\mathbf{S}, \mathbf{u}, \mathbf{z})$, where the shift is $-\mathbf{S}^\top \mathbf{u} + \mathbf{z} \pmod{1}$. Nevertheless, we show that this adaptive shifted variant remains hard under the assumed hardness of CLWE.

Proposition 4.1. *Let $\lambda \in \mathbb{N}$ be a security parameter. Assume $\text{CLWE}(\lambda, \beta/\sqrt{2}, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ is hard (Definition 3.7) for parameters $\beta = \beta(\lambda)$ and $\gamma = \gamma(\lambda)$ satisfying $\beta = 1/\text{poly}(\lambda)$ and $\gamma \geq \sqrt{3\lambda\omega(\log \lambda)}$. Then for every dimension $n = \text{poly}(\lambda) \geq \lambda$ and number of secrets $\ell = \text{poly}(\lambda)$ such that $n - \ell \geq \lambda$, $\text{GP}_{\text{shift}}(n, \ell, \beta, \gamma)$ is hard.*

We prove this proposition in Section 8, where we include a more careful analysis of the advantage.

Pancake alignment We introduce a new problem, the (key-dependent shifted) *pancake alignment problem*. At a high level, the problem asks, given oracle access to (KDS) Gaussian pancake samples for a random, unknown secret $\mathbf{S}$, find a direction $\mathbf{v}$ aligned with $\mathbf{S}$, that is, $\mathbf{v}$ has large inner product with many of the secret directions given by the columns of $\mathbf{S}$.

Definition 4.8 (Pancake alignment problem). *Let $n, \ell \in \mathbb{N}$ be the dimension and number of secrets, let $\gamma > 0$ and $\beta \in (0, 1)$ be pancake parameters, and let $t \in \mathbb{R}$ with $1 \leq t \leq n$ be the approximation parameter. The pancake alignment problem, $t\text{-PA}(n, \ell, \beta, \gamma)$, is defined as follows. Given oracle access to KDS pancake samples $\text{GP}_{\beta, \gamma}^{(\ell)}(\mathbf{S}, \cdot, \cdot)$ for a random secret $\mathbf{S} \leftarrow V_{\ell, n}$, output a direction $\mathbf{v} \in \mathbb{S}^{n-1}$ aligned with the secret,*

$$\|\mathbf{S}^\top \mathbf{v}\|_2^2 \geq t \frac{\ell}{n}.$$

Definition 4.9 (Pancake alignment hardness). *Let $t = t(\lambda)$, $n = n(\lambda)$, $\ell = \ell(\lambda)$, $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$ be parameter families and let $\delta = \delta(\lambda)$ be the advantage. We say a PPT algorithm $\mathcal{A}$ solves $t\text{-PA}(n, \ell, \beta, \gamma)$ with advantage δ if they succeed at solving the pancake alignment problem with probability δ taken over $\mathbf{S} \leftarrow V_{\ell, n}$, as well as the randomness of $\mathcal{A}$ and its oracle. We say $t\text{-PA}(n, \ell, \beta, \gamma)$ is hard if every PPT adversary solves the problem with at most negligible advantage.*

Recall that the bound $t(\ell/n)$ comes from the fact that $\|\mathbf{S}^\top \mathbf{v}\|_2^2 = \Theta(\ell/n)$ with high probability over $\mathbf{S}$ when $\mathbf{v}$ is fixed and independent of $\mathbf{S}$. Asking that the problem be hard means that Gaussian pancake samples do not leak enough information about the secret $\mathbf{S}$ to make the alignment task significantly easier. In Section 7, we give reductions from CLWE to pancake alignment for certain approximations t . In particular, for $t \geq \sqrt{\lambda\omega(\log \lambda)}$, we give a reduction from the GP_{shift} distinguishing task, which in turn yields a reduction from CLWE with normalized binary secrets. This is interesting in our setting, where the dimension n will be the number of weights in a machine learning model and may be much larger than the security parameter λ .

Theorem 4.1. *Let $\lambda \in \mathbb{N}$ be a security parameter. Assume $\text{CLWE}(\lambda, \beta/\sqrt{2}, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ is hard (Definition 3.7) for parameters $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$ satisfying $\beta = 1/\text{poly}(\lambda)$ and $\gamma \geq \sqrt{3\lambda\omega(\log \lambda)}$. Then for any $n = \text{poly}(\lambda)$ and $\ell = \text{poly}(\lambda)$ such that $n - \ell \geq \lambda$ and $\ell = o(n)$, there exists some $t = t(\lambda)$ satisfying*

$$t = O(\sqrt{\lambda} \log \lambda)$$

such that $t\text{-PA}(n, \ell, \beta, \gamma)$ is hard.

Proof. By Proposition 4.1, $\text{GP}_{\text{shift}}(\lambda, 1, \beta, \gamma)$ is hard. The claim then follows from Proposition 7.2. $\square$

We also give a reduction from CLWE when $t \approx 1/\epsilon$, where ϵ denotes the PA adversary's advantage. In our application, the adversary's advantage will later translate into the failure probability of robustness in our fingerprinting scheme. In many ML applications, negligible failure probability may not be necessary; an inverse-polynomial or even small constant failure probability may be acceptable. We present the Theorem for single secret ($\ell = 1$) pancake alignment, but one could additionally extend it to multiple secrets using Lemma 7.3.

Theorem 4.2. *Let $\lambda \in \mathbb{N}$ be a security parameter. Assume $\text{CLWE}(\lambda, \beta/\sqrt{2}, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ is hard for parameters $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$ satisfying $\beta = 1/\text{poly}(\lambda)$ and $\gamma \geq \sqrt{3\lambda\omega(\log \lambda)}$. Then for any $\epsilon = \epsilon(\lambda)$ and any $n = \text{poly}(\lambda) \geq \lambda$, there exists some t satisfying*

$$t = \frac{1}{\epsilon}(1 + o(1))$$

such that no PPT adversary solves $t\text{-PA}(n, 1, \beta, \gamma)$ with advantage more than ϵ .

Proof. By Proposition 4.1, $\text{GP}_{\text{shift}}(\lambda, 1, \beta, \gamma)$ is hard. The claim follows from Proposition 7.3. $\square$

The statements above provide strong evidence of hardness for certain values of t , based on reductions from well-studied hard problems. However, our best known attack achieves only negligible advantage once $t = \Theta(1)$ is some constant. We therefore conjecture hardness for a broader range of t than what our reductions formally establish.

Assumption 4.3. *For any $n = \text{poly}(\lambda) \geq \lambda$, any $\ell = \ell(\lambda)$ such that $n - \ell \geq \lambda$ and $\ell = o(n)$, any $\beta = 1/\text{poly}(\lambda)$, and any $\gamma = \gamma(\lambda) \geq \sqrt{3\lambda\omega(\log \lambda)}$, there exists some constant $t = \Theta(1)$ such that $t\text{-PA}(n, \ell, \beta, \gamma)$ is hard.*

5 Robust Fingerprinting

Here, we give a construction of an ℓ_2 -robust fingerprinting scheme for noisy training algorithms. In Section 5.1, we formally define noisy training algorithms and discuss natural learning algorithms that fall within this class. In Section 5.2, we show how to embed messages into the outputs of noisy training algorithms. In Section 5.3, we present our construction and state its guarantees. Finally, in Section 5.4, we discuss honest weight transformations that are captured by ℓ_2 -robustness.

5.1 Noisy training algorithms

We give a fingerprinting scheme that applies to any algorithm in the class $(\sigma, n)\text{-NoisyTrain}$, which is parameterized by a noise level σ and a dimension n . At a high level, this class captures training procedures whose output weights contain n coordinates that can be expressed as the sum of a Gaussian component and a component sampled independently of that Gaussian.

More precisely, a training algorithm belongs to $(\sigma, n)\text{-NoisyTrain}$ if it produces weights in $\mathbb{R}^{n'}$ (for some $n' \geq n$) via a process equivalent to the following:

1. Sample the first n coordinates $\mathbf{w}[1 : n]$.
2. Independently sample a noise vector $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$.
3. Sample the remaining coordinates $\mathbf{w}[n + 1, n']$, possibly depending on $\mathbf{g}$.

The final output weights are then $\mathbf{w} + (\mathbf{g}, \mathbf{0})$ —that is, the noise $\mathbf{g}$ is added to the first n coordinates which were sampled independently of $\mathbf{g}$. We formalize this class below, and then give some examples of learning algorithms that fall within it.

Definition 5.1 (*NoisyTrain*). *Let $\lambda \in \mathbb{N}$ be the security parameter, let $\sigma = \sigma(\lambda)$ be the noise level, and let $n = n(\lambda)$ be the number of noisy coordinates. A PPT training algorithm Train is in the class $(\sigma, n)\text{-NoisyTrain}$ if there exists a subset of coordinates I with $|I| = n$ and a pair of PPT algorithms (SampleFirst , SampleRest) such that for every valid training specification spec and data distribution D , the vector $\mathbf{w}^*$ generated by sampling*

$$(\mathbf{w}_I, \text{state}) \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec}), \quad \mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}, \quad \mathbf{w}_{-I} \leftarrow \text{SampleRest}^D(1^\lambda, \text{spec}, \text{state}, \mathbf{g}),$$

then setting $\mathbf{w}_I^* = \mathbf{w}_I + \mathbf{g}$ and $\mathbf{w}_{-I}^* = \mathbf{w}_{-I}$ is distributed the same as standard training,

$$\mathbf{w}^* \stackrel{d}{=} \mathbf{w}_{\text{Train}}, \quad \text{where } \mathbf{w}_{\text{Train}} \leftarrow \text{Train}^D(1^\lambda, \text{spec}).$$

Remark 5.1. *The auxiliary state allows `SampleRest` to continue a partial computation performed by `SampleFirst`. For example, `SampleFirst` may sample the entire vector $\mathbf{w}$ independently of the noise $\mathbf{g}$, output only $\mathbf{w}_I$, and pass $\mathbf{w}$ to `SampleRest` through the state. `SampleRest` may then fine-tune $\mathbf{w}_{-I}$, possibly depending on $\mathbf{g}$. For example, it could use $\mathbf{w}_I + \mathbf{g}$ as the weights of an earlier layer while fine-tuning the remaining coordinates.*

We list examples of natural training algorithms that admit such decompositions.

Differentially private gradient descent via the Gaussian mechanism [ACGM+16] Consider gradient descent in which Gaussian noise is added to the gradient updates¹¹. The training algorithm obtains a dataset $(\mathbf{X}, \mathbf{y}) \leftarrow D$, arbitrarily initializes $\mathbf{w}^{(0)} \in \mathbb{R}^n$, and computes

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \left(\nabla L_{\mathbf{X}, \mathbf{y}}(\mathbf{w}^{(t-1)}) + \mathbf{g}^{(t)} \right), \quad \mathbf{g}^{(t)} \leftarrow D_{\mathbb{R}^n, \sigma}.$$

Here L is a loss function and $\eta \in \mathbb{R}$ is the step size. After T iterations, the output is $\mathbf{w}^{(T)} \in \mathbb{R}^n$. This algorithm falls in $(\eta\sigma, n)$ -NoisyTrain with noise added to every coordinate. Concretely, `SampleFirst` ^{$D(1^\lambda, \text{spec})$} runs the first $T - 1$ iterations as above and outputs the noiseless part of the final update,

$$\mathbf{w} = \mathbf{w}^{(T-1)} - \eta \nabla L_{\mathbf{X}, \mathbf{y}}(\mathbf{w}^{(T-1)}),$$

while `SampleRest` is trivial (there are no remaining coordinates to sample). Since $\mathbf{g}^{(T)} \leftarrow D_{\mathbb{R}^n, \sigma}$ implies $\eta \mathbf{g}^{(T)} \leftarrow D_{\mathbb{R}^n, \eta\sigma}$, we may equivalently sample the final noise as $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \eta\sigma}$ and output $\mathbf{w} + \mathbf{g}$, which is identically distributed to $\mathbf{w}^{(T)}$. This same construction extends naturally to the case where noise is added to only a subset of coordinates. We additionally show in Section B that in certain settings, noise accumulates across training so that the algorithm falls in the class (σ, n) -Train for a larger effective noise level σ .

One caveat for this application is that recent work [SH26] shows that replacing Gaussian noise with Gaussian pancake noise in private training allows the secret key holder to extract private information from the resulting weights. Thus, fingerprinting such an algorithm requires assuming that the auditor and model provider, who hold sk , do not exploit it for this purpose. Since the model provider already performed the private training and the auditor is assumed to be honest, we do not view this as a major additional assumption. Moreover, to any party without the key, (e.g., the general public) the resulting weights remain computationally indistinguishable from private weights.

Noisy data. There is a body of work that studies training on data that itself is naturally noise [SKPS+22]. Moreover, there is a body of work on artificial noise injection, which provides evidence that *intentionally* adding noise to data may improve generalization [Bis95] or adversarial robustness [CRK19]. In particular, a training algorithm may use data $(\mathbf{X}', \mathbf{y}') \leftarrow D_{\text{noisy}}$, where

$$\mathbf{X}' \stackrel{d}{=} \mathbf{X} + \Xi \quad \text{and} \quad \mathbf{y}' \stackrel{d}{=} \mathbf{y} + \xi,$$

with $(\mathbf{X}, \mathbf{y}) \leftarrow D$ denoting the underlying clean data and Ξ and ξ denoting independent noise.

Consider a setting in which the trainer has access to clean data but is willing to inject independent noise during training; for example, to improve generalization, or because the guarantees of training on noisy data suffice for their application. We show in Section B that, for certain loss functions and Gaussian Ξ, ξ , this noise propagates through gradient descent linearly, so that the final weights decompose as the weights obtained by training on the clean data, plus an independent additive Gaussian component. Such training algorithms are compatible with our fingerprinting scheme.

Random weights. In some training procedures, a subset of the weights may themselves be random. For example, the random feature model [RR07] uses a fixed layer of Gaussian random weights, with only the final layer's weights trained. Any learning algorithm using this model is in the class $(1, n)$ -NoisyTrain: `SampleFirst` outputs the $\mathbf{0}$ vector (since this part of the weights is not learned at all), we sample the noise $\mathbf{g} \leftarrow D_{\mathbb{R}^n, 1}$, and `SampleRest` trains the remaining weights using $\mathbf{g}$ as the random layer.

¹¹We omit per-example gradient clipping for simplicity; it does not affect the argument below.

This observation also suggests a heuristic extension of our fingerprinting scheme beyond the setting covered by our formal guarantees: even when a model’s random weights are not exactly Gaussian, one can apply the same fingerprinting procedure as if they were. Our formal guarantees would no longer necessarily hold, but this offers a natural heuristic approach for fingerprinting models with sufficiently random weights.

Injecting noise into parameters. It may be that the training algorithm itself does not naturally admit a Gaussian decomposition, but the trainer is willing to add Gaussian noise to a subset of the weights after training; for example, under the heuristic that this noise will not significantly affect model quality, or that noise may induce some desired behavior [AJVZ26, Section 3]. In this case, **SampleFirst** and **SampleRest** run the original training algorithm to completion then the trainer adds the noise $\mathbf{g}$ post hoc to the component given by **SampleFirst**.

5.2 Embedding messages into weights

Our core technique is to take the continuous Gaussian vector $\mathbf{g}$ present in the decomposition of the weights, and slightly perturb it to obtain a “pseudogaussian” fingerprinted vector $\mathbf{g}_{\text{FP}}$. The perturbation embeds a message $\mathbf{z}$ that can be recovered from $\mathbf{w} + \mathbf{g}_{\text{FP}}$, while the marginal distribution of $\mathbf{g}_{\text{FP}}$ remains computationally indistinguishable from that of $\mathbf{g}$.

More specifically, we will “round” $\mathbf{g}$ to a vector that is statistically close to a KDS-Gaussian pancake vector $\text{GP}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}, \mathbf{z})$. We show in Lemma 5.1 that, after appropriately scaling the weights and projecting onto the secret directions, the embedded message can be approximately decoded.

$$\frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w} + \mathbf{g}_{\text{FP}}) = \mathbf{z} + \text{small noise} \pmod{1}.$$

We state our embedding algorithm, **Embed**($\mathbf{S}, \mathbf{u}, \mathbf{z}, \mathbf{g}$), and its guarantees below. It is inspired by the discrete Gaussian sampler of [Pei10].

Embed($\mathbf{S}, \mathbf{u}, \mathbf{z}, \mathbf{g}$)

- 1: Let $\hat{\gamma} := \sqrt{\gamma^2 + \beta^2}$
- 2: For $i \in [\ell]$ denote $z'_i := z_i - \mathbf{S}[i]^\top \mathbf{u}$
- 3: For $i \in [\ell]$, let $t_i = (z'_i \bmod 1) - \frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g}$
- 4: For $i \in [\ell]$, sample $h_i \leftarrow D_{\mathbb{Z}+t_i, \frac{\hat{\gamma}}{\sqrt{2}}}$, $e_i \leftarrow D_{\mathbb{R}, \frac{\hat{\gamma}\beta}{\gamma}}$
- 5: Compute

$$\mathbf{g}_{\text{FP}} = (\mathbf{I} - \mathbf{S}\mathbf{S}^\top) \mathbf{g} + \frac{\sigma\gamma}{\hat{\gamma}^2} \sum_{i=1}^{\ell} \left(\frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g} + h_i + e_i \right) \mathbf{S}[i]$$

6: **return** $\mathbf{g}_{\text{FP}}$

Remark 5.2. One may wonder why we do not directly sample $\mathbf{g}_{\text{FP}}$ from the distribution $\text{GP}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}, \mathbf{z})$, rather than first sampling $\mathbf{g}$ and then rounding it to a Gaussian pancake vector. Indeed, this would suffice for our construction. The benefit of our approach is that it ensures that $\mathbf{w} + \mathbf{g}$ remains close in Euclidean distance to the fingerprinted output $\mathbf{w} + \mathbf{g}_{\text{FP}}$ (Lemma 5.4). This closeness will be useful in Section 6, where we modify the fingerprint to achieve unforgeability.

For all statements in this section, let $n, \ell \in \mathbb{N}$ with $1 \leq \ell < n$ be the dimension and number of secrets respectively, and let $\sigma, \gamma \in \mathbb{R}_{>0}$, and $\beta \in (0, 1)$ be parameters. We use **Embed** to denote the algorithm **Embed** instantiated with these parameters. Unless otherwise stated, the statements hold without any other restrictions on the parameters.

First, we show that the embedded message $\mathbf{z}$ can be approximately recovered from the embedded weights $\mathbf{w} + \mathbf{g}_{\text{FP}}$.

Lemma 5.1 (Approximate recovery of embedded message). *Let the secret $\mathbf{S} \in V_{\ell,n}$, the message $\mathbf{z} \in [0,1]^\ell$, the weights $\mathbf{w} \in \mathbb{R}^n$, and the input $\mathbf{g} \in \mathbb{R}^n$ be fixed. Let*

$$\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}, \mathbf{z}, \mathbf{g}).$$

Then

$$\frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w} + \mathbf{g}_{\text{FP}}) = \mathbf{z} + \mathbf{k} + \mathbf{e}, \quad \text{where } \mathbf{k} \in \mathbb{Z}^\ell \text{ and } \mathbf{e} \leftarrow D_{\mathbb{R}^\ell, \frac{\hat{\gamma}\beta}{\gamma}} \text{ conditioned on } \mathbf{k}.$$

Proof. For fixed $i \in [\ell]$, the i th coordinate satisfies

$$\left(\frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w} + \mathbf{g}_{\text{FP}}) \right)_i = \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}[i]^\top \mathbf{w} + \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}[i]^\top \mathbf{g}_{\text{FP}}.$$

By definition of $\mathbf{g}_{\text{FP}}$, this is equal to

$$\frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}[i]^\top \mathbf{w} + \frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g} + h_i + e_i$$

for some $h_i \in \mathbb{Z} + (z_i - \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}[i]^\top \mathbf{w} \bmod 1) - \frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g}$ and some $e_i \leftarrow D_{\mathbb{R}, \frac{\hat{\gamma}\beta}{\gamma}}$. Since

$$\left(\frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}[i]^\top \mathbf{w} + \frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g} + h_i \right) = z_i \pmod{1},$$

we get that,

$$\left(\frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w} + \mathbf{g}_{\text{FP}}) \right)_i = z_i + k_i + e_i, \quad \text{for some } k_i \in \mathbb{Z}.$$

□

Next, we show that the distribution of $\mathbf{g}_{\text{FP}}$ is statistically indistinguishable from a KDS-Gaussian pancake vector. This will later imply that under CLWE, $\mathbf{g}_{\text{FP}}$ is *computationally* indistinguishable from a continuous Gaussian. This relies on the following Lemma, that shows we can “round” a Gaussian vector to a discrete Gaussian.

Lemma 5.2 ([Pei10], Special case of Theorem 3.1). *Let $\sigma_1, \sigma_2 > 0$ be standard deviations such that $\sigma_1 > \eta_\epsilon(\mathbb{Z})$. Let $z \in \mathbb{R}$ be arbitrary. Consider the experiment:*

$$\text{Choose } x_2 \leftarrow D_{\mathbb{R}, \sigma_2} \text{ then choose } x_1 \leftarrow x_2 + D_{\mathbb{Z}+z-x_2, \sigma_1}.$$

Then the marginal distribution of x_1 is within 8ϵ in statistical distance of $D_{\mathbb{Z}+z, \sqrt{\sigma_1^2 + \sigma_2^2}}$.

Lemma 5.3 (Rounding preserves the KDS-Gaussian pancake distribution). *Let $\epsilon \in (0, 1)$ be a parameter. Let $\mathbf{S} \in V_{\ell,n}$, $\mathbf{z} \in [0,1]^\ell$, and $\mathbf{u} \in \mathbb{R}^n$ be arbitrary random variables. Assume $\frac{\hat{\gamma}}{\sqrt{2}} \geq \eta_\epsilon(\mathbb{Z})$. Let*

$$\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}(\mathbf{S}, \mathbf{u}, \mathbf{z}, \mathbf{g}), \quad \text{where } \mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$$

Then $\mathbf{g}_{\text{FP}}$ is within $8\epsilon\ell$ statistical distance of $\sigma\mathbf{y}$, where $\mathbf{y} \leftarrow \text{GP}_{\beta, \gamma}^{(\ell)}(\mathbf{S}, \mathbf{u}, \mathbf{z})$.

Proof. Since the $\mathbf{S}[i]$ ’s are orthonormal, the random variables $\frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g}$ for $i \in [\ell]$ are independent $D_{\mathbb{R}, \frac{\hat{\gamma}}{\sqrt{2}}}$ and are independent of the orthogonal subspace $(\mathbf{I} - \mathbf{S}\mathbf{S}^\top)\mathbf{g}$. We can rewrite

$$\mathbf{g}_{\text{FP}} \stackrel{d}{=} (\mathbf{I} - \mathbf{S}\mathbf{S}^\top)\mathbf{g} + \frac{\sigma\gamma}{\hat{\gamma}^2} \sum_{i=1}^{\ell} (g_i^* + h_i^* + e_i) \mathbf{S}[i], \quad \text{where } g_i^* \leftarrow D_{\mathbb{R}, \frac{\hat{\gamma}}{\sqrt{2}}}, h_i^* \leftarrow D_{\mathbb{Z}+(z_i - \mathbf{S}[i]^\top \mathbf{u} \bmod 1) - g_i^*, \frac{\hat{\gamma}}{\sqrt{2}}}.$$

Lemma 5.2 tells us that for each $i \in [\ell]$, $g_i^* + h_i^*$ is 8ϵ close in statistical distance to a sample from $D_{\mathbb{Z}+(z_i - \mathbf{S}[i]^\top \mathbf{u} \bmod 1), \hat{\gamma}}$. Following the observation made in Remark 4.4, triangle inequality gives us the result. □

Finally, we show that rounding $\mathbf{g}$ to $\mathbf{g}_{\text{FP}}$, has the additional benefit that $\mathbf{g}_{\text{FP}}$ remains close in ℓ_2 distance to the initially sampled $\mathbf{g}$. In particular, their distance is roughly $\sigma\sqrt{\ell}$, despite both vectors lying in $\mathbb{R}^n$. In our application, $n \gg \ell$, since n is the number of model weights while ℓ need only be $\omega(\log \lambda)$.

Lemma 5.4 (Rounding introduces a small ℓ_2 perturbation.). *Let $\mathbf{S} \in V_{\ell,n}$, $\mathbf{z} \in [0,1)^\ell$, and $\mathbf{u} \in \mathbb{R}^n$ be fixed. Let $\ell, \hat{\gamma}$ satisfy $\frac{\hat{\gamma}}{\sqrt{2}} \geq \sqrt{\frac{\ln(6\ell)}{\pi}}$. Let*

$$\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}(\mathbf{S}, \mathbf{u}, \mathbf{z}, \mathbf{g}), \quad \text{where } \mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}.$$

Then there exists a constant $C > 0$ such that

$$\Pr \left[\|\mathbf{g} - \mathbf{g}_{\text{FP}}\|_2 > \sigma\sqrt{C\ell} \right] \leq 2e^{-\ell} + 3 \cdot 4^{-\ell}.$$

Proof. We can rewrite $\mathbf{g} = (\mathbf{I} - \mathbf{S}\mathbf{S}^\top)\mathbf{g} + \sum_{i=1}^\ell (\mathbf{S}[i]^\top \mathbf{g})\mathbf{S}[i]$ since the columns of $\mathbf{S}$ are orthonormal. By the definition of $\mathbf{g}_{\text{FP}}$, we have

$$\mathbf{g}_{\text{FP}} = (\mathbf{I} - \mathbf{S}\mathbf{S}^\top)\mathbf{g} + \frac{\sigma\gamma}{\hat{\gamma}^2} \sum_{i=1}^\ell \left(\frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g} + h_i + e_i \right) \mathbf{S}[i],$$

where $h_i \leftarrow D_{\mathbb{Z}+a_i(\mathbf{g})}$ for $a_i(\mathbf{g}) = (z_i - \mathbf{S}[i]^\top \mathbf{u} \bmod 1) - \frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}[i]^\top \mathbf{g}$ and $e_i \leftarrow D_{\mathbb{R}, \frac{\hat{\gamma}\beta}{\gamma}}$. Therefore, their difference satisfies

$$\mathbf{g} - \mathbf{g}_{\text{FP}} = \sum_{i=1}^\ell \left(\left(1 - \frac{\gamma}{\hat{\gamma}\sqrt{2}}\right) \mathbf{S}[i]^\top \mathbf{g} - \frac{\sigma\gamma}{\hat{\gamma}^2} h_i - \frac{\sigma\gamma}{\hat{\gamma}^2} e_i \right) \mathbf{S}[i].$$

Since rotations preserve Euclidean norm, we can analyze the norm of $\mathbf{g} - \mathbf{g}_{\text{FP}}$ after its been rotated using the rotation that maps $\mathbf{S}[i]$ to the i th canonical vector $\mathbf{b}^{(i)}$. This tells us that

$$\|\mathbf{g} - \mathbf{g}_{\text{FP}}\|_2 = \left\| \left(1 - \frac{\gamma}{\hat{\gamma}\sqrt{2}}\right) \mathbf{S}^\top \mathbf{g} - \frac{\sigma\gamma}{\hat{\gamma}^2} (\mathbf{h} + \mathbf{e}) \right\|_2$$

where $\mathbf{S}^\top \mathbf{g} \leftarrow D_{\mathbb{R}^\ell, \sigma}$, $\mathbf{h} \leftarrow D_{\mathbb{Z}^\ell + \mathbf{a}(\mathbf{g}), \frac{\hat{\gamma}}{\sqrt{2}}}$ for $\mathbf{a}(\mathbf{g}) = (\mathbf{z} - \mathbf{S}^\top \mathbf{u} \bmod 1) - \frac{\hat{\gamma}}{\sqrt{2}\sigma} \mathbf{S}^\top \mathbf{g}$, and $\mathbf{e} \leftarrow D_{\mathbb{R}^\ell, \frac{\hat{\gamma}\beta}{\gamma}}$.

We bound each term separately. By Lemma 3.4, $\|\mathbf{S}^\top \mathbf{g}\|_2 > \sigma\sqrt{\frac{5}{2\pi}}\ell$ with probability at most $e^{-\ell}$ and $\|\mathbf{e}\|_2 > \frac{\hat{\gamma}\beta}{\gamma}\sqrt{\frac{5}{2\pi}}\ell$ with probability at most $e^{-\ell}$. Moreover, by Lemma 3.2, $\frac{\hat{\gamma}}{\sqrt{2}} \geq \eta_{1/2}(\mathbb{Z}^\ell)$, so by Lemma 3.3, conditioned on $\mathbf{g}$, $\mathbf{h}$ satisfies $\|\mathbf{h}\|_2 > \frac{\hat{\gamma}}{\sqrt{2}}\sqrt{\ell}$ with probability at most $3 \cdot 4^{-\ell}$. This means the same is true with probability taken over $\mathbf{g}$ and $\mathbf{h}$.

A union bound gives us that with probability at least $1 - (2e^{-\ell} + 3 \cdot 4^{-\ell})$,

$$\begin{aligned} \|\mathbf{g} - \mathbf{g}_{\text{FP}}\|_2 &\leq \left(1 - \frac{\gamma}{\hat{\gamma}\sqrt{2}}\right) \sigma\sqrt{\frac{5}{2\pi}}\ell + \frac{\sigma\gamma}{\hat{\gamma}^2} \left(\frac{\hat{\gamma}}{\sqrt{2}}\sqrt{\ell} + \frac{\hat{\gamma}\beta}{\gamma}\sqrt{\frac{5}{2\pi}}\ell\right) \\ &\leq \left(2\sqrt{\frac{5}{2\pi}} + \frac{1}{\sqrt{2}}\right) \sigma\sqrt{\ell}, \end{aligned}$$

using the fact that $\gamma \leq \hat{\gamma}$ and $\beta \leq \hat{\gamma}$. □

5.3 Fingerprinting construction

Equipped with this tool, we can construct a fingerprinting scheme for any algorithm in the class (σ, n) -NoisyTrain (Definition 5.1). The key consists of random secret $\mathbf{S} \leftarrow V_{\ell,n}$ and message $\mathbf{z} \leftarrow [0,1)^\ell$. To fingerprint n -dimensional weights $\mathbf{w}_I$, we generate $\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}_I, \mathbf{z}, \mathbf{g})$ and output $\mathbf{w}_I + \mathbf{g}_{\text{FP}}$. To verify, we check whether we can approximately recover the randomly sampled $\mathbf{z}$ from the corresponding

indices of the input weights $\mathbf{w}'_I$. If so, we declare the weights to be fingerprinted. In contrast, fixed weights that are independent of $\mathbf{z}$ are unlikely to pass this check.

Let Train be in the class (σ, n) -NoisyTrain and denote $(\text{SampleFirst}, \text{SampleRest})$ its decomposition and I the subset of coordinates which are generated by SampleFirst , as per Definition 5.1. Our scheme is parameterized by pancake parameters γ, β, ℓ .

Gen(1^λ)

- 1: Compute $\ell = \ell(\lambda), n = n(\lambda), \gamma = \gamma(\lambda), \beta = \beta(\lambda), \sigma = \sigma(\lambda)$.
- 2: Sample $\mathbf{S} \leftarrow V_{\ell, n}$.
- 3: Sample $\mathbf{z} \leftarrow [0, 1]^\ell$.
- 4: **return** $(\mathbf{S}, \mathbf{z})$.

FP ^{D} (($\mathbf{S}, \mathbf{z}$), spec)

- 1: Compute $(\mathbf{w}_I, \text{state}) \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$.
 - 2: Sample $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$.
 - 3: Let $\hat{\gamma} = \sqrt{\gamma^2 + \beta^2}$.
 - 4: Sample, as in Section 5.2,
- $$\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}\left(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}_I, \mathbf{z}, \mathbf{g}\right).$$
- 5: Compute $\mathbf{w}_{-I} \leftarrow \text{SampleRest}^D(1^\lambda, \text{spec}, \text{state}, \mathbf{g}_{\text{FP}})$.
 - 6: Let $\mathbf{w}_{\text{FP}}$ be the vector such that $(\mathbf{w}_{\text{FP}})_I = \mathbf{w}_I + \mathbf{g}_{\text{FP}}$ and $(\mathbf{w}_{\text{FP}})_{-I} = \mathbf{w}_{-I}$.
 - 7: **return** $\mathbf{w}_{\text{FP}}$.

The verification algorithm is parameterized by a threshold $\alpha \in (0, 1)$.

Ver _{α} (($\mathbf{S}, \mathbf{z}$), $\mathbf{w}'$)

- 1: Compute $\boldsymbol{\zeta} = (\frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top \mathbf{w}'_I - \mathbf{z}) \text{ cmod } 1 \in [-1/2, 1/2)^\ell$.
- 2: **return** 1 if $\|\boldsymbol{\zeta}\|_2 \leq \alpha\sqrt{\ell}$, 0 otherwise.

Remark 5.3. The scheme above is presented with respect to (σ, n) -NoisyTrain algorithms for which a subset of the weights can be expressed as a base model plus Gaussian noise. Importantly, however, $\mathbf{g}$ need not be Gaussian for the correctness or soundness guarantees. Thus, heuristically, one could apply the construction directly to naturally trained weights: treat the naturally trained weights as $\mathbf{g}$, set $\mathbf{w}_I = \mathbf{0}$, and fingerprint them using the same procedure. If the naturally trained weights are sufficiently random and resemble a Gaussian distribution, then one may heuristically hope for analogous robustness and undetectability guarantees.

5.3.1 Construction guarantees

Theorem 5.5 (Non-robust fingerprinting). *Let $\lambda \in \mathbb{N}$ be the security parameter. Assume $\text{CLWE}(\lambda, \beta/\sqrt{2}, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ is hard (Definition 3.7) for parameters $\gamma = \gamma(\lambda)$ and $\beta = \beta(\lambda)$ satisfying*

$$\gamma \geq \sqrt{3\lambda\omega(\log \lambda)} \quad \text{and} \quad \frac{\sqrt{5(\gamma^2 + \beta^2)}\beta}{\sqrt{2\pi}\gamma} < \frac{\sqrt{6}}{12}.$$

Let $n = \text{poly}(\lambda)$ with $n = \omega(\lambda)$ be the number of noisy weights and let $\sigma = \sigma(\lambda)$ be the noise level of the training algorithm. Let Train be any training algorithm in the class (σ, n) -NoisyTrain (Definition 5.1). Set the number of secrets $\ell = \omega(\log \lambda)$ such that $n - \ell \geq \lambda$. For any threshold α such that $\frac{\sqrt{5(\gamma^2 + \beta^2)}\beta}{\sqrt{2\pi}\gamma} < \alpha \leq \frac{\sqrt{6}}{12}$, the construction $(\text{Gen}, \text{FP}, \text{Ver}_\alpha)$ is an undetectable fingerprinting scheme (Definition 4.2) with respect to Train .

Remark 5.4 (Concrete parameters). *Our fingerprinting scheme can be instantiated with parameters $\gamma = \sqrt{3\lambda \log \lambda}$ and $\beta = \Theta(1/\sqrt{\lambda})$. Using the reduction from binary-secret LWE to CLWE of [GVV22] (Theorem 3.8), security follows from the hardness of binary-secret LWE in dimension λ with noise $\sigma = \Theta(\sqrt{\lambda \log \lambda})$ and modulus $q = \Theta(\lambda \log \lambda)$. The noise rate $\sigma/q = \Theta(1/\sqrt{\lambda})$ is relatively large, so security can be based on small dimensions λ . Instantiating with $\lambda = 512$, $\sigma = 100$, and $q = 10000$, the LWE estimator of [APS15] reports a cost of 2^{147} operations for the best attack.*

Proof. Correctness and soundness follow from Lemmas 5.6 and 5.8. For undetectability, note that by Proposition 4.1, $\text{GP}_{\text{shift}}(n, \ell, \beta, \gamma)$ is hard. Now, suppose for contradiction, that there exists a PPT distinguisher $\mathcal{A}^{O(\cdot)}$ that distinguishes $\mathcal{O}_{\text{Train}}$ from $\mathcal{O}_{\text{FP}}$ with inverse polynomial advantage. We construct a PPT adversary $\mathcal{B}$ against the GP_{shift} problem (Definition 4.7). In particular, $\mathcal{B}$ is given oracle access to O' , which is either the null oracle Null or the Gaussian pancake oracle $\text{GP}(\mathbf{S}, \cdot, \cdot)$. $\mathcal{B}$ samples $\mathbf{z} \leftarrow [0, 1]^\ell$. To simulate a query (D, spec) to $\mathcal{A}$'s oracle, $\mathcal{B}$ computes $(\mathbf{w}_I, \text{state}) \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$ and then samples

$$\mathbf{y} \leftarrow O' \left(\frac{\hat{\gamma}^2}{\gamma \sigma} \mathbf{w}_I, \mathbf{z} \right).$$

$\mathcal{B}$ then samples the rest $\mathbf{w}_{-I} \leftarrow \text{SampleRest}^D(1^\lambda, \text{spec}, \text{state}, \sigma \mathbf{y})$ and returns $\mathbf{w}_{\text{FP}}$ where $(\mathbf{w}_{\text{FP}})_I = \mathbf{w}_I + \sigma \mathbf{y}$ and $(\mathbf{w}_{\text{FP}})_{-I} = \mathbf{w}_{-I}$ to $\mathcal{A}$.

When $O' = \text{Null}$, we have $\mathbf{y} \leftarrow D_{\mathbb{R}^\ell, 1}$, and hence $\sigma \mathbf{y} \leftarrow D_{\mathbb{R}^\ell, \sigma}$. The definition of $(\text{SampleFirst}, \text{SampleRest})$ (Definition 5.1) tells us that on each query (D, spec) , the simulated oracle exactly gives an output from $\text{Train}^D(1^\lambda, \text{spec})$. On the other hand, when $O' = \text{GP}(\mathbf{S}, \cdot, \cdot)$, $\mathbf{y}$ is distributed as

$$\mathbf{y} \leftarrow \text{GP} \left(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma \sigma} \mathbf{w}_I, \mathbf{z} \right).$$

For our setting of γ , we have

$$\frac{\hat{\gamma}}{\sqrt{2}} \geq \frac{\gamma}{\sqrt{2}} \geq \sqrt{\lambda} \geq \eta_\epsilon(\mathbb{Z}),$$

where $\epsilon = 2^{-\lambda}$ by Lemma 3.1. By Lemma 5.3, on each query (D, spec) , the response $\mathbf{w}_{\text{FP}}$ generated by $\mathcal{B}$ is within $8 \cdot 2^{-\lambda} \ell$ statistical distance of the response $\text{FP}^D((\mathbf{S}, \mathbf{z}), \text{spec})$, which uses $\mathbf{g}_{\text{FP}}$ generated using Embed rather than $\sigma \mathbf{y}$. Since $\mathcal{A}$ makes at most $\text{poly}(\lambda)$ adaptive queries, a standard hybrid argument implies that in this case, $\mathcal{B}$'s distribution is within $2^{-\lambda} \text{poly}(\lambda) = \text{negl}(\lambda)$ statistical distance of $\mathcal{A}$'s distribution with the $\text{FP}((\mathbf{S}, \mathbf{z}), \cdot)$ oracle. Thus, $\mathcal{B}$ inherits the distinguishing advantage of $\mathcal{A}$ with a $\text{negl}(\lambda)$ loss, which is still inverse polynomial. This contradicts the assumed hardness of GP_{shift} . $\square$

Corollary 5.5.1 (Robust fingerprinting from CLWE). *Let $\lambda \in \mathbb{N}$ be the security parameter. Assume that $\text{CLWE}(\lambda, \beta/\sqrt{2}, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ is hard (Definition 3.7) for parameters $\gamma = \sqrt{3\lambda \log \lambda}$ and $\beta = 1/\text{poly}(\lambda)$ satisfying $\frac{\sqrt{5(\gamma^2 + \beta^2)}\beta}{\sqrt{2\pi}\gamma} = o(1) < \frac{\sqrt{6}}{12}$.*

Let $n = \text{poly}(\lambda)$ such that $n = \omega(\lambda)$ and $n - \lambda \geq \lambda$ be the number of noisy weights and let $\sigma = \sigma(\lambda)$ be the training algorithm's noise level. Let Train be any training algorithm in the class (σ, n) -NoisyTrain (Definition 5.1). Then there exists some

$$\Delta = \Omega \left(\frac{\sigma \sqrt{n}}{\lambda^{3/4} \log^{3/2} \lambda} \right)$$

such that there exists an $\text{L2}(\Delta)$ -robust (Definition 4.3) fingerprinting scheme with respect to Train.

Proof. We can set the number of secrets $\ell = \lambda$ and instantiate $(\text{Gen}, \text{FP}, \text{Ver}_\alpha)$ with $\alpha = \frac{\sqrt{6}}{12}$. Correctness, soundness, and undetectability follow from Theorem 5.5, together with the fact that $\frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5}{2\pi}} < \frac{\sqrt{6}}{12}$. For robustness, first see that Theorem 4.1 implies that there exists some $t = O(\sqrt{\lambda \log \lambda})$ such that $t\text{-PA}(n, \ell, \beta, \gamma)$ is hard. Combining this with the fact that $\frac{\gamma}{\hat{\gamma}^2} = \frac{\gamma}{\gamma^2 + o(1)} = \frac{1}{O(\sqrt{\lambda \log \lambda})}$ and applying Lemma 5.9 with $\alpha_2 = \frac{\sqrt{6}}{12} - \frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5}{2\pi}} = \alpha - o(1)$ gives the claimed robustness. $\square$

Corollary 5.5.2 (Robust fingerprinting from CLWE and Pancake Alignment). *Under the same setting as Corollary 5.5.1, additionally assume Assumption 4.3. Then there exists*

$$\Delta = \Omega\left(\frac{\sigma\sqrt{n}}{\sqrt{\lambda}\log\lambda}\right)$$

such that there exists an $\mathsf{L2}(\Delta)$ -robust fingerprinting scheme with respect to Train .

Proof. The proof is identical to that of Corollary 5.5.1, except that Assumption 4.3 guarantees the existence of some $t = \Theta(1)$ such that $t\text{-PA}(n, \ell, \beta, \gamma)$ is hard. $\square$

5.3.2 Proofs

For the remainder of this section, let, $n = \text{poly}(\lambda)$, $\ell = \text{poly}(\lambda)$ with $\ell < n$, $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$, and $\sigma = \sigma(\lambda)$ be functions of the security parameter. Let $\hat{\gamma} = \sqrt{\gamma^2 + \beta^2}$. When we refer to our construction, we mean instantiated with these parameters. Individual lemmas may impose additional constraints on these parameters; the parameters may otherwise be chosen arbitrarily subject to those constraints.

Correctness Correctness holds because when we scale the fingerprinted weights and look down the secret direction, we get a vector near $\mathbb{Z} + \mathbf{z}$.

Lemma 5.6 (Correctness). *Let the number of secrets satisfy $\ell = \omega(\log \lambda)$ and the threshold*

$$\alpha \geq \frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5}{2\pi}}.$$

Then the construction $(\mathsf{Gen}, \mathsf{FP}, \mathsf{Ver}_\alpha)$ satisfies correctness.

Proof. We prove something stronger, that correctness holds for *every* key $\mathbf{sk} \in \text{supp}(\mathsf{Gen}(1^\lambda))$. Let $\mathbf{sk} = (\mathbf{S}, \mathbf{z})$ and (D, spec) be fixed, and let $\mathbf{w}_{\text{FP}} \leftarrow \mathsf{FP}^D((\mathbf{S}, \mathbf{z}), \text{spec})$ where $(\mathbf{w}_{\text{FP}})_I = \mathbf{w}_I + \mathbf{g}_{\text{FP}}$ with $(\mathbf{w}_I, \cdot) \leftarrow \mathsf{SampleFirst}^D(1^\lambda, \text{spec})$ and $\mathbf{g}_{\text{FP}} \leftarrow \mathsf{Embed}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}_I, \mathbf{z}, \mathbf{g})$. By Lemma 5.1, we have that

$$\boldsymbol{\zeta} = \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w}_I + \mathbf{g}_{\text{FP}}) - \mathbf{z} \bmod 1 = \mathbf{e} \bmod 1, \quad \text{where } \mathbf{e} \leftarrow D_{\mathbb{R}^\ell, \frac{\hat{\gamma}\beta}{\gamma}}$$

We get that

$$\|\boldsymbol{\zeta}\|_2 = \|\mathbf{e} \bmod 1\|_2 \leq \|\mathbf{e}\|_2 \leq \frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5\ell}{2\pi}} \leq \alpha\sqrt{\ell}$$

with probability at least $1 - \exp(-\ell)$, by Lemma 3.4. $\square$

Soundness Soundness holds because a fixed vector independent of the uniformly random $\mathbf{z} \leftarrow [0, 1)^\ell$ is unlikely to be near $\mathbb{Z} + \mathbf{z}$. The following lemmas formalize this.

Lemma 5.7. *Let $\ell \in \mathbb{N}$ and let $\mathbf{z} \leftarrow [-1/2, 1/2)^\ell$ be uniformly random. Then*

$$\Pr \left[\|\mathbf{z}\|_2^2 \leq \frac{\ell}{24} \right] \leq 2 \exp(-\ell/18).$$

Proof. We know that $\|\mathbf{z}\|_2^2 = \sum_{i=1}^\ell z_i^2$, where $z_i \leftarrow [-1/2, 1/2)$ satisfies $0 \leq z_i^2 \leq 1/4$ and $\mathbb{E}[z_i^2] = 1/12$. By Hoeffding's inequality,

$$\Pr \left[\sum_{i=1}^\ell z_i^2 - \frac{\ell}{12} \leq -t \right] \leq 2 \exp \left(-\frac{2t^2}{\ell(1/4)^2} \right).$$

Taking $t = \ell/24$ we obtain

$$\Pr \left[\sum_{i=1}^\ell z_i^2 \leq \frac{\ell}{12} - \frac{\ell}{24} \right] \leq 2 \exp \left(-\frac{2(\ell/24)^2}{\ell/16} \right) = 2 \exp \left(-\frac{\ell}{18} \right)$$

$\square$

Lemma 5.8 (Soundness). *Let the number of secrets $\ell = \omega(\log \lambda)$ and let the threshold satisfy $\alpha \in (0, \frac{\sqrt{6}}{12}]$. Then the construction $(\text{Gen}, \text{FP}, \text{Ver}_\alpha)$ satisfies soundness.*

Proof. We show that for any $\mathbf{S} \in V_{\ell, n}$, soundness holds with overwhelming probability over $\mathbf{z} \leftarrow [0, 1)^\ell$. Let $\mathbf{w}' \in \mathbb{R}^n$ be fixed. Then

$$\zeta = (\mathbf{z} - \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top \mathbf{w}') \text{ cmod } 1 \stackrel{d}{=} U([-1/2, 1/2)^\ell),$$

since subtracting a fixed vector and reducing modulo 1 preserves the uniform distribution. Lemma 5.7 tells us that this norm is more than $\sqrt{\frac{\ell}{24}} = \frac{\sqrt{6}}{12} \sqrt{\ell} \geq \alpha \sqrt{\ell}$ with probability at least $1 - \exp(-\ell/18)$, which means the verifier rejects with this probability. $\square$

Robustness In order to remove the fingerprint by adding a small perturbation $\mathbf{v} \in \mathbb{R}^n$ to fingerprinted weights, the adversary must find $\mathbf{v} \in \mathbb{R}^n$ of small norm that is sufficiently aligned with the secret directions, so that the verification test $\left\| \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top ((\mathbf{w}_{\text{FP}})_I + \mathbf{v}) - \mathbf{z} \text{ cmod } 1 \right\|_2$ exceeds the threshold $\alpha \sqrt{\ell}$. The pancake alignment assumption asserts that this problem is computationally hard.

Lemma 5.9 (Robustness). *Assume that for some parameters $t = t(\lambda)$, $\gamma = \gamma(\lambda) \geq \sqrt{3\lambda\omega(\log \lambda)}$, and $\beta = \beta(\lambda)$ such that $\frac{\sqrt{5(\gamma^2 + \beta^2)}\beta}{\sqrt{2\pi}\gamma} < \frac{1}{2}$, no PPT adversary can solve the t -PA(n, ℓ, β, γ) problem with advantage $\epsilon = \epsilon(\lambda)$ (Definition 4.9). For any threshold $\alpha = \alpha(\lambda)$ satisfying*

$$\alpha = \alpha_1 + \alpha_2, \quad \text{for some } \alpha_1 \geq \frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5}{2\pi}}, \alpha_2 \in (0, \frac{1}{2} - \alpha_1).$$

and any $\Delta = \Delta(\lambda)$ satisfying

$$\Delta \leq \frac{\gamma}{\hat{\gamma}^2} \cdot \frac{\alpha_2 \sigma \sqrt{n}}{\sqrt{t}},$$

the construction $(\text{Gen}, \text{FP}, \text{Ver}_\alpha)$ is $(\text{L2}(\Delta), \delta)$ -robust (Definition 4.3) against adversaries making $m = \text{poly}(\lambda)$ queries to their fingerprinting oracle, where $\delta := m(8\ell 2^{-\lambda} + \exp(-\ell) + \epsilon)$.

Proof. For contradiction, say there exists a PPT adversary $\mathcal{A}$ making m queries such that with probability more than $8m\ell 2^{-\lambda} + m \exp(-\ell) + m\epsilon$, $\mathbf{w}' \leftarrow \mathcal{A}^{\text{O}_{\text{FP}}(1^\lambda, (\mathbf{S}, \mathbf{z}), \cdot, \cdot)}(1^\lambda)$ is rejected despite being close to one of the observed weights. That is, there exists some $\mathbf{w}_{\text{FP}} \in \mathcal{W}$ such that

$$\|\zeta\|_2 = \left\| \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top ((\mathbf{w}_{\text{FP}})_I + \mathbf{v}) - \mathbf{z} \text{ cmod } 1 \right\|_2 > \alpha \sqrt{\ell}$$

where $\mathbf{v} := \mathbf{w}'_I - (\mathbf{w}_{\text{FP}})_I$ is the adversary's perturbation to the coordinates at I , which satisfies $\|\mathbf{v}\|_2 \leq \Delta$. Recall from Lemma 5.1, this implies

$$\left\| \mathbf{e} + \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top \mathbf{v} \text{ cmod } 1 \right\|_2 > \alpha \sqrt{\ell}, \quad \text{where } \mathbf{e} \leftarrow D_{\mathbb{R}^\ell, \frac{\hat{\gamma}}{\gamma}\beta}.$$

We swap to a hybrid where the game ends with $\mathcal{A}$ losing if for any $\mathbf{e}$, $\|\mathbf{e}\|_2 > \frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5\ell}{2\pi}} \geq \alpha_1 \sqrt{\ell}$. From Lemma 3.4 and a union bound, this occurs with probability at most $m \exp(-\ell)$, so $\mathcal{A}$'s winning probability in this game is more than $8m\ell 2^{-\lambda} + m\epsilon$.

Now, we show if $\mathcal{A}$ wins in this hybrid, there is an adversary $\mathcal{B}$ that breaks alignment. $\mathcal{B}^{\text{GP}(\mathbf{S}, \cdot, \cdot)}$ samples a random $\mathbf{z} \leftarrow [0, 1)^\ell$. $\mathcal{B}$ simulates fingerprinting queries (D, spec) to $\mathcal{A}$ by computing $(\mathbf{w}_I, \text{state}) \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$, sampling $\mathbf{y} \leftarrow \text{GP}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}_I, \mathbf{z})$ with its Gaussian pancake oracle, and sampling $\mathbf{w}_{-I} \leftarrow \text{SampleRest}^D(1^\lambda, \text{spec}, \text{state}, \sigma\mathbf{y})$. It outputs $\mathbf{w}$ with the noise $\sigma\mathbf{y}$ added to the indices at I . $\mathcal{B}$ collects these outputs in a set $\mathcal{W}$ and finishes executing $\mathcal{A}$ to get $\mathbf{w}'$. Its final output is $\frac{\mathbf{w}'_I - (\mathbf{w}_{\text{FP}}^*)_I}{\|\mathbf{w}'_I - (\mathbf{w}_{\text{FP}}^*)_I\|_2}$ for a uniformly random $\mathbf{w}_{\text{FP}}^* \in \mathcal{W}$.

For our choice of γ , by Lemma 3.1, $\frac{\hat{\gamma}}{\sqrt{2}} \geq \sqrt{\lambda} \geq \eta_{2-\lambda}(\mathbb{Z})$. By Lemma 5.3 and the triangle inequality, the output distribution of $\mathcal{B}$ is within $8m\ell 2^{-\lambda}$ statistical distance of the output of $\mathcal{A}$. Therefore, with probability more than $m\epsilon$, there is some $\mathbf{w}_{\text{FP}} \in \mathcal{W}$ and $\mathbf{e} \in \mathbb{R}^\ell$ such that

$$\left\| \mathbf{e} + \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w}'_I - (\mathbf{w}_{\text{FP}})_I) \text{ cmod } 1 \right\|_2 > \alpha\sqrt{\ell}, \quad \text{where } \|\mathbf{w}'_I - (\mathbf{w}_{\text{FP}})_I\|_2 \leq \Delta \text{ and } \|\mathbf{e}\|_2 \leq \alpha_1\sqrt{\ell}.$$

Since center modding only decreases the norm, triangle inequality tells us

$$\left\| \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w}'_I - (\mathbf{w}_{\text{FP}})_I) \right\|_2 > \alpha\sqrt{\ell} - \|\mathbf{e}\|_2 \geq (\alpha - \alpha_1)\sqrt{\ell} \geq \alpha_2\sqrt{\ell}.$$

Combining this with the fact that $\|\mathbf{w}'_I - (\mathbf{w}_{\text{FP}})_I\|_2^2 \leq \Delta^2 \leq \frac{\gamma^2}{\hat{\gamma}^4} \cdot \frac{\alpha_2^2 \sigma^2 n}{t}$, we see

$$\frac{\|\mathbf{S}^\top (\mathbf{w}'_I - (\mathbf{w}_{\text{FP}})_I)\|_2^2}{\|\mathbf{w}'_I - (\mathbf{w}_{\text{FP}})_I\|_2^2} \geq \frac{1}{\Delta^2} \cdot \frac{\sigma^2 \gamma^2 \alpha_2^2 \ell}{\hat{\gamma}^4} \geq t \frac{\ell}{n}.$$

Therefore, $\mathcal{B}$ succeeds on the pancake alignment problem when such a $\mathbf{w}_{\text{FP}}$ exists (which we argued happens with more than $m\epsilon$ probability) and it correctly guesses $\mathbf{w}_{\text{FP}}^* = \mathbf{w}_{\text{FP}}$. Since $\mathcal{A}$ makes m queries, $\mathcal{B}$ obtains advantage at least $\frac{m\epsilon}{m} = \epsilon$, contradicting the assumption. $\square$

5.4 Discussion

Ultimately, an adversary must introduce a perturbation $\mathbf{v}$ satisfying

$$\|\mathbf{v}\|_2 = \tilde{\Theta}\left(\frac{\sigma\sqrt{n}}{\lambda^{3/4}}\right)$$

in order to remove the fingerprint. At first glance, this requirement may not seem particularly restrictive. The fingerprinter is already willing to release models that are indistinguishable from the outputs of Train , which adds noise of norm roughly $\sigma\sqrt{n}$ to the base model $\mathbf{w}_I$. Thus, adding noise of a smaller magnitude $\tilde{\Theta}(\sigma\sqrt{n}/\lambda^{3/4})$ to the already-noised model could plausibly preserve the model's quality while still being sufficient to remove the fingerprint. We therefore do not claim that removing the fingerprint necessarily degrades the model. However, we argue that our guarantee captures many transformations that an honest customer would apply.

In more detail, let B denote the root-mean-square magnitude of the base weights, $B^2 := \frac{1}{n} \sum_{i=1}^n (w_I)_i^2$, so that $\|\mathbf{w}_I\|_2 = B\sqrt{n}$. One can intuitively view B as a measure of the typical absolute value of a single weight. Write the training noise as $\sigma = \varepsilon B$ for some $\varepsilon = o(1)$; that is, the magnitude of the added noise $\|\mathbf{g}\|_2$ is of slightly smaller order than that of the base weights, $\sigma\sqrt{n} = o(\|\mathbf{w}_I\|_2)$. If a user modifies *every* weight, the root-mean-square magnitude of their perturbation must be at least

$$\Delta_n := \frac{\|\mathbf{v}\|_2}{\sqrt{n}} = \tilde{\Theta}\left(\frac{\varepsilon B}{\lambda^{3/4}}\right),$$

in order to remove the fingerprint, which is of slightly smaller order than the root-mean-square of the weights themselves. If instead the user modifies only $k \ll n$ weights, the same budget must be concentrated on those coordinates, so the root-mean-square magnitude of the perturbation must be at least

$$\Delta_k := \frac{\|\mathbf{v}\|_2}{\sqrt{k}} = \tilde{\Theta}\left(\frac{\varepsilon B}{\lambda^{3/4}} \sqrt{\frac{n}{k}}\right),$$

which exceeds the root-mean-square magnitude B of the weights whenever $k = \tilde{O}(\varepsilon^2 n / \lambda^{3/2})$ is a small enough fraction of the weights. In this regime, removing the fingerprint would require modifying the affected weights by more than their typical value. Our robustness guarantee therefore covers transformations such as the following.

- **Quantization:** A model provider may round weights to a lower precision (e.g., to reduce storage requirements [LGH24]). If rounding changes each coordinate by at most Δ_n in absolute distance, for instance by rounding them to the nearest value in $\Delta_n\mathbb{Z}$, then the fingerprint is preserved.
- **Zeroing out small weights:** A model provider may sparsify the model by zeroing some weights [HPTD15]. If k weights of absolute value at most Δ_k are zeroed out, then the fingerprint remains.
- **Adding random noise:** A model provider may add more Gaussian noise to the weights [AJVZ26]. For $\mathbf{v} \leftarrow D_{\mathbb{R}^n, \sigma/\lambda^{3/4}}$ the norm is concentrated around $\sigma\sqrt{n}/\lambda^{3/4}$. Hence, adding slightly less noise than the training algorithm’s noise preserves the fingerprint.

Moreover, many practical model modifications only alter a low-dimensional subspace of the weight vector. Indeed, in our setting, where a customer outsources training to a powerful model provider, we would expect the customer to have substantially less compute, and so additional training should not be on the entire model.

- **Fine-tuning a subset of weights:** Many fine-tuning procedures update only a subset of $k \ll n$ weights, leaving the rest unchanged. If fine-tuning introduces a perturbation with root-mean-square magnitude at most Δ_k to the updated weights, then the fingerprint is preserved.
- **LoRA:** Low-rank adaptation (LoRA) [HSA+21] freezes weights and learns a low-rank matrix, $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ of rank $k \ll n$, which is flattened and added to the weights. Removing the fingerprint requires $\|\mathbf{M}\|_F = \Theta(\sigma\sqrt{n}/\lambda^{3/4})$. Since $\mathbf{M}$ has at most k nonzero singular values, the fingerprint is preserved whenever the root-mean-square singular value of the update is at most Δ_k .

6 Unforgeable Fingerprinting

We adapt our construction to achieve unforgeability by leveraging a certain ℓ_2 locality-sensitive hash, which is a seeded hash function h_s that allows one to approximate the ℓ_2 distance between two inputs $\mathbf{x}, \mathbf{y}$ by approximating the ℓ_2 distance between their hashes $h_s(\mathbf{x}), h_s(\mathbf{y})$, provided that the inputs are chosen *without* knowledge of the hash seed s . For our locality-sensitive hash, we use a binary Johnson–Lindenstrauss projection [JL+84; Ach03]. Specifically, the seed is a random matrix $\mathbf{P} \leftarrow \{\pm 1\}^{m \times n}$, and the hash is the scaled projection $h_{\mathbf{P}}(\mathbf{x}) = \frac{1}{\sqrt{m}}\mathbf{P}\mathbf{x}$.

To learn fingerprinted weights, rather than embedding a random $\mathbf{z} \in \{0, 1\}^\ell$ as we do in the robust construction, we instead embed both a hash seed s and the hash $h_s(\mathbf{w} + \mathbf{g})$ of the trained weights into the fingerprinted weights $\mathbf{w} + \mathbf{g}_{\text{FP}}$ ¹². We do so in a manner satisfying the following properties:

1. An adversary that observes the fingerprinted weights $\mathbf{w} + \mathbf{g}_{\text{FP}}$ cannot learn the hash seed s and therefore cannot adaptively use the seed to circumvent the closeness check. We achieve this by embedding an *encryption* of the seed and hash. Since we can only *approximately* recover the ciphertext from the embedding, we additionally encode the ciphertext using an error-correcting code.
2. When decoding a message from candidate weights $\mathbf{w}'$, if $\mathbf{w}'$ is ℓ_2 close to previously fingerprinted weights, we recover the corresponding hash and seed. Otherwise, we either recover $\perp$ or one of the previously embedded hash and seed pairs. We achieve this by also embedding the randomness used in the encryption, as well as a MAC of this randomness to authenticate it, so that during verification we only proceed if the ciphertext corresponds to a valid encryption. This idea was inspired by the non-malleable transformation of [AACD+25].

To verify a candidate $\mathbf{w}'$, the verifier attempts to decode a hash and its seed from $\mathbf{w}'$ and check if $h_s(\mathbf{w}')$ is close to decoded hash.

¹²We require a hash because we need to reduce the message size sufficiently to embed it into $\mathbf{g}_{\text{FP}}$.

6.1 Unforgeable fingerprinting construction

For this section, let $n = n(\lambda)$ and $\sigma = \sigma(\lambda)$ be parameters and let **Train** be a training algorithm in the class (σ, n) -NoisyTrain. Denote $(\text{SampleFirst}, \text{SampleRest})$ its decomposition, and I the set of noisy coordinates, as in Definition 5.1. Let $\gamma = \gamma(\lambda), \beta = \beta(\lambda)$ be pancake parameters. Let $\eta = \eta(\lambda)$ be a precision parameter for rounding, and let $B = B(\lambda)$ be a bound on the magnitude of the weights; that is, for every (D, spec) , the weights $\mathbf{w}_I \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$ satisfy $\|\mathbf{w}_I\|_\infty \leq B$.

Let $\Sigma = (\text{Gen}, \text{Enc}, \text{Dec})$ be a symmetric key encryption scheme (Definition 3.3) that requires $s = \text{poly}(\lambda)$ bits of randomness per encryption, with message length $q_{\text{in}} = \text{poly}(\lambda)$ (to be specified later) and ciphertext length $q_{\text{out}} = q_{\text{in}} + O(\lambda)$. For a particular $m = \text{poly}(\lambda)$ to be specified later, let

$$\{\text{PRF}_k : \{0, 1\}^\lambda \rightarrow \{\pm 1\}^{m \times n} \times \{0, 1\}^s\} \quad \text{and} \quad \{\text{MAC}_k : \{0, 1\}^{q_{\text{in}} - 2\lambda} \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda\}$$

be pseudorandom function families (Definition 3.2). Let $(\text{Encode}, \text{Decode})$ be an information-theoretic error-correcting code with constant rate $\rho \in (0, 1)$ that can correct up to a constant $\epsilon \in (0, \frac{1}{4})$ fraction of Hamming errors¹³.

Assume n is large enough so that we can choose the number of secret directions $\ell = \omega(\log \lambda)$ to satisfy $\frac{q_{\text{out}}}{\rho} \leq \ell < n$. That is, using the error-correcting code, we can encode ciphertexts that encrypt q_{in} bit messages into ℓ bit codewords. We detail our construction $\text{UF} = (\text{Gen}, \text{FP}, \text{Ver})$ below.

Our key generation algorithm $\text{UF.Gen}(1^\lambda)$ does as follows

- Sample $\mathbf{S} \leftarrow V_{\ell, n}$.
- Sample PRF keys $\mathbf{k} \leftarrow \{0, 1\}^\lambda, \mathbf{k}_{\text{MAC}} \leftarrow \{0, 1\}^\lambda$.
- Sample an encryption key $\text{sk} \leftarrow \Sigma.\text{Gen}(1^\lambda)$.
- **return** $(\mathbf{S}, \mathbf{k}, \mathbf{k}_{\text{MAC}}, \text{sk})$.

The fingerprinting algorithm $\text{UF.FP}^D((\mathbf{S}, \mathbf{k}, \mathbf{k}_{\text{MAC}}, \text{sk}), \text{spec})$ does

- Sample $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$ and compute $(\mathbf{w}_I, \text{state}) \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$. If $\|\mathbf{g}\|_\infty > \sigma\sqrt{\lambda}$, **return** $\perp$.
- Sample a seed $\mathbf{r} \leftarrow \{0, 1\}^\lambda$ and let $(\mathbf{P}, \mathbf{t}^{(1)}) = \text{PRF}_k(\mathbf{r}) \in \{\pm 1\}^{m \times n} \times \{0, 1\}^s$.
- Let the message $\mathbf{z} = \lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w}_I + \mathbf{g}) \rfloor_\eta$.
- Authenticate the message and randomness $\mathbf{t}^{(2)} = \text{MAC}_{\mathbf{k}_{\text{MAC}}}(\mathbf{z}, \mathbf{r}) \in \{0, 1\}^\lambda$.
- Generate a ciphertext using $\mathbf{t}^{(1)}$ as the randomness,

$$\mathbf{ct} = \Sigma.\text{Enc}(\text{sk}, (\mathbf{z}, \mathbf{r}, \mathbf{t}^{(2)}); \mathbf{t}^{(1)}).$$

- Encode the ciphertext and map it to $\{\frac{1}{4}, \frac{3}{4}\}^\ell$,

$$\mathbf{c}' = \frac{3}{4} - \frac{1}{2}\mathbf{c}, \quad \text{where } \mathbf{c} = \text{Encode}(\mathbf{ct}) \in \{0, 1\}^\ell.$$

- Sample $\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}(\mathbf{S}, \frac{\gamma^2}{\gamma\sigma}\mathbf{w}_I, \mathbf{c}', \mathbf{g})$.
- Sample $\mathbf{w}_{-I} \leftarrow \text{SampleRest}^D(1^\lambda, \text{spec}, \text{state}, \mathbf{g}_{\text{FP}})$.
- Let $\mathbf{w}_{\text{FP}}$ be the vector such that $(\mathbf{w}_{\text{FP}})_I = \mathbf{w}_I + \mathbf{g}_{\text{FP}}$ and $(\mathbf{w}_{\text{FP}})_{-I} = \mathbf{w}_{-I}$.
- **return** $\mathbf{w}_{\text{FP}}$.

The verification algorithm $\text{UF.Ver}_\tau((\mathbf{S}, \mathbf{k}, \mathbf{k}_{\text{MAC}}, \text{sk}), \mathbf{w}')$, parameterized by a threshold $\tau = \tau(\lambda)$ does

¹³Such codes exist, for example [ABNN+92; NN90; Ta-17].

- Compute $\zeta = \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top \mathbf{w}' \bmod 1 \in [-1/2, 1/2)^\ell$ and let

$$\tilde{\mathbf{c}} = \begin{cases} 0 & \text{if } \zeta_i < 0 \\ 1 & \text{otherwise} \end{cases}.$$

- Compute $(\tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{t}}^{(2)}) \leftarrow \Sigma.\text{Dec}(\text{sk}, \text{Decode}(\tilde{\mathbf{c}}))$.
- Compute $(\tilde{\mathbf{P}}, \tilde{\mathbf{t}}^{(1)}) = \text{PRF}_k(\tilde{\mathbf{r}})$ and $\mathbf{c} = \text{Encode}(\Sigma.\text{Enc}(\text{sk}, (\tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{t}}^{(2)}); \tilde{\mathbf{t}}^{(1)}))$. Check if

$$\text{MAC}_{k_{\text{MAC}}}(\tilde{\mathbf{z}}, \tilde{\mathbf{r}}) \neq \tilde{\mathbf{t}}^{(2)} \text{ or } \|\tilde{\mathbf{c}} - \mathbf{c}\|_0 > \epsilon\ell,$$

if either occurs, **return 0**.

- **return 1** if $\left\| \tilde{\mathbf{z}} - \frac{1}{\sqrt{m}} \tilde{\mathbf{P}} \mathbf{w}' \right\|_2 \leq \tau$, 0 otherwise.

Using this construction, we can obtain the following guarantees.

Theorem 6.1 (Robust and unforgeable fingerprinting). *Let $\lambda \in \mathbb{N}$ be a (sufficiently large) security parameter. Assume all of the following:*

- $\text{CLWE}(\lambda, \beta/\sqrt{2}, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ is hard (Definition 3.7) for $\gamma = \sqrt{3\lambda} \log \lambda$ and $\beta = 1/\text{poly}(\lambda)$ small enough so that $\frac{\sqrt{\gamma^2 + \beta^2} \beta}{\gamma} \sqrt{\frac{5}{2\pi}} = o(1)$.
- For any $q_{\text{in}} = \text{poly}(\lambda)$ there is an IND-CPA secure encryption scheme (Definition 3.3) with message length q_{in} and ciphertext length $q_{\text{out}} = q_{\text{in}} + O(\lambda)$,
- For any input and output lengths $p_{\text{in}} = \text{poly}(\lambda)$ and $p_{\text{out}} = \text{poly}(\lambda)$, there exists a PRF family (Definition 3.2).

Let $n = \text{poly}(\lambda)$ such that $n \geq \omega(\lambda \log \lambda) + \lambda$ be the number of noisy weights, and let $\sigma = \sigma(\lambda)$ such that σ and $1/\sigma$ are polynomially bounded be the noise level of the training algorithm. Let Train be any training algorithm in the class (σ, n) -NoisyTrain (Definition 5.1), that outputs weights with infinity norm bounded by $B = \text{poly}(\lambda)$. Then there exists

$$\Delta^* = \Theta\left(\frac{\sigma\sqrt{n}}{\lambda^{3/4} \log^{3/2} \lambda}\right)$$

such that for any $\Delta_{\text{RB}} \leq \Delta^*$, there exists an $\text{L2}(\Delta_{\text{RB}})$ -robust (Definition 4.3) and $\text{L2}(\Delta_{\text{UF}})$ -unforgeable (Definition 4.4) fingerprinting scheme, for a particular Δ_{UF} satisfying $\Delta_{\text{RB}} \leq \Delta_{\text{UF}} \leq O(\Delta_{\text{RB}} + \sigma\sqrt{\lambda \log \lambda})$.

Proof. We instantiate our construction UF with the following parameters. Let $\delta_{\text{JL}} \in (0, 1)$ and $\epsilon \in (0, 1/4)$ be any constants. We set the rounding parameter $\eta = \sigma$ and the hashed dimension $m = \frac{4+2\lambda}{\delta_{\text{JL}}^2/2 - \delta_{\text{JL}}^3/3} = \Theta(\lambda)$. Let Σ be an IND-CPA encryption scheme that requires $s = \text{poly}(\lambda)$ bits of randomness per encryption and has message size $q_{\text{in}} = \text{poly}(\lambda)$ defined as

$$q_{\text{in}} = m \log\left(\frac{2n(B + \sigma\sqrt{\lambda})}{\sigma\sqrt{m}} + 1\right) + 2\lambda = O(\lambda \log \lambda),$$

and ciphertext size $q_{\text{out}} = q_{\text{in}} + O(\lambda)$. Let PRF be a PRF family with input size λ , output size $m\eta + s$, and let MAC be a PRF family with input size $q_{\text{in}} - \lambda$, output size λ . Let the number of secrets $\ell = \left\lceil \frac{q_{\text{out}}}{\rho} \right\rceil$, where $\rho \in (0, 1)$ is the constant rate of the error-correcting code corresponding to error rate ϵ . We have $\lambda \leq \ell \leq \left\lceil \frac{q_{\text{in}} + O(\lambda)}{\rho} \right\rceil \leq O(\lambda \log \lambda)$, which implies $\sqrt{\frac{\gamma^2 + \beta^2}{2}} \geq \frac{\gamma}{\sqrt{2}} \geq \sqrt{\lambda \omega(\log \lambda)} \geq \sqrt{\ln \frac{6\ell}{\pi}}$ for λ sufficiently large.

Syntax Since $n \geq \omega(\lambda \log \lambda) + \lambda$, $n - \ell \geq \lambda$ and $n > m$ for sufficiently large λ . Moreover, see that with Σ , we encrypt messages $\lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_\sigma$, where $\mathbf{P} \in \{\pm 1\}^{m \times n}$, $\|\mathbf{w}\|_\infty \leq B$, and $\|\mathbf{g}\|_\infty \leq \sigma\sqrt{\lambda}$, along with two λ -bit strings. Since $\|\frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g})\|_\infty \leq n(B + \sigma\sqrt{\lambda})/\sqrt{m}$, after rounding to $\sigma\mathbb{Z}$, the message can be represented using q_{in} bits, and the resulting ciphertext has length q_{out} . The error-correcting code therefore produces a codeword $\mathbf{c}$ of length $\frac{q_{\text{out}}}{\rho} \leq \ell$.

Undetectability Undetectability follows from the same argument as Theorem 5.5, except that the Gaussian pancake adversary must sample its own decryption key $\mathbf{sk}$ and PRF keys $\mathbf{k}, \mathbf{k}_{\text{MAC}}$ at the beginning and compute the message $\mathbf{c}'$ for each FP query in order to sample $\mathbf{y} \leftarrow O'(\frac{\gamma^2}{\gamma\sigma} \mathbf{w}_I, \mathbf{c}')$. Note that the abort condition $\|\mathbf{g}\|_\infty > \sigma\sqrt{\lambda}$ changes the distinguishing advantage by at most a negligible amount by Lemma 3.4.

Robustness By Claims 6.1 through 6.4, the robustness security game is $\text{negl}(\lambda)$ -indistinguishable from G_4^{RB} . By Theorem 4.1 and the fact that $n - \ell \geq \lambda$ and $\ell = o(n)$, the CLWE assumption implies that there exists some $t = O(\sqrt{\lambda} \log \lambda)$ such that $t\text{-PA}(n, \ell, \beta, \gamma)$ is hard. Combining this with the fact that $\frac{\gamma}{\gamma^2} = \frac{1}{O(\sqrt{\lambda} \log \lambda)}$, then if we choose robustness parameter

$$\Delta_{\text{RB}} = \frac{\gamma}{\gamma^2} \cdot \frac{\sigma(\frac{\sqrt{\epsilon}}{4} - o(1))\sqrt{n}}{\sqrt{t}} = \Theta\left(\frac{\sigma\sqrt{n}}{\lambda^{3/4} \log^{3/2} \lambda}\right)$$

and threshold $\tau = \sqrt{1 + \delta_{\text{JL}}}(\Delta_{\text{RB}} + \sigma\sqrt{C\ell}) + \sigma\sqrt{m}$, then by Lemma 6.2 no PPT adversary can win the robustness game with inverse-polynomial probability.

Unforgeability From Claims 6.1 to 6.4, the unforgeability security game is $\text{negl}(\lambda)$ -indistinguishable from G_4^{UF} . By Lemma 6.4, if we use the same threshold τ as we used for robustness, then we achieve the claimed unforgeability. $\square$

Corollary 6.1.1. *Under the same setting as Theorem 6.1, additionally assume Assumption 4.3. Then there exists*

$$\Delta^* = \Theta\left(\frac{\sigma\sqrt{n}}{\sqrt{\lambda} \log \lambda}\right)$$

such that for any $\Delta_{\text{RB}} \leq \Delta^*$, there exists an $\text{L2}(\Delta_{\text{RB}})$ -robust and $\text{L2}(\Delta_{\text{UF}})$ -unforgeable fingerprinting scheme, for a particular Δ_{UF} satisfying $\Delta_{\text{RB}} \leq \Delta_{\text{UF}} \leq O(\Delta_{\text{RB}} + \sigma\sqrt{\lambda} \log \lambda)$.

6.2 Hybrids

For ease of notation, we prove security in the case where Train outputs weights in $\mathbb{R}^n$, so that the SampleRest algorithm is trivial. The same arguments extend straightforwardly to the general case. We give a number of hybrids, for both the robustness and unforgeability games. We mark changes between hybrids in **red**. Let Ver^{f_1, f_2} denote our verification algorithms using the functions f_1, f_2 instead of $\text{PRF}_{\mathbf{k}}, \text{MAC}_{\mathbf{k}_{\text{MAC}}}$ respectively. We define the following hybrids.

- G_0^{RB} and G_0^{UF} , defined below, are identical to the robustness and unforgeability security games for the predicates $\text{L2}(\Delta_{\text{RB}})$ and $\text{L2}(\Delta_{\text{UF}})$ respectively (Definitions 4.3 and 4.4), except that $\mathcal{A}$ is given access to the fingerprinting oracle FP^{f_1, f_2} (defined below), and the final verification check is performed using $\text{Ver}_\tau^{f_1, f_2}$, where f_1 and f_2 are uniformly random functions. In addition to replacing the PRFs with random functions, the fingerprinting algorithm FP^{f_1, f_2} differs from the real fingerprinting algorithm in two ways: it aborts if any of the randomly sampled seeds $\mathbf{r}$ collide, and it maintains a set Π of certain message-codeword pairs.

$G_0^{\text{RB}}(\mathcal{A}, \lambda, \tau, \Delta_{\text{RB}})$

- 1: Π , $\Pi_{\text{FP}} = \emptyset$.
- 2: Sample $\mathbf{S} \leftarrow V_{\ell, n}$, $\text{sk} \leftarrow \Sigma.\text{Gen}(1^\lambda)$, and uniformly random functions
$$f_1 : \{0, 1\}^\lambda \rightarrow \{\pm 1\}^{m \times n} \times \{0, 1\}^s, \quad f_2 : \{0, 1\}^{q_{\text{in}} - 2\lambda} \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda.$$
- 3: $\mathbf{w}' \leftarrow \mathcal{A}_{\text{FP}^{f_1, f_2}}^{\mathcal{O}}((\mathbf{S}, \text{sk}), \cdot, \cdot)(1^\lambda)$.
- 4: Output 1 if both:
 - $\text{Ver}_{\tau}^{f_1, f_2}(\mathbf{S}, \text{sk}, \mathbf{w}') = 0$.
 - $\exists \mathbf{w} \in \Pi_{\text{FP}}$ such that $\|\mathbf{w} - \mathbf{w}'\|_2 \leq \Delta_{\text{RB}}$.

$G_0^{\text{UF}}(\mathcal{A}, \lambda, \tau, \Delta_{\text{UF}})$

- 1: Π , $\Pi_{\text{FP}} = \emptyset$.
- 2: Sample $\mathbf{S} \leftarrow V_{\ell, n}$, $\text{sk} \leftarrow \Sigma.\text{Gen}(1^\lambda)$, and uniformly random functions
$$f_1 : \{0, 1\}^\lambda \rightarrow \{\pm 1\}^{m \times n} \times \{0, 1\}^s, \quad f_2 : \{0, 1\}^{q_{\text{in}} - 2\lambda} \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda.$$
- 3: $\mathbf{w}' \leftarrow \mathcal{A}_{\text{FP}^{f_1, f_2}}^{\mathcal{O}}((\mathbf{S}, \text{sk}), \cdot, \cdot)(1^\lambda)$.
- 4: Output 1 if both:
 - $\text{Ver}_{\tau}^{f_1, f_2}(\mathbf{S}, \text{sk}, \mathbf{w}') = 1$.
 - $\forall \mathbf{w} \in \Pi_{\text{FP}}, \|\mathbf{w} - \mathbf{w}'\|_2 > \Delta_{\text{UF}}$.

$\text{FP}^{f_1, f_2}((\mathbf{S}, \text{sk}), (\mathcal{T}, \text{spec}))$

- 1: Sample $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$ and compute $(\mathbf{w}, \text{state}) \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$. If $\|\mathbf{g}\|_\infty > \sigma\sqrt{\lambda}$, **return** $\perp$.
- 2: Sample $\mathbf{r} \leftarrow \{0, 1\}^\lambda$. **If $\mathbf{r}$ is equal to a previously sampled seed, abort and output 0.**
- 3: Let $(\mathbf{P}, \mathbf{t}^{(1)}) = f_1(\mathbf{r})$, let $\mathbf{z} = \lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_\eta$, and let $\mathbf{t}^{(2)} = f_2(\mathbf{z}, \mathbf{r})$.
- 4: Compute $\mathbf{ct} = \Sigma.\text{Enc}(\text{sk}, (\mathbf{z}, \mathbf{r}, \mathbf{t}^{(2)}); \mathbf{t}^{(1)})$ and $\mathbf{c} = \text{Encode}(\mathbf{ct}) \in \{0, 1\}^\ell$.
- 5: **Add $((\mathbf{z}, \mathbf{P}), \mathbf{c})$ to Π .**
- 6: For $i \in [\ell]$, let $c'_i = \frac{3}{4} - \frac{c_i}{2}$.
- 7: Sample $\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}, \mathbf{c}', \mathbf{g})$.
- 8: Add $\mathbf{w} + \mathbf{g}_{\text{FP}}$ to Π_{FP} .
- 9: **return** $\mathbf{w} + \mathbf{g}_{\text{FP}}$.

- G_1^{RB} and G_1^{UF} are identical to G_0^{RB} and G_0^{UF} , respectively, except that the final verification check is performed using $\text{Ver}_{\tau, \text{no-decrypt}}$ instead of $\text{Ver}_{\tau}^{f_1, f_2}$. The algorithm $\text{Ver}_{\tau, \text{no-decrypt}}$, defined below, uses the set Π to answer verification queries without needing the decryption key sk .

$\text{Ver}_{\tau, \text{no-decrypt}}(\mathbf{S}, \mathbf{w}')$

- 1: Compute $\zeta = \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top \mathbf{w}' \text{ cmod } 1 \in [-1/2, 1/2)^\ell$.
- 2: $\tilde{\mathbf{c}} = \begin{cases} 0 & \text{if } \zeta_i < 0 \\ 1 & \text{otherwise} \end{cases}$.
- 3: Compute $(\tilde{\mathbf{z}}, \tilde{\mathbf{P}}) = \begin{cases} (\mathbf{z}, \mathbf{P}) & \text{if there exists } \mathbf{c} \text{ such that } ((\mathbf{z}, \mathbf{P}), \mathbf{c}) \in \Pi \text{ with } \|\tilde{\mathbf{c}} - \mathbf{c}\|_0 \leq \epsilon\ell \\ \perp & \text{otherwise.} \end{cases}$.
- 4: If $(\tilde{\mathbf{z}}, \tilde{\mathbf{P}}) = \perp$, **return** 0.
- 5: **return** 1 if $\left\| \tilde{\mathbf{z}} - \frac{1}{\sqrt{m}} \tilde{\mathbf{P}} \mathbf{w}' \right\|_2 \leq \tau$, 0 otherwise.

- G_2^{RB} and G_2^{UF} are identical to G_1^{RB} and G_1^{UF} , respectively, except that the adversary's fingerprinting oracle is replaced by an oracle $\text{FP}_{\text{enc-0}}$. This oracle is identical to FP^{f_1, f_2} , except that it samples $\mathbf{P} \leftarrow \{\pm 1\}^{m \times n}$, $\mathbf{r} \leftarrow \{0, 1\}^\lambda$, $\mathbf{t}^{(1)} \leftarrow \{0, 1\}^s$, and $\mathbf{t}^{(2)} \leftarrow \{0, 1\}^\lambda$ independently, without evaluating f_1 and f_2 , and in Step 4, the oracle instead generates the ciphertext as $\mathbf{ct} = \Sigma.\text{Enc}(\text{sk}, \mathbf{0}; \mathbf{t}^{(1)})$.
- G_3^{RB} and G_3^{UF} are identical to G_2^{RB} and G_2^{UF} , respectively, except that we defer sampling $\mathbf{P}$ until after the execution of $\mathcal{A}$. We give the definition of G_3^{RB} below; G_3^{UF} is defined analogously, with the final winning condition modified accordingly.

$G_3^{\text{RB}}(\mathcal{A}, \lambda, \tau, \Delta_{\text{RB}})$

- 1: $\Pi, \Pi_{\text{FP}} = \emptyset$.
- 2: Sample $\mathbf{S} \leftarrow V_{\ell, n}$, $\text{sk} \leftarrow \Sigma.\text{Gen}(1^\lambda)$.
- 3: $\mathbf{w}' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{FP}'_{\text{enc-0}}}((\mathbf{S}, \text{sk}), \cdot, \cdot)}(1^\lambda)$.
- 4: For each $(\mathbf{w} + \mathbf{g}, \mathbf{c}) \in \Pi$, sample $\mathbf{P} \leftarrow \{\pm 1\}^{m \times n}$ and replace the entry in Π with $((\lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_\eta, \mathbf{P}), \mathbf{c})$.
- 5: Output 1 if both:
 - $\text{Ver}_{\tau, \text{no-decrypt}}(\mathbf{S}, \text{sk}, \mathbf{w}') = 0$.
 - $\exists \mathbf{w} \in \Pi_{\text{FP}}$ such that $\|\mathbf{w} - \mathbf{w}'\|_2 \leq \Delta_{\text{RB}}$.

$\text{FP}'_{\text{enc-0}}((\mathbf{S}, \text{sk}), (\mathcal{T}, \text{spec}))$

- 1: Sample $\mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}$ and compute $(\mathbf{w}, \text{state}) \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec})$. If $\|\mathbf{g}\|_\infty > \sigma\sqrt{\lambda}$, **return** $\perp$.
- 2: Compute $\mathbf{ct} = \Sigma.\text{Enc}(\text{sk}, \mathbf{0}; \mathbf{t}^{(1)})$ where $\mathbf{t}^{(1)} \leftarrow \{0, 1\}^s$ and $\mathbf{c} = \text{Encode}(\mathbf{ct}) \in \{0, 1\}^\ell$.
- 3: Add $(\mathbf{w} + \mathbf{g}, \mathbf{c})$ to Π .
- 4: For $i \in [\ell]$, let $c'_i = \frac{3}{4} - \frac{c_i}{2}$.
- 5: Sample $\mathbf{g}_{\text{FP}} \leftarrow \text{Embed}(\mathbf{S}, \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{w}, \mathbf{c}', \mathbf{g})$.
- 6: Add $\mathbf{w} + \mathbf{g}_{\text{FP}}$ to Π_{FP} .
- 7: **return** $\mathbf{w} + \mathbf{g}_{\text{FP}}$.

- G_4^{RB} and G_4^{UF} are identical to G_3^{RB} and G_3^{UF} , respectively, except that the games abort and output 0 if, in any fingerprinting query made by $\mathcal{A}$, $\|\mathbf{g} - \mathbf{g}_{\text{FP}}\|_2 > \sigma\sqrt{C\ell}$ for a particular constant $C > 0$.

6.3 Security proofs

For two security games G, G' that take as input an adversary $\mathcal{A}$, a security parameter λ , and auxiliary inputs aux , we write $G \approx_c G'$ if the two games are computationally indistinguishable; that is, for every PPT

adversary $\mathcal{A}$ and every auxiliary input $\mathbf{aux}$,

$$|\Pr[G(\mathcal{A}, \lambda, \mathbf{aux}) = 1] - \Pr[G'(\mathcal{A}, \lambda, \mathbf{aux}) = 1]| \leq \text{negl}(\lambda).$$

We write $G \approx_s G'$ if the two games are statistically indistinguishable, meaning that the above condition holds even for computationally *unbounded* adversaries.

Claim 6.1. *Assume PRF and MAC are pseudorandom function families. Then the robustness and unforgeability games for the predicates $\text{L2}(\Delta_{\text{RB}})$ and $\text{L2}(\Delta_{\text{UF}})$ are computationally indistinguishable from G_0^{RB} and G_0^{UF} , respectively.*

Proof. We prove the claim using three hybrids. First, swap PRF_k with a uniformly random function f_1 . By the pseudorandomness of the PRF, this changes any adversary $\mathcal{A}$'s winning probability by at most a negligible amount. Next, replace the PRF $\text{MAC}_{k_{\text{MAC}}}$ with a uniformly random function f_2 .

It remains to account for the additional abort condition in FP^{f_1, f_2} . Since $\mathcal{A}$ makes at most $\text{poly}(\lambda)$ fingerprinting queries, and each seed $\mathbf{r}$ is sampled independently and uniformly from $\{0, 1\}^\lambda$, the probability that any two seeds collide is $\text{negl}(\lambda)$, changing the winning probability by at most a negligible amount. $\square$

Claim 6.2. *Assume that Σ satisfies perfect correctness. Then*

$$G_0^{\text{RB}} \approx_s G_1^{\text{RB}} \quad \text{and} \quad G_0^{\text{UF}} \approx_s G_1^{\text{UF}}.$$

Proof. The only difference between G_0 and G_1 is how the final verification is performed. Let $\mathcal{A}$ be an adversary and denote $\mathbf{w}' \leftarrow \mathcal{A}^{\text{O}_{\text{FP}}(1^\lambda, (\mathbf{s}, \text{sk}), \cdot, \cdot)}$ its output. Denote by Π, Π_{FP} the sets collected during $\mathcal{A}$'s fingerprinting queries and $\tilde{\mathbf{c}}$ the codeword obtained from the weights $\mathbf{w}'$ during the final verification check. We consider two cases.

First, suppose there exists some $\mathbf{c}$ such that $((\mathbf{z}, \mathbf{P}), \mathbf{c}) \in \Pi$ and $\|\tilde{\mathbf{c}} - \mathbf{c}\|_0 \leq \epsilon\ell$. In G_1 , the final verification $\text{Ver}_{\tau, \text{no-decrypt}}$ sets $(\tilde{\mathbf{z}}, \tilde{\mathbf{P}})$ to $(\mathbf{z}, \mathbf{P})$. We show that $\text{Ver}_{\tau}^{f_1, f_2}$ in G_0 does the same. Since $((\mathbf{z}, \mathbf{P}), \mathbf{c}) \in \Pi$, there exists some $\mathbf{r} \leftarrow \{0, 1\}^\lambda$, $\mathbf{t}^{(1)}$, and $\mathbf{t}^{(2)}$ such that $(\mathbf{P}, \mathbf{t}^{(1)}) = f_1(\mathbf{r})$, $\mathbf{t}^{(2)} = f_2(\mathbf{z}, \mathbf{r})$, and $\mathbf{c} = \text{Encode}(\mathbf{ct})$, where $\mathbf{ct} = \Sigma.\text{Enc}(\text{sk}, (\mathbf{z}, \mathbf{r}, \mathbf{t}^{(2)}); \mathbf{t}^{(1)})$. Since the error-correcting code corrects an ϵ -fraction of errors, $\tilde{\mathbf{c}}$ decodes to $\mathbf{ct}$. By perfect correctness of Σ , decrypting $\mathbf{ct}$ recovers $(\mathbf{z}, \mathbf{r}, \mathbf{t}^{(2)})$ exactly. Therefore, the check that $f_2(\mathbf{z}, \mathbf{r}) = \mathbf{t}^{(2)}$ passes, and $\text{Ver}_{\tau}^{f_1, f_2}$ recovers $\tilde{\mathbf{P}} = \mathbf{P}$ using $f_1(\mathbf{r})$ and sets $\tilde{\mathbf{z}} = \mathbf{z}$, just as in G_1 .

Now suppose that for every $((\mathbf{z}, \mathbf{P}), \mathbf{c}) \in \Pi$, we have $\|\tilde{\mathbf{c}} - \mathbf{c}\|_0 > \epsilon\ell$. In this case, in G_1 , $\text{Ver}_{\tau, \text{no-decrypt}}$ outputs 0. We show that $\text{Ver}_{\tau}^{f_1, f_2}$ also outputs 0 except with negligible probability. Suppose, for contradiction, that $\text{Ver}_{\tau}^{f_1, f_2}$ outputs 1 with inverse polynomial probability. Whenever this occurs, $\text{Ver}_{\tau}^{f_1, f_2}$ decrypts $\tilde{\mathbf{c}}$ to some $(\tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{t}}^{(2)}) \neq \perp$ and evaluates $f_1(\tilde{\mathbf{r}})$ to obtain $(\tilde{\mathbf{P}}, \tilde{\mathbf{t}}^{(1)})$. These values then satisfy

$$f_2(\tilde{\mathbf{z}}, \tilde{\mathbf{r}}) = \tilde{\mathbf{t}}^{(2)} \quad \text{and} \quad \|\tilde{\mathbf{c}} - \mathbf{c}\|_0 \leq \epsilon\ell, \quad \text{where } \mathbf{c} = \text{Encode}(\Sigma.\text{Enc}(\text{sk}, (\tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{t}}^{(2)}); \tilde{\mathbf{t}}^{(1)})).$$

If $f_2(\tilde{\mathbf{z}}, \tilde{\mathbf{r}}) = \tilde{\mathbf{t}}^{(2)}$ and the pair $(\tilde{\mathbf{z}}, \tilde{\mathbf{r}})$ is equal to a message-random seed sampled in a previous fingerprinting query, then the corresponding codeword $\mathbf{c}$ would have been added to Π . This contradicts our assumption that $\tilde{\mathbf{c}}$ is more than $\epsilon\ell$ far in Hamming from every codeword in Π . Therefore, the pair $(\tilde{\mathbf{z}}, \tilde{\mathbf{r}})$ was not used in a previous fingerprinting query, and so this is the first time f_2 is being called on it. Since f_2 is a uniformly random function, its output on the pair is uniformly random. Hence, the probability of the above event is negligible, contradicting our assumption. $\square$

Claim 6.3. *Assume Σ is IND-CPA secure. Then*

$$G_1^{\text{RB}} \approx_c G_2^{\text{RB}} \quad \text{and} \quad G_1^{\text{UF}} \approx_c G_2^{\text{UF}}.$$

Proof. We prove the claim for the robustness games, the argument for the unforgeability games is identical. Let $\mathcal{A}$ be a PPT adversary, let $q = \text{poly}(\lambda)$ be the number of fingerprinting queries made by $\mathcal{A}$, and define hybrids $H_0, \dots, H_q$ where, in H_i , the first i fingerprinting queries are answered as in G_2^{RB} and the rest as in G_1^{RB} . Then $H_0 = G_1^{\text{RB}}$ and $H_q = G_2^{\text{RB}}$. We will show $H_{i-1} \approx_c H_i$ for each $i \in [q]$.

Suppose PPT $\mathcal{A}$ distinguishes H_{i-1} from H_i with inverse polynomial advantage. We build an IND-CPA adversary $\mathcal{B}$. $\mathcal{B}$ simulates the game for $\mathcal{A}$ exactly, answering all queries other than the i th precisely as in H_{i-1}/H_i (the two hybrids agree there). At the i th query (D, spec) , $\mathcal{B}$ samples

$$\mathbf{w} \leftarrow \text{SampleFirst}^D(1^\lambda, \text{spec}), \quad \mathbf{g} \leftarrow D_{\mathbb{R}^n, \sigma}, \quad \mathbf{r} \leftarrow \{0, 1\}^\lambda, \quad \mathbf{P} \leftarrow \{\pm 1\}^{m \times n}, \quad \mathbf{t}^{(2)} \leftarrow \{0, 1\}^\lambda,$$

and aborts if $\mathbf{r}$ collides with any previous seed, as the fingerprinting algorithm in H_i would. Let $\mathbf{z} := \lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_\eta$. Since in H_{i-1} , all $\mathbf{r}$'s are distinct and f_1, f_2 are uniformly random functions, $f_1(\mathbf{r})$ and $f_2(\mathbf{z}, \mathbf{r})$ are both uniformly random, independent of prior queries; so $\mathcal{B}$'s freshly sampled $\mathbf{P}, \mathbf{t}^{(2)}$ have exactly the right distribution. $\mathcal{B}$ then decides on two messages

$$\mathbf{m}^{(0)} = (\mathbf{z}, \mathbf{r}, \mathbf{t}^{(2)}), \quad \mathbf{m}^{(1)} = \mathbf{0},$$

submits $(\mathbf{m}^{(0)}, \mathbf{m}^{(1)})$ to its challenger, and receives $\mathbf{ct} \leftarrow \Sigma.\text{Enc}(\text{sk}, \mathbf{m}^{(b)}; \mathbf{t}^{(1)})$ for a fresh $\mathbf{t}^{(1)} \leftarrow \{0, 1\}^s$. We can see that if $b = 0$, $\mathbf{ct}$ is identically distributed to the i th ciphertext in H_{i-1} and if $b = 1$, $\mathbf{ct}$ is identically distributed to the i th ciphertext in H_i .

$\mathcal{B}$ then encodes $\mathbf{ct}$, computes $\mathbf{c}' = \frac{3}{4} - \frac{1}{2}\mathbf{c}$, samples $\mathbf{g}_{\text{FP}}$ via **Embed**, updates Π, Π_{FP} as the game prescribes, and returns $\mathbf{w} + \mathbf{g}_{\text{FP}}$ to $\mathcal{A}$; all remaining queries and the final verification are simulated as in the respective hybrid. Note that this is possible because, in these hybrids, verification does not require knowing the decryption key sk , and no part of the game requires knowing the encryption randomness $\mathbf{t}^{(1)}$.

Thus $\mathcal{B}$ perfectly simulates H_{i-1} when $b = 0$ and H_i when $b = 1$, so $\mathcal{A}$'s distinguishing advantage transfers directly to $\mathcal{B}$'s IND-CPA advantage, contradicting the security of Σ . Hence $H_{i-1} \approx_c H_i$ for all $i \in [q]$, and by a hybrid argument $G_1^{\text{RB}} \approx_c G_2^{\text{RB}}$. $\square$

Claim 6.4. $G_2^{\text{RB}} \stackrel{d}{=} G_3^{\text{RB}}$ and $G_2^{\text{UF}} \stackrel{d}{=} G_3^{\text{UF}}$. Moreover, assuming $\ell = \omega(\log \lambda)$ and $\sqrt{\frac{\gamma^2 + \beta^2}{2}} \geq \ln \sqrt{\frac{6\ell}{\pi}}$, then $G_3^{\text{RB}} \approx_s G_4^{\text{RB}}$ and $G_3^{\text{UF}} \approx_s G_4^{\text{UF}}$.

Proof. The first two claims follow because in G_2 , the only use of $\mathbf{P}$ before the final verification check is in constructing the hidden $(\mathbf{z}, \mathbf{P}) \in \Pi$. The fingerprinted weights no longer depend on $(\mathbf{z}, \mathbf{P})$, and Π is not exposed to $\mathcal{A}$. Thus, we may defer sampling $\mathbf{P}$ until after $\mathcal{A}$ terminates and reconstruct Π with exactly the same distribution.

The latter two claims follow from Lemma 5.4, which guarantees that, for each of the polynomially many fingerprinting queries, $\|\mathbf{g} - \mathbf{g}_{\text{FP}}\|_2 \leq \sigma\sqrt{C\ell}$ for a particular constant $C > 0$, except with negligible probability. $\square$

Lemma 6.2 (Hardness of robustness game). *Let $\delta_{\text{JL}} \in (0, 1)$ and $\epsilon \in (0, \frac{1}{4})$ be constants and let the projected dimension $m = m(\lambda)$ and number of secrets $\ell = \ell(\lambda)$ satisfy*

$$m \geq \frac{4 + 2\omega(\log \lambda)}{\delta_{\text{JL}}^2/2 - \delta_{\text{JL}}^3/3} \quad \text{and} \quad \omega(\log \lambda) \leq \ell \leq o(n).$$

Let $\beta = \beta(\lambda), \gamma = \gamma(\lambda)$ such that $\frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5}{2\pi}} < \frac{\sqrt{\epsilon}}{4}$ be parameters. Assume there exists some $t = t(\lambda)$ such that no PPT adversary can solve $t\text{-PA}(n, \ell, \beta, \gamma)$ with advantage $\delta = \delta(\lambda)$. Then for any robustness parameter $\Delta_{\text{RB}} = \Delta_{\text{RB}}(\lambda)$ satisfying

$$\Delta_{\text{RB}} \leq \frac{\gamma}{\hat{\gamma}^2} \cdot \frac{\sigma\alpha\sqrt{n}}{\sqrt{t}}, \quad \text{where } \alpha = \frac{\sqrt{\epsilon}}{4} - \frac{\hat{\gamma}\beta}{\gamma} \sqrt{\frac{5}{2\pi}},$$

any threshold $\tau = \tau(\lambda)$ such that

$$\tau \geq \sqrt{1 + \delta_{\text{JL}}} \left(\Delta_{\text{RB}} + \sigma\sqrt{C\ell} \right) + \eta\sqrt{m}, \quad \text{where } C > 0 \text{ is a particular constant,}$$

and every PPT adversary $\mathcal{A}$ making $q = \text{poly}(\lambda)$ fingerprinting queries,

$$\Pr[G_4^{\text{RB}}(\mathcal{A}, \lambda, \tau, \Delta_{\text{RB}}) = 1] \leq \text{negl}(\lambda) + q\delta$$

Proof. Let $\mathcal{A}$ be a PPT adversary. Denote by the random variables Π, Π_{FP} the sets collected during G_4^{RB} and $\mathbf{w}'$ the output of $\mathcal{A}$. Condition on the event that there exists $\mathbf{w}_{\text{FP}} = (\mathbf{w} + \mathbf{g}_{\text{FP}}) \in \Pi_{\text{FP}}$ which satisfies $\|\mathbf{w}' - \mathbf{w}_{\text{FP}}\|_2 \leq \Delta_{\text{RB}}$. We will first show that, except with probability $q\delta + \text{negl}(\lambda)$ over $\mathbf{S}$ and the adversary's randomness, the verifier recovers the message associated with $\mathbf{w}_{\text{FP}}$ from $\mathbf{w}'$. Conditioned on this event, we then show that the final verification test accepts except with probability $2^{-\omega(\log \lambda)}$ over the randomness of the $\mathbf{P}$'s.

Recovering the message. Let $\mathbf{c}$ be the codeword used while generating $\mathbf{w}_{\text{FP}}$ and let $\mathbf{c}' = \frac{3}{4} - \frac{1}{2}\mathbf{c}$. Following the proof of Lemma 5.9

$$\zeta = \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top \mathbf{w}' = \left(\mathbf{c}' + \mathbf{e} + \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w}' - \mathbf{w}_{\text{FP}}) \right) \pmod{1},$$

where $\mathbf{e} \leftarrow D_{\mathbb{R}^\ell, \frac{\hat{\gamma}\beta}{\gamma}}$. Define $\mathbf{e}' := \mathbf{e} + \frac{\hat{\gamma}^2}{\gamma\sigma} \mathbf{S}^\top (\mathbf{w}' - \mathbf{w}_{\text{FP}}) \pmod{1}$. Since $\|\mathbf{w}' - \mathbf{w}_{\text{FP}}\|_2 \leq \Delta_{\text{RB}} \leq \frac{\gamma\sigma\alpha\sqrt{n}}{\hat{\gamma}^2\sqrt{\ell}}$, we can again follow the proof of Lemma 5.9 to see that, assuming that no PPT adversary can solve $t\text{-PA}(n, \ell, \beta, \gamma)$ with advantage δ , together our choice of α implies that

$$\|\mathbf{e}'\|_2 \leq \frac{\sqrt{\epsilon\ell}}{4}$$

except with probability at most $q\delta + \text{negl}(\lambda)$.

When this occurs, Lemma 6.3 tells us that at most $\epsilon\ell$ coordinates satisfy $|e'_i| \geq \frac{1}{4}$. Recall $\zeta_i = (\frac{3}{4} - \frac{c_i}{2} + e'_i) \pmod{1}$, so whenever $|e'_i| < 1/4$, the sign of ζ_i is 1 if $c_i = 1$, -1 if $c_i = 0$. Consequently, $\|\tilde{\mathbf{c}} - \mathbf{c}\|_0 \leq \epsilon\ell$. Therefore, in $\text{Ver}_{\tau, \text{no-decrypt}}$, $(\tilde{\mathbf{z}}, \tilde{\mathbf{P}})$ is set to the message associated with $\mathbf{c}$,

$$(\tilde{\mathbf{z}}, \tilde{\mathbf{P}}) = \left(\lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_\eta, \mathbf{P} \right)$$

where $\mathbf{P} \leftarrow \{\pm 1\}^{m \times n}$ is fresh and $\mathbf{g}$ is the standard Gaussian vector sampled while producing $\mathbf{w}_{\text{FP}}$.

Final verification test. Now, fix a secret $\mathbf{S}$ and fix the adversary's randomness such that the recovery event described above occurs and the game doesn't abort early. Denote $\mathbf{w}'$ the adversary's output and $\mathbf{w}_{\text{FP}} = (\mathbf{w} + \mathbf{g}_{\text{FP}}) \in \Pi_{\text{FP}}$ the close weights, which are now both fixed. Denote by $\mathbf{g}$ the continuous Gaussian vector sampled during the fingerprinting query that returned $\mathbf{w}_{\text{FP}}$.

Note $\mathbf{P}$ is a uniformly random matrix in $\{\pm 1\}^{m \times n}$ that is independent of the fixed vectors $\mathbf{w} + \mathbf{g}$ and $\mathbf{w}'$. By Johnson-Lindenstrauss for binary matrices [Ach03, Theorem 1.1] and our choice of m ,

$$\Pr_{\mathbf{P}} \left[\left\| \frac{1}{\sqrt{m}} \mathbf{P}((\mathbf{w} + \mathbf{g}) - \mathbf{w}') \right\|_2 > \sqrt{1 + \delta_{\text{JL}}} \|(\mathbf{w} + \mathbf{g}) - \mathbf{w}'\|_2 \right] \leq 2^{-\omega(\log \lambda)}.$$

Moreover, since the game aborts unless $\|\mathbf{g} - \mathbf{g}_{\text{FP}}\|_2 \leq \sigma\sqrt{C\ell}$, by triangle inequality,

$$\begin{aligned} \|(\mathbf{w} + \mathbf{g}) - \mathbf{w}'\|_2 &\leq \|(\mathbf{w} + \mathbf{g}) - \mathbf{w}_{\text{FP}}\|_2 + \|\mathbf{w}_{\text{FP}} - \mathbf{w}'\|_2 \\ &\leq \sigma\sqrt{C\ell} + \Delta_{\text{RB}}, \end{aligned}$$

Taking into account the rounding error, we have that verification accepts except with probability $2^{-\omega(\log \lambda)}$:

$$\begin{aligned} \left\| \lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_\eta - \frac{1}{\sqrt{m}} \mathbf{P}\mathbf{w}' \right\|_2 &\leq \left\| \lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_\eta - \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \right\|_2 + \left\| \frac{1}{\sqrt{m}} \mathbf{P}((\mathbf{w} + \mathbf{g}) - \mathbf{w}') \right\|_2 \\ &\leq \eta\sqrt{m} + \sqrt{1 + \delta_{\text{JL}}} (\Delta_{\text{RB}} + \sigma\sqrt{C\ell}) \leq \tau. \end{aligned}$$

Therefore, whenever G_4^{RB} outputs 1, either message recovery fails or we fall in this $2^{-\omega(\log \lambda)}$ probability event that verification fails. The result follows by a union bound. $\square$

The proof above relied on a simple fact, that many of the coordinates of an ℓ_2 bounded vector will be small in magnitude.

Lemma 6.3. *Let $\alpha \in (0, 1)$, $\mathbf{e} \in [-1/2, 1/2]^\ell$ with $\|\mathbf{e}\|_2 \leq \alpha\sqrt{\ell}$. It holds that*

$$|\{i : |e_i| \geq 1/4\}| \leq 16\alpha^2\ell$$

Proof. Let $T := \{i : |e_i| \geq 1/4\}$. This follows from the fact that

$$\begin{aligned} \sum_{i \in T} e_i^2 + \sum_{i \notin T} e_i^2 &\leq \alpha^2 \ell \\ \implies |T|/16 + \sum_{i \notin T} e_i^2 &\leq \alpha^2 \ell \\ \implies |T| &\leq 16\alpha^2 \ell. \end{aligned}$$

□

Lemma 6.4 (Hardness of unforgeability game). *Let $\delta_{\text{JL}} \in (0, 1)$ be any constant and let the projected dimension $m = m(\lambda)$ and number of secrets $\ell = \ell(\lambda)$ satisfy*

$$m \geq \frac{4 + 2\omega(\log \lambda)}{\delta_{\text{JL}}^2/2 - \delta_{\text{JL}}^3/3} \quad \text{and} \quad \omega(\log \lambda) \leq \ell \leq o(n).$$

For any threshold $\tau = \tau(\lambda) > 0$, any unforgeability parameter $\Delta_{\text{UF}} = \Delta_{\text{UF}}(\lambda)$ satisfying

$$\Delta_{\text{UF}} \geq \frac{\tau + \eta\sqrt{m}}{\sqrt{1 - \delta_{\text{JL}}}} + \sigma\sqrt{C\ell}, \quad \text{where } C > 0 \text{ is a particular constant,}$$

and every adversary $\mathcal{A}$,

$$\Pr[G_4^{\text{UF}}(\mathcal{A}, \lambda, \tau, \Delta_{\text{UF}}) = 1] \leq \text{negl}(\lambda).$$

Proof. Fix $\mathbf{S} \in V_{\ell, n}$ and the randomness of the adversary $\mathcal{A}$, such that 1) their output $\mathbf{w}'$ satisfies $\|\mathbf{w}' - \mathbf{w}_{\text{FP}}\|_2 > \Delta_{\text{UF}}$ for all $\mathbf{w}_{\text{FP}} \in \Pi_{\text{FP}}$, 2) $(\tilde{\mathbf{z}}, \tilde{\mathbf{P}})$ computed during $\text{Ver}_{\tau, \text{no-decrypt}}$ is not $\perp$, and 3) the game does not abort early. Note all these events must occur in order for G_4^{UF} to output 1. Then $(\tilde{\mathbf{z}}, \tilde{\mathbf{P}})$ must equal $(\lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \rfloor_{\eta}, \mathbf{P})$ for a random $\mathbf{P} \leftarrow \{\pm 1\}^{m \times n}$ where $\mathbf{w} + \mathbf{g}$ corresponds to some fingerprinted $(\mathbf{w} + \mathbf{g}_{\text{FP}}) \in \Pi_{\text{FP}}$. Again by Johnson-Lindenstrauss [Ach03, Theorem 1.1],

$$\Pr_{\mathbf{P}} \left[\left\| \frac{1}{\sqrt{m}} \mathbf{P}((\mathbf{w} + \mathbf{g}) - \mathbf{w}') \right\|_2 < \sqrt{1 - \delta_{\text{JL}}} \|(\mathbf{w} + \mathbf{g}) - \mathbf{w}'\|_2 \right] \leq 2^{-\omega(\log \lambda)}.$$

Moreover, we know that

$$\|\mathbf{w}' - (\mathbf{w} + \mathbf{g}_{\text{FP}})\|_2 > \Delta_{\text{UF}} \implies \|\mathbf{w}' - (\mathbf{w} + \mathbf{g})\|_2 > \Delta_{\text{UF}} - \sigma\sqrt{C\ell}.$$

Combining these, verification rejects except with probability $2^{-\omega(\log \lambda)}$:

$$\begin{aligned} \left\| \left\lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \right\rfloor_{\eta} - \frac{1}{\sqrt{m}} \mathbf{P} \mathbf{w}' \right\|_2 &\geq \left\| \frac{1}{\sqrt{m}} \mathbf{P}((\mathbf{w} + \mathbf{g}) - \mathbf{w}') \right\|_2 - \left\| \left\lfloor \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \right\rfloor_{\eta} - \frac{1}{\sqrt{m}} \mathbf{P}(\mathbf{w} + \mathbf{g}) \right\|_2 \\ &> \sqrt{1 - \delta_{\text{JL}}} (\Delta_{\text{UF}} - \sigma\sqrt{C\ell}) - \eta\sqrt{m} \geq \tau. \end{aligned}$$

So G_4^{UF} outputting 1 implies a negligible probability event. □

7 Pancake Alignment Cryptanalysis

We include cryptanalysis to better understand the hardness of the pancake alignment problem. We first show in Section 7.1 that when the approximation parameter satisfies $t < 1$, there is an efficient attack. However, we show once $t > 1$ is a constant, this same attack no longer works, and that the problem is hard for an adversary that makes *no* queries to its Gaussian pancake oracle. We then give two reductions from CLWE to pancake alignment. The first, in Section 7.2, applies for $t = \tilde{\Omega}(\lambda)$, while the second, in Section 7.3, applies for $t = 1/\epsilon$, where ϵ is the adversary's advantage.

7.1 An efficient attack for $t < 1$

In this section, we show that when $t < 1$, an adversary that outputs a fixed unit vector $\mathbf{v} \in \mathbb{S}^{n-1}$ succeeds with non-negligible probability. We also show that this same attack fails to succeed when $t > 1$ is a constant. First, we present a fact that makes it easier to analyze the distribution of $\mathbf{S}^\top \mathbf{v}$ for uniformly random $\mathbf{S} \leftarrow V_{\ell,n}$.

Fact 7.0.1. *Let $\ell, n \in \mathbb{N}$ be dimensions such that $1 \leq \ell \leq n$. Let $\mathbf{v} \in \mathbb{S}^{n-1}$ be fixed and let $\mathbf{S} \leftarrow V_{\ell,n}$. Then $\mathbf{S}^\top \mathbf{v}$ is distributed as the first ℓ coordinates of a vector sampled uniformly from the unit sphere $\mathbb{S}^{n-1}$.*

Proof. Let $\mathbf{R}$ be the rotation that rotates $\mathbf{v}$ to the first canonical basis vector, $\mathbf{R}\mathbf{v} = \mathbf{b}^{(1)}$. Since the uniform distribution over $V_{\ell,n}$ is rotationally invariant,

$$\mathbf{S}^\top \mathbf{v} \stackrel{d}{=} (\mathbf{R}^\top \mathbf{S})^\top \mathbf{v} = \mathbf{S}^\top \mathbf{b}^{(1)}.$$

Therefore, it suffices to show that the first row of $\mathbf{S}$ satisfies the claim. Let $\mathbf{Q} \leftarrow V_{n,n}$. It is a standard fact that $\mathbf{S} \leftarrow V_{\ell,n}$ for $\ell \leq n$ can be obtained by taking the first ℓ columns of $\mathbf{Q}$, i.e.,

$$\mathbf{S} = (\mathbf{Q}[1] \parallel \dots \parallel \mathbf{Q}[\ell])$$

Formally, $\mathbf{Q} \leftarrow V_{n,n}$ means $\mathbf{Q}$ sampled from the Haar measure over the orthogonal group $O(n)$. By Haar invariance, $\mathbf{Q}$ and $\mathbf{Q}^\top$ have the same distribution. Hence the first row of $\mathbf{Q}$ has the same distribution as the first column, which is uniform on the unit sphere $\mathbb{S}^{n-1}$. We can conclude that the first row of $\mathbf{S}$, which consists of the first ℓ entries of the first row of $\mathbf{Q}$, satisfies

$$(\mathbf{Q}[1]_1, \mathbf{Q}[2]_1, \dots, \mathbf{Q}[\ell]_1) \stackrel{d}{=} (s_1, \dots, s_\ell), \quad \mathbf{s} \leftarrow \mathbb{S}^{n-1}.$$

□

We can now obtain tail bounds on the alignment of a fixed vector $\mathbf{v}$ independent of $\mathbf{S}$.

Proposition 7.1. *Let $\ell, n \in \mathbb{N}$ with $n - \ell > \ell^{14}$, let $\mathbf{v} \in \mathbb{S}^{n-1}$ be fixed, and let $\mathbf{S} \leftarrow V_{\ell,n}$ be random. For any $s > 0$,*

$$\Pr[\|\mathbf{S}^\top \mathbf{v}\|_2^2 < \frac{\ell - 3s}{n}] \leq \exp\left(-\frac{s^2}{\ell}\right),$$

and

$$\Pr[\|\mathbf{S}^\top \mathbf{v}\|_2^2 > \frac{\ell + 3s}{n}] \leq \exp\left(-\frac{s^2}{\ell + s}\right).$$

Proof. Due to Fact 7.0.1, it suffices to analyze the squared norm of the first ℓ coordinates of $\mathbf{s} \leftarrow \mathbb{S}^{n-1}$, $\sum_{i=1}^{\ell} s_i^2$, which is distributed as the Beta distribution, $\text{Beta}(\frac{\ell}{2}, \frac{n-\ell}{2})$. Using [Sko23, Theorem 1], we can see that $\|\mathbf{S}^\top \mathbf{v}\|_2^2$ is concentrated around its mean:

$$\Pr[\|\mathbf{S}^\top \mathbf{v}\|_2^2 > \frac{\ell}{n} + \epsilon] \leq \exp\left(-\frac{\epsilon^2}{2(v + \frac{c\epsilon}{3})}\right) \quad \text{and} \quad \Pr[\|\mathbf{S}^\top \mathbf{v}\|_2^2 < \frac{\ell}{n} - \epsilon] \leq \exp\left(-\frac{\epsilon^2}{2v}\right)$$

for any $\epsilon > 0$. Here, v is the variance and c is an error term; in our case

$$v := \frac{2\ell(n-\ell)}{n^2(n+2)}, \quad c := \frac{4(n-2\ell)}{n(n+1)}.$$

In particular, if we set $\epsilon = \frac{3s}{n}$, we have

$$\begin{aligned} 2v &= \frac{4\ell(n-\ell)}{n^2(n+2)} \leq 4\frac{\ell}{n^2} \\ 2\frac{c\epsilon}{3} &= \frac{8s(n-2\ell)}{n^2(n+1)} \leq 8\frac{s}{n^2} \end{aligned}$$

¹⁴We can get a similar bound for the case where $n - \ell \leq \ell$.

Meaning $\|\mathbf{S}^\top \mathbf{v}\|_2^2 > \frac{\ell+3s}{n}$ with probability at most

$$\exp\left(-\frac{\epsilon^2}{2(v + \frac{c\epsilon}{3})}\right) \leq \exp\left(-\frac{9s^2/n^2}{8(\ell+s)/n^2}\right) \leq \exp(-s^2/(\ell+s))$$

and $\|\mathbf{S}^\top \mathbf{v}\|_2^2 < \frac{\ell-3s}{n}$ with probability at most $\exp(-s^2/\ell)$. $\square$

Remark 7.1. Proposition 7.1 characterizes the regimes in which the attack that outputs a fixed vector $\mathbf{v}$ succeeds. For simplicity, we consider the setting where $\ell = \omega(\log \lambda)$. When $\ell = O(\log \lambda)$, a similar analysis applies, but the attack ceases to succeed once $t \geq \omega(\log \lambda)$.

- When $t < 1$ is a constant, setting $s = \frac{\ell(1-t)}{3}$ and applying the first part of the proposition shows that $\|\mathbf{S}^\top \mathbf{v}\|_2^2 \geq t \frac{\ell}{n}$ with probability $1 - \exp(-\Theta(\ell))$, therefore the attack succeeds.
- When $t > 1$ is a constant, setting $s = \frac{\ell(t-1)}{3}$ and applying the second part of the proposition shows that $\|\mathbf{S}^\top \mathbf{v}\|_2^2 > t \frac{\ell}{n}$ with probability at most $\exp(-\Theta(\ell))$, which is negligible; the attack no longer succeeds.

Another simple attack worth considering is for the adversary to output a normalized Gaussian pancake vector $\frac{\mathbf{y}}{\|\mathbf{y}\|_2}$, where $\mathbf{y}$ is obtained from its $\mathbf{GP}_{\beta, \gamma}$ oracle. Since $\mathbf{y}$ is correlated with the secret, it is not immediately clear whether this affects alignment. However, recall from Remark 4.4, the alignment between the secret $\mathbf{S}$ and a sample $\mathbf{y}$ satisfies

$$\|\mathbf{S}^\top \mathbf{y}\|_2^2 \stackrel{d}{=} \left\| \frac{\gamma}{\hat{\gamma}^2} (\mathbf{k} + \mathbf{e}) \right\|_2^2, \quad \text{where } \mathbf{k} \leftarrow D_{\mathbb{Z}^\ell + \mathbf{t}, \hat{\gamma}} \text{ for some shift } \mathbf{t} \in [0, 1)^\ell \text{ and } \mathbf{e} \leftarrow D_{\mathbb{R}^\ell, \frac{\hat{\gamma}\beta}{\gamma}}.$$

The random variable $\frac{\gamma}{\hat{\gamma}^2} (\mathbf{k} + \mathbf{e})$ is (approximately)¹⁵ sub-Gaussian with parameter $\frac{\gamma}{\hat{\gamma}^2} \sqrt{\hat{\gamma}^2 + (\frac{\hat{\gamma}\beta}{\gamma})^2} = 1$ [MP11, Lemma 2.8], which means that $\|\mathbf{S}^\top \mathbf{y}\|_2^2 \leq O(\ell)$ with overwhelming probability in ℓ . Combining this with the fact that the norm of a Gaussian pancake vector satisfies $\|\mathbf{y}\|_2^2 = \Theta(n)$ with overwhelming probability, then the alignment $\frac{\|\mathbf{S}^\top \mathbf{y}\|_2^2}{\|\mathbf{y}\|_2^2} \leq O(\ell/n)$, so such an attack fails once t is a large enough constant.

7.2 Reduction from KDS Gaussian Pancake for $t = \tilde{\Omega}(\sqrt{\lambda})$

In this section, we present the proof for the key component of Theorem 4.1. This reduction consists of three parts. We first show that, assuming the KDS Gaussian pancake problem is hard, then for a single secret $\ell = 1$, the pancake alignment problem is hard when $t = \sqrt{n\omega(\log \lambda)}$, where n is the dimension. Next, we show how to lift hardness from lower to higher dimensions: hardness in dimension $\lambda < n$ implies hardness in dimension n for roughly the same t . Combined with the first result, this gives hardness for $t = \sqrt{\lambda\omega(\log \lambda)}$ even when $n > \lambda$. Finally, we show that hardness for the single-secret problem implies hardness for the multi-secret problem with roughly the same t , provided that the number of secrets satisfies $\ell = o(n)$.

Proposition 7.2. Let $\lambda \in \mathbb{N}$ be a security parameter and let $\gamma = \gamma(\lambda), \beta = \beta(\lambda)$ be parameters. Assume that no PPT adversary solves $\mathbf{GP}_{\text{shift}}(\lambda, 1, \beta, \gamma)$ with advantage $\epsilon = \epsilon(\lambda)$. Then for any $n = \text{poly}(\lambda)$, $\ell = \ell(\lambda)$ satisfying $n - \ell \geq \lambda$ and $\ell = o(n)$, there exists some t^* satisfying

$$t^* = O(\sqrt{\lambda} \log \lambda)$$

such that for any $t = t(\lambda) \geq t^*$, no PPT adversary can solve $t\text{-PA}(n, \ell, \beta, \gamma)$ with advantage more than $\ell\sqrt{\epsilon} + \text{negl}(\lambda)$.

Proof. Fix n, ℓ satisfying the conditions of the theorem. By Lemma 7.1 with $k = \log^2 \lambda$, no PPT adversary can solve $t'\text{-PA}(\lambda, 1, \beta, \gamma)$ with advantage $\sqrt{\epsilon} + \text{negl}(\lambda)$, where

$$t' = \sqrt{2\lambda} \log \lambda + \frac{\sqrt{2\lambda} \log \lambda}{\sqrt{\lambda - 1}} = \Theta(\sqrt{\lambda} \log \lambda).$$

¹⁵When $\hat{\gamma} > \eta_\epsilon(\mathbb{Z}^\ell)$, it is $\ln(\frac{1+\epsilon}{1-\epsilon})$ sub-Gaussian as in [MP11].

By Lemma 7.2 and the fact that $n - \ell + 1 > \lambda$, no PPT adversary can solve t'' -PA($n - \ell + 1, 1, \beta, \gamma$) with advantage $\sqrt{\epsilon + \text{negl}(\lambda)} + \text{negl}(\lambda)$ where t'' satisfies

$$t'' \left(\frac{\lambda}{t'' + r(n - \ell + 1)} \right) = t', \quad \text{where } r := 4 \frac{\lambda - 1}{n - \ell},$$

which, when rearranged, tells us that

$$t'' = \frac{(t'/\lambda)r(n - \ell + 1)}{1 - (t'/\lambda)} = \Theta\left(\frac{t'}{1 - (\log \lambda / \sqrt{\lambda})}\right) = O(t').$$

Finally, by Lemma 7.3, no PPT adversary can solve t^* -PA(n, ℓ, β, γ) with advantage $\ell(\sqrt{\epsilon + \text{negl}(\lambda)} + \text{negl}(\lambda))$ where

$$t^* = \frac{n}{n - \ell + 1} t'' = \frac{t''}{1 - o(1)}, \quad \text{when } \ell = o(n).$$

Because increasing t only makes the problem harder, for any $t \geq t^* = O(\sqrt{\lambda} \log \lambda)$, no PPT adversary can solve t -PA(n, ℓ, β, γ) with advantage $\ell\sqrt{\epsilon + \text{negl}(\lambda)}$. $\square$

7.2.1 The reduction for a single secret

Before beginning, observe that for any rotation $\mathbf{R} \in \text{SO}(n)$, it is easy to generate samples with a rotated secret, $\text{GP}_{\beta, \gamma}(\mathbf{R}\mathbf{s}, \cdot, \cdot)$, using only an oracle for the unrotated secret $\text{GP}_{\beta, \gamma}(\mathbf{s}, \cdot, \cdot)$. On input $z \in [0, 1]$, $\mathbf{u} \in \mathbb{R}^n$, simply rotate the sample generated by the oracle,

$$\mathbf{R}\mathbf{y}, \quad \text{where } \mathbf{y} \leftarrow \text{GP}(\mathbf{s}, \mathbf{R}^\top \mathbf{u}, z).$$

We can see that

$$\mathbf{R}\mathbf{y} \stackrel{d}{=} \mathbf{R}\mathbf{v} + (k/\gamma' + e)\mathbf{R}\mathbf{s},$$

where $\mathbf{R}\mathbf{v}$ is distributed as Gaussian from the subspace orthogonal to $\mathbf{R}\mathbf{s}$, and $k \leftarrow D_{\mathbb{Z}+b, \hat{\gamma}}$ with a shift b satisfying

$$b = (-\mathbf{s}^\top (\mathbf{R}^\top \mathbf{u}) + z \bmod 1) = (-\mathbf{R}\mathbf{s})^\top \mathbf{u} + z \bmod 1),$$

which is exactly a sample from $\text{GP}(\mathbf{R}\mathbf{s}, \mathbf{u}, z)$. This observation will allow us to define a Gaussian pancake adversary that uses a pancake alignment adversary on simulated oracles with randomly rotated secrets. If the pancake alignment adversary is successful, then in the real-oracle case each output vector, after being rotated back, exhibits a common directional bias towards the secret $\mathbf{s}$. In contrast, when the oracle is Null, the output of the pancake alignment adversary is independent of the random rotation. Rotating the resulting vectors back therefore produces uniformly distributed vectors with no common preferred direction. This distinction can be exploited to distinguish the Gaussian pancake oracle from Null.

Lemma 7.1. *Let $n = n(\lambda)$ be the dimension, and let $\beta = \beta(\lambda)$, $\gamma = \gamma(\lambda)$ be parameters. Assume there is no PPT adversary that can solve $\text{GP}_{\text{shift}}(n, 1, \beta, \gamma)$ with advantage ϵ . Then for any $k = k(\lambda)$, there exists some $t^* = \Theta(\sqrt{nk})$ such that for any $t \geq t^*$, no PPT adversary can solve t -PA($n, 1, \beta, \gamma$) with advantage $\sqrt{\epsilon + 8 \exp(-k)}$.*

Proof. For contradiction, assume there is some k , some $t \geq (\sqrt{2kn} + \frac{\sqrt{2kn}}{\sqrt{n-1}}) = \Theta(\sqrt{nk})$, and some PPT adversary $\mathcal{A}$ such that for $\mathbf{s} \leftarrow \mathbb{S}^{n-1}$, $\mathbf{v} \leftarrow \mathcal{A}^{\text{GP}(\mathbf{s}, \cdot, \cdot)}(1^\lambda)$,

$$\Pr[(\mathbf{v}^\top \mathbf{s})^2 \geq t/n] = \sqrt{\epsilon + 8 \exp(-k)}, \quad \text{where } \mathbf{v} \in \mathbb{S}^{n-1}.$$

Then we can construct an adversary $\mathcal{B}^{O(\cdot)}$ against the (worst case) Gaussian pancake distinguishing problem. $\mathcal{B}^{O(\cdot)}(1^\lambda)$ does as follows:

- For $i \in [2]$ sample a uniformly random rotation $\mathbf{R}_i \leftarrow \text{SO}(n)$.
- For $i \in [2]$, run $\mathbf{v}^{(i)} \leftarrow \mathcal{A}^{\mathbf{R}_i O(\mathbf{R}_i^\top \cdot)}(1^\lambda)$ independently with fresh randomness.

- Output 1 if

$$|(\mathbf{R}_1^\top \mathbf{v}^{(1)})^\top (\mathbf{R}_2^\top \mathbf{v}^{(2)})| \geq \sqrt{\frac{2k}{n}},$$

0 otherwise.

Here, $\mathbf{R}_i O(\mathbf{R}_i^\top \cdot, \cdot)$ is the oracle described above, that takes inputs $z \in [0, 1)$, $\mathbf{u} \in \mathbb{R}^n$, queries from the oracle $\mathbf{y} \leftarrow O(\mathbf{R}_i^\top \mathbf{u}, z)$, and rotates the output sample $\mathbf{R}_i \mathbf{y}$. We can see what happens in each case.

Null case: When $\mathcal{B}$ is given a null oracle, meaning $O(\cdot, \cdot)$ ignores its inputs and outputs standard Gaussian vectors, each query to the rotated oracle $\mathbf{R}_i O(\mathbf{R}_i^\top \cdot, \cdot)$ returns $\mathbf{R}_i \mathbf{g}$ where $\mathbf{g} \leftarrow D_{\mathbb{R}^n, 1}$. By rotational invariance, each query is distributed exactly as a query to $O(\cdot, \cdot)$. Therefore, outputs $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}$ are independently distributed as outputs of $\mathcal{A}^{O(\cdot, \cdot)}(1^\lambda)$. We can see that $\mathbf{u}^{(1)} := \mathbf{R}_1^\top \mathbf{v}^{(1)}$ and $\mathbf{u}^{(2)} := \mathbf{R}_2^\top \mathbf{v}^{(2)}$ are distributed as independent uniform vectors from $\mathbb{S}^{n-1}$, since $\mathbf{v}^{(i)}$ is independent of $\mathbf{R}_i$. By Lemma 3.5 and symmetry, conditioned on $\mathbf{u}^{(1)}$,

$$|(\mathbf{u}^{(1)})^\top \mathbf{u}^{(2)}| \geq \sqrt{\frac{2k}{n}},$$

with probability at most $4 \exp(-k)$ over $\mathbf{u}^{(2)}$. Therefore, the same holds with probability over $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$.

Pancake case: When $O(\cdot, \cdot)$ outputs Gaussian pancake vectors for the secret $\mathbf{s} \in \mathbb{S}^{n-1}$, as mentioned above, $\mathbf{R}_i O(\mathbf{R}_i^\top \cdot, \cdot)$ has the same distribution as an oracle $\text{GP}(\mathbf{R}_i \mathbf{s}, \cdot, \cdot)$. We can express the adversary's output as its component in the secret direction plus its component in the orthogonal directions:

$$\mathbf{v}^{(i)} = \sqrt{1 - \delta_i^2} \mathbf{a}^{(i)} + \delta_i \mathbf{R}_i \mathbf{s},$$

where $\delta_i := (\mathbf{R}_i \mathbf{s})^\top \mathbf{v}^{(i)} = \mathbf{s}^\top \mathbf{u}^{(i)}$ is the component in the secret direction, and $\mathbf{a}^{(i)}$ is a random vector in the subspace orthogonal to $\mathbf{R}_i \mathbf{s}$. We then get that after unrotating the adversary's output,

$$\mathbf{u}^{(i)} = \mathbf{R}_i^\top \mathbf{v}^{(i)} = \sqrt{1 - \delta_i^2} \tilde{\mathbf{a}}^{(i)} + \delta_i \mathbf{s},$$

where $\tilde{\mathbf{a}}$ is some vector in the subspace orthogonal to $\mathbf{s}$. We can see that the alignment

$$\begin{aligned} |(\mathbf{u}^{(1)})^\top \mathbf{u}^{(2)}| &= |(\sqrt{1 - \delta_1^2} \tilde{\mathbf{a}}^{(1)} + \delta_1 \mathbf{s})^\top (\sqrt{1 - \delta_2^2} \tilde{\mathbf{a}}^{(2)} + \delta_2 \mathbf{s})| \\ &= |\sqrt{1 - \delta_1^2} \sqrt{1 - \delta_2^2} ((\tilde{\mathbf{a}}^{(1)})^\top \tilde{\mathbf{a}}^{(2)}) + \delta_1 \delta_2| \\ &\geq |\delta_1 \delta_2| - |(\tilde{\mathbf{a}}^{(1)})^\top \tilde{\mathbf{a}}^{(2)}|, \end{aligned}$$

where the last inequality follows from reverse triangle inequality and the fact that $|\delta_i| \leq 1$.

To deal with the second term, intuitively notice that $\tilde{\mathbf{a}}^{(i)}$ is obtained by un-rotating the component $\mathbf{a}^{(i)}$, which describes the portion of the adversary's output that is orthogonal to $\mathbf{R}_i \mathbf{s}$. Because the adversary only has access to the rotation $\mathbf{R}_i$ through the direction $\mathbf{R}_i \mathbf{s}$ revealed by the oracle, it has no information about how $\mathbf{R}_i$ acts on the subspace $\mathbf{s}^\perp$. Therefore, when we un-rotate it to get $\tilde{\mathbf{a}}^{(i)} = \mathbf{R}_i^\top \mathbf{a}^{(i)}$, we get a uniformly random direction within $\mathbf{s}^\perp$. Lemma 7.4 formalizes this intuition, guaranteeing that $\tilde{\mathbf{a}}^{(i)}$ is uniform over $\mathbb{S}^{n-1} \cap \mathbf{s}^\perp$. Since the vectors $\tilde{\mathbf{a}}^{(i)}$ are independent and uniform over the same $n - 1$ dimensional subspace and rotations preserve inner product, we know their inner product is distributed as the inner product of two random vectors that are uniform on $\mathbb{S}^{n-2}$. Using the same argument as the null case, we get that

$$|(\tilde{\mathbf{a}}^{(1)})^\top \tilde{\mathbf{a}}^{(2)}| \geq \sqrt{\frac{2k}{n-1}},$$

with probability at most $4 \exp(-k)$. For the first term, we have $|\delta_1 \delta_2| \geq \frac{t}{n}$ with probability at least $\epsilon + 8 \exp(-k)$, as this occurs when the pancake alignment adversary succeeds on two independent runs. Therefore, with probability more than $\epsilon + 4 \exp(-k)$,

$$|(\mathbf{u}^{(1)})^\top \mathbf{u}^{(2)}| \geq \frac{t}{n} - \sqrt{\frac{2k}{n-1}},$$

which is at least $\sqrt{\frac{2k}{n}}$ due to our lower bound on t . Therefore, $\mathcal{B}$'s distinguishing advantage is at least ϵ , contradicting the assumed Gaussian pancake hardness. $\square$

7.2.2 Reducing low-dimensional PA to high-dimensional PA

In the prior section, we get a $\sqrt{n}$ term in t . Here, we present a way to decrease t when the dimension n is much larger than the security parameter λ , by showing that the hardness of the lower dimension version implies the hardness of a higher dimension version with the a similar t . In doing so, we can instead set $t = \sqrt{\lambda\omega(\log \lambda)}$ even in $n > \lambda$ dimensions.

Lemma 7.2. *Let $\beta = \beta(\lambda), \gamma = \gamma(\lambda), t = t(\lambda)$ be parameters, and let $\epsilon = \epsilon(\lambda)$ be the advantage. There exists some t^* satisfying*

$$t^* = O\left(t\left(\frac{\lambda}{t + \lambda}\right)\right),$$

such that if there is no PPT adversary that solves t^* -PA($\lambda, 1, \beta, \gamma$) with advantage ϵ , then for any $n = \text{poly}(\lambda) \geq \lambda$, there is no PPT adversary that solves t -PA($n, 1, \beta, \gamma$) with advantage $\epsilon + \text{negl}(\lambda)$.

Proof. For contradiction, say there is an $n \geq \lambda$ and an adversary $\mathcal{A}$ that solves the n -dimensional problem. Then we construct a PPT adversary $\mathcal{B}^{\text{GP}(\mathbf{s}, \cdot, \cdot)}$ for $\mathbf{s} \in \mathbb{S}^{\lambda-1}$ as follows. $\mathcal{B}$ first chooses an arbitrary $\mathbf{U} \in V_{\lambda, n}$ and samples a random $\mathbf{R} \leftarrow \text{SO}(n)$. It implements an oracle $\text{GP}(\mathbf{R}\mathbf{U}\mathbf{s}, \mathbf{u} \in \mathbb{R}^n, z \in [0, 1])$ by doing

- Sample $\mathbf{g} \leftarrow D_{\mathbb{R}^{n-\lambda}, 1}$.
- Sample $\mathbf{y} \leftarrow \text{GP}(\mathbf{s}, \mathbf{U}^\top \mathbf{R}^\top \mathbf{u}, z)$ using its oracle.
- Output $(\mathbf{R}(\mathbf{U}\mathbf{y} + \mathbf{U}^\perp \mathbf{g}))$.

$\mathcal{B}$ then simulates $\mathbf{v} \leftarrow \mathcal{A}^{\text{GP}(\mathbf{R}\mathbf{U}\mathbf{s}, z, \cdot)}(1^\lambda)$ and outputs $\frac{\mathbf{U}^\top \mathbf{R}^\top \mathbf{v}}{\|\mathbf{U}^\top \mathbf{R}^\top \mathbf{v}\|_2}$.

We see that $\mathcal{B}$ perfectly simulates the distribution of $\mathcal{A}^{\text{GP}(\mathbf{s}', \cdot, \cdot)}$ for $\mathbf{s}' = \mathbf{R}\mathbf{U}\mathbf{s} \leftarrow \mathbb{S}^{n-1}$ (by the same argument as in the proof of Lemma 8.4). Therefore,

$$\alpha^2 := (\mathbf{s}^\top (\mathbf{U}^\top \mathbf{R}^\top \mathbf{v}))^2 \geq \frac{t}{n}$$

with probability at least $\epsilon + \text{negl}(\lambda)$. Recall we can rewrite

$$\mathbf{v} = \sqrt{1 - \alpha^2} \mathbf{a} + \alpha (\mathbf{R}\mathbf{U}\mathbf{s})$$

where $\mathbf{a} \in (\mathbf{R}\mathbf{U}\mathbf{s})^\perp$. When we un-rotate $\mathbf{v}$, we get

$$\mathbf{R}^\top \mathbf{v} = \sqrt{1 - \alpha^2} \mathbf{a}' + \alpha (\mathbf{U}\mathbf{s})$$

where $\mathbf{a}' \leftarrow \mathbb{S}^{n-1} \cap (\mathbf{U}\mathbf{s})^\perp$, by Lemma 7.4. Then, when we project, we get

$$\mathbf{U}^\top \mathbf{R}^\top \mathbf{v} = \sqrt{1 - \alpha^2} \mathbf{U}^\top \mathbf{a}' + \alpha \mathbf{s}$$

where the norm satisfies

$$\|\mathbf{U}^\top \mathbf{R}^\top \mathbf{v}\|_2^2 = (1 - \alpha^2) \|\mathbf{U}^\top \mathbf{a}'\|_2^2 + \alpha^2,$$

By Lemma 7.6, $\|\mathbf{U}^\top \mathbf{a}'\|_2^2 \stackrel{d}{=} \text{Beta}(\frac{\lambda-1}{2}, \frac{n-\lambda}{2})$, which has mean $\frac{\lambda-1}{n-1}$.

Using standard concentration results [Sko23, Theorem 1], $\|\mathbf{U}^\top \mathbf{a}'\|_2^2 \leq 4 \frac{\lambda-1}{n-1}$ with probability at least $1 - \text{negl}(\lambda)$. Denote $r := 4 \frac{\lambda-1}{n-1}$. Conditioned on this and $\alpha^2 \geq t/n$, we have,

$$\begin{aligned} \frac{(\mathbf{s}^\top \mathbf{U}^\top \mathbf{R}^\top \mathbf{v})^2}{\|\mathbf{U}^\top \mathbf{R}^\top \mathbf{v}\|_2^2} &\geq \frac{\alpha^2}{r + \alpha^2(1 - r)} \\ &\geq \frac{1}{1 + \frac{r}{\alpha^2}} \\ &\geq \frac{1}{1 + r \frac{n}{t}} \end{aligned}$$

Therefore, if we set the approximation t^* such that the above alignment is always larger:

$$\frac{1}{1 + r \frac{n}{t}} \geq \frac{t^*}{\lambda} \iff t^* \leq t \left(\frac{\lambda}{t + rn} \right) = \Theta \left(\frac{t\lambda}{t + \lambda} \right)$$

then $\mathcal{B}$ succeeds with probability at least ϵ on *any* $\mathbf{s}$. $\square$

7.2.3 Reducing single-secret PA to multi-secret PA

Lemma 7.3. *Let $n = n(\lambda)$ be the dimension and $\ell = \ell(\lambda)$ with $n - \ell \geq \lambda$ and $\ell = o(n)$ be the number of secrets. Let $\beta = \beta(\lambda), \gamma = \gamma(\lambda), t = t(\lambda)$ be parameters and let $\epsilon = \epsilon(\lambda)$ be the advantage. There exists $t^* = t(1 - o(1))$ such that if no PPT adversary can solve t^* -PA($n - \ell + 1, 1, \beta, \gamma$) with advantage ϵ , then no PPT adversary can solve t -PA(n, ℓ, β, γ) with advantage $\ell\epsilon$.*

Proof. For the sake of contradiction, suppose there exists an adversary $\mathcal{A}$ that solves the multi-secret problem; that is, with probability $\ell\epsilon$ over $\mathbf{S} \leftarrow V_{\ell, n}$ and $\mathbf{v} \leftarrow \mathcal{A}^{\text{GP}(\mathbf{S}, \cdot, \cdot)}(1^\lambda)$,

$$\sum_{i=1}^{\ell} (\mathbf{S}[i]^\top \mathbf{v})^2 \geq t \frac{\ell}{n}.$$

Denote $n' := n - \ell + 1$. We construct an adversary $\mathcal{B}^{\text{GP}(\mathbf{s}, \cdot, \cdot)}$ for the lower-dimensional 1-secret problem. At the beginning, $\mathcal{B}$ samples $\mathbf{R} \leftarrow \text{SO}(n)$ and $j \leftarrow [\ell]$. Given inputs $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{z} \in [0, 1]^\ell$, $\mathcal{B}$ simulates a query to $\text{GP}(\mathbf{S}, \mathbf{u}, \mathbf{z})$ as follows:

- Let $\mathbf{u}' = \mathbf{R}^\top \mathbf{u}$.
- Sample $\mathbf{y}_1 \leftarrow \text{GP}(\mathbf{I}_{j-1}, \mathbf{u}'[1 : j-1], \mathbf{z}[1 : j-1]) \in \mathbb{R}^{j-1}$.
- Sample $\mathbf{y}_2 \leftarrow \text{GP}(\mathbf{s}, \mathbf{u}'[j : n' + j - 1], z_j) \in \mathbb{R}^{n'}$ using the oracle.
- Sample $\mathbf{y}_3 \leftarrow \text{GP}(\mathbf{I}_{\ell-j}, \mathbf{u}'[n' + j : n], \mathbf{z}[j + 1 : \ell]) \in \mathbb{R}^{\ell-j}$.
- Let $\mathbf{y}^* = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix}$, and return $\mathbf{R}\mathbf{y}^*$.

$\mathcal{B}$ then runs $\mathbf{v} \leftarrow \mathcal{A}^{O(\cdot, \cdot)}(1^\lambda)$, where O is the oracle described above, and outputs $\frac{(\mathbf{R}^\top \mathbf{v})[j : n' + j - 1]}{\|(\mathbf{R}^\top \mathbf{v})[j : n' + j - 1]\|_2}$.

Conditioned on j , on each oracle call, the sample $\mathbf{y}^*$ before rotation is distributed as a sample from $\text{GP}(\mathbf{S}^*, \mathbf{R}^\top \mathbf{u}, \mathbf{z})$, where

$$\mathbf{S}^* = \begin{bmatrix} \mathbf{I}_{j-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{s} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{\ell-j} \end{bmatrix}.$$

The columns of $\mathbf{S}^*$ are orthonormal. Therefore, after applying the random rotation, $\mathbf{R}\mathbf{y}^*$ is distributed as $\text{GP}(\mathbf{S}, \mathbf{u}, \mathbf{z})$ where $\mathbf{S} = \mathbf{R}\mathbf{S}^*$ is uniformly distributed in $V_{\ell, n}$. Conditioned on j , $\mathbf{v}$ is then distributed exactly as if it were generated by $\mathcal{A}$ with an oracle $\text{GP}(\mathbf{S}, \cdot, \cdot)$ for a uniformly random secret.

Now define the events

$$E_i = \left\{ (\mathbf{S}[i]^\top \mathbf{v})^2 \geq \frac{t}{n} \right\}.$$

If $\mathcal{B}$ picks j and E_j occurs, then we can see that $\mathcal{B}$ will succeed with the approximation $t^* := \frac{tn'}{n}$. This is because

$$\begin{aligned} (\mathbf{s}^\top (\mathbf{R}^\top \mathbf{v}[j : n' + j - 1]))^2 &= (\mathbf{S}^*[j]^\top (\mathbf{R}^\top \mathbf{v}))^2 \\ &= (\mathbf{S}[j]^\top \mathbf{v})^2 \\ &\geq \frac{t}{n} = \frac{tn'}{n} \end{aligned}$$

So $\mathcal{B}$ succeeds with approximation factor $t^* = \frac{t(n-\ell+1)}{n} = t(1 - o(1))$ when $\ell = o(n)$. Note this also true of the normalized vector, since $\|(\mathbf{R}^\top \mathbf{v})[j : n' + j - 1]\|_2 \leq 1$.

It remains to bound the probability of this event when j is chosen uniformly from $[\ell]$. By assumption, $\|\mathbf{S}^\top \mathbf{v}\|_2^2 \geq t \frac{\ell}{n}$ with probability $\ell\epsilon$, which implies that at least one E_i must occur with this probability. Hence, $\Pr\left[\bigcup_{i=1}^{\ell} E_i\right] \geq \ell\epsilon$. Since $\mathcal{B}$ chooses j uniformly from $[\ell]$, its success probability is $\frac{1}{\ell} \sum_{i=1}^{\ell} \Pr[E_i] \geq \frac{1}{\ell} \Pr\left[\bigcup_{i=1}^{\ell} E_i\right] \geq \epsilon$, contradicting the assumed hardness. $\square$

7.2.4 Facts about the uniform distribution on the sphere restricted to a subspace

Lemma 7.4. *Let $\ell, n \in \mathbb{N}$ with $\ell < n$ and $\mathbf{S} \in V_{\ell, n}$ be fixed. Let $\mathbf{R} \leftarrow \text{SO}(n)$ be a uniformly random rotation. Let $\mathbf{a} \in \mathbb{R}^n$ be a random vector that can be generated as $\mathbf{a} \leftarrow \mathcal{A}(\mathbf{RS})$ by some (potentially unbounded) adversary $\mathcal{A}$. Then if we project $\mathbf{a}$ to the subspace orthogonal to $\mathbf{RS}$ and un-rotate it,*

$$\mathbf{R}^\top \Pi_{(\mathbf{RS})^\perp} \mathbf{a},$$

its direction is distributed uniformly over $\mathbb{S}^{n-1} \cap \mathbf{S}^\perp$ (conditioned on this vector being non-zero).

Proof. Since $\mathbf{S}$ has orthonormal columns and $\mathbf{R}$ is orthogonal, $\mathbf{RS}$ also has orthonormal columns, so $\Pi_{(\mathbf{RS})^\perp} = \mathbf{I} - (\mathbf{RS})(\mathbf{RS})^\top$. Let $\mathbf{Q} \in \text{SO}(n)$ be any rotation satisfying $\mathbf{QS} = \mathbf{S}$. We know that

$$(\mathbf{RQ})\mathbf{S} = \mathbf{R}(\mathbf{QS}) = \mathbf{RS}.$$

We have the deterministic identity

$$\mathbf{Q}^\top \mathbf{R}^\top (\mathbf{I} - (\mathbf{RS})(\mathbf{RS})^\top) \mathcal{A}(\mathbf{RS}) = (\mathbf{RQ})^\top (\mathbf{I} - (\mathbf{RQS})(\mathbf{RQS})^\top) \mathcal{A}(\mathbf{RQS}).$$

Since $\mathbf{R}$ is uniformly (Haar) random, $\mathbf{RQ} \stackrel{d}{=} \mathbf{R}$, so we have

$$\mathbf{Q}^\top \mathbf{R}^\top (\mathbf{I} - (\mathbf{RS})(\mathbf{RS})^\top) \mathbf{a} \stackrel{d}{=} \mathbf{R}^\top (\mathbf{I} - (\mathbf{RS})(\mathbf{RS})^\top) \mathbf{a}.$$

Since this holds for every $\mathbf{Q} \in \text{SO}(n)$ with $\mathbf{QS} = \mathbf{S}$ and the same invariance holds for the direction of $\mathbf{R}^\top (\mathbf{I} - (\mathbf{RS})(\mathbf{RS})^\top) \mathbf{a}$, then by Lemma 7.5,

$$\frac{\mathbf{R}^\top (\mathbf{I} - (\mathbf{RS})(\mathbf{RS})^\top) \mathbf{a}}{\|\mathbf{R}^\top (\mathbf{I} - (\mathbf{RS})(\mathbf{RS})^\top) \mathbf{a}\|_2} \leftarrow \mathbb{S}^{n-1} \cap \mathbf{S}^\perp.$$

$\square$

Lemma 7.5. *Let $\ell, n \in \mathbb{N}$ with $\ell < n$, let $\mathbf{S} \in V_{\ell, n}$ be fixed, and let $\mathbf{a} \in \mathbb{S}^{n-1} \cap \mathbf{S}^\perp$ be random. If, for every $\mathbf{Q} \in \text{SO}(n)$ satisfying $\mathbf{QS} = \mathbf{S}$, $\mathbf{Qa} \stackrel{d}{=} \mathbf{a}$, then $\mathbf{a}$ is distributed uniformly over $\mathbb{S}^{n-1} \cap \mathbf{S}^\perp$.*

Proof. Let $\mathbf{M} \in \text{SO}(n)$ be a change of coordinates such that

$$\mathbf{M}(\mathbf{S}^\perp) = \text{span}(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n-\ell)}),$$

where $\mathbf{b}^{(i)}$ is the i th canonical basis vector. Consider the pushforward distribution $\mathbf{Ma}$. For every $\mathbf{Q}$ as above,

$$(\mathbf{MQM}^\top)(\mathbf{Ma}) = \mathbf{M}(\mathbf{Qa}) \stackrel{d}{=} \mathbf{Ma}.$$

So $\mathbf{Ma}$ is invariant under every rotation of the form $\mathbf{MQM}^\top$. Since $\mathbf{Q}$ fixes $\mathbf{S}$ pointwise and acts arbitrarily on $\mathbf{S}^\perp$,

$$\mathbf{MQM}^\top = \begin{bmatrix} \mathbf{P} & 0 \\ 0 & \mathbf{I}_\ell \end{bmatrix}$$

for some $\mathbf{P} \in \text{SO}(n - \ell)$, i.e. $\mathbf{Q}$ in $\mathbf{M}$'s coordinate system acts only on $\mathbf{MS}^\perp = \text{span}(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n-\ell)})$. Thus $\mathbf{Ma}$, which is only supported on the first $n - \ell$ coordinates, is invariant under every rotation acting on those coordinates. Identifying

$$\mathbb{S}^{n-1} \cap \text{span}(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n-\ell)}) \cong \mathbb{S}^{n-\ell-1},$$

this means $\mathbf{M}\mathbf{a}$ is invariant under $\text{SO}(n - \ell)$ acting on $\mathbb{S}^{n-\ell-1}$. By uniqueness of the Haar measure, the distribution of $\mathbf{M}\mathbf{a}$ must be uniform over $\mathbb{S}^{n-1} \cap \text{span}(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n-\ell)})$. It follows that $\mathbf{a} = \mathbf{M}^\top(\mathbf{M}\mathbf{a})$ is uniformly distributed over

$$\mathbf{M}^\top \left(\mathbb{S}^{n-1} \cap \text{span}(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n-\ell)}) \right) = \mathbb{S}^{n-1} \cap \mathbf{S}^\perp.$$

□

Lemma 7.6. *Let $\lambda, n \in \mathbb{N}$ such that $1 < \lambda < n$, let $\mathbf{x} \in \mathbb{S}^{n-1}$ be fixed, and let $\mathbf{a} \leftarrow \mathbb{S}^{n-1} \cap \mathbf{x}^\perp$. Let $W \subseteq \mathbb{R}^n$ be a λ -dimensional subspace containing $\mathbf{x}$. Then*

$$\|\Pi_W \mathbf{a}\|_2^2 \stackrel{d}{=} \text{Beta}\left(\frac{\lambda-1}{2}, \frac{n-\lambda}{2}\right),$$

where Π_W denotes the orthogonal projection onto W .

Proof. Since $\mathbf{a} \in \mathbf{x}^\perp$,

$$\Pi_W \mathbf{a} = \Pi_{W \cap \mathbf{x}^\perp} \mathbf{a}.$$

The subspace $\mathbf{x}^\perp$ has dimension $n - 1$, and

$$\dim(W \cap \mathbf{x}^\perp) = \lambda - 1,$$

since $\mathbf{x} \in W$. Since changing the coordinate system preserves Euclidean norms, we may choose a coordinate system adapted to $\mathbf{x}$ and W . Namely, let the final basis vector be $\mathbf{x}$ and choose the first $\lambda - 1$ basis vectors to form an orthonormal basis for $W \cap \mathbf{x}^\perp$. These choices are compatible since $W \cap \mathbf{x}^\perp$ is orthogonal to $\mathbf{x}$. Completing these vectors to an orthonormal basis $\{\mathbf{d}^{(i)}\}$ gives a coordinate system in which

$$\mathbf{x}^\perp = \text{span}(\mathbf{d}^{(1)}, \dots, \mathbf{d}^{(n-1)})$$

and

$$W \cap \mathbf{x}^\perp = \text{span}(\mathbf{d}^{(1)}, \dots, \mathbf{d}^{(\lambda-1)}).$$

Therefore, in this coordinate system, we can express

$$\mathbf{a} = [a_1, \dots, a_{n-1}, 0]^\top,$$

where $[a_1, \dots, a_{n-1}]$ is uniformly distributed on $\mathbb{S}^{n-2}$, and we have

$$\|\Pi_W \mathbf{a}\|_2^2 = \sum_{i=1}^{\lambda-1} a_i^2,$$

which is distributed as

$$\|\Pi_W \mathbf{a}\|_2^2 \stackrel{d}{=} \text{Beta}\left(\frac{\lambda-1}{2}, \frac{n-\lambda}{2}\right).$$

□

7.3 Reduction from KDS Gaussian Pancake for $t = \frac{1}{\text{advantage}}$

Here, we give the proof of Theorem 4.2. This reduction is particularly interesting when we only need to rule out adversaries that succeed with high probability. The idea behind this reduction is that if an adversary can produce a vector $\mathbf{v}$ that is often biased toward the secret direction, then repeatedly un-rotating the outputs of $\mathcal{A}^{\text{GP}(\mathbf{R}\mathbf{s}, \cdot)}$ for random rotations $\mathbf{R}$ should yield vectors that remain consistently biased toward $\mathbf{s}$. In contrast, repeatedly un-rotating the outputs of $\mathcal{A}^{\text{Null}(\cdot, \cdot)}$ should produce vectors whose directions are uniformly distributed. We leverage this gap to construct a distinguisher. Notably, this reduction only requires the relatively weak assumption that the GP problem is hard for almost all instances.

Proposition 7.3. Let $n = n(\lambda) \geq \lambda$ be the dimension, and let $\beta = \beta(\lambda), \gamma = \gamma(\lambda)$ be parameters. Assume no PPT adversary can solve $\text{GP}_{\text{shift}}(n, 1, \beta, \gamma)$ with advantage $1 - \text{negl}(\lambda)$. Then for any $\epsilon = \epsilon(\lambda)$, there exists

$$t^* = \frac{1}{\epsilon}(1 + o(1))$$

such for any $t = t(\lambda) \geq t^*$, no PPT adversary can solve $t\text{-PA}(n, 1, \beta, \gamma)$ with advantage ϵ .

Proof. For contradiction, assume there is some PPT adversary $\mathcal{A}$ such that for $\mathbf{s} \leftarrow \mathbb{S}^{n-1}, \mathbf{v} \leftarrow \mathcal{A}^{\text{GP}(\mathbf{s}, z, \cdot)}(1^\lambda)$ satisfies

$$\Pr[(\mathbf{v}^\top \mathbf{s})^2 \geq t/n] = \epsilon, \quad \text{where } \mathbf{v} \in \mathbb{S}^{n-1}.$$

Then we can construct an adversary $\mathcal{B}^{O(\cdot, \cdot)}$ against the Gaussian pancake distinguishing problem. Let $m = \omega(n^3)$. $\mathcal{B}^{O(\cdot, \cdot)}(1^\lambda)$ does as follows:

- For $i \in [m]$ sample a uniformly random rotation $\mathbf{R}_i \leftarrow \text{SO}(n)$.
- For $i \in [m]$, run $\mathbf{v}^{(i)} \leftarrow \mathcal{A}^{\mathbf{R}_i O(\mathbf{R}_i^\top \cdot, \cdot)}(1^\lambda)$ independently with fresh randomness.
- Compute the empirical covariance matrix $\hat{\Sigma} = \frac{1}{m} \sum_{i=1}^m \mathbf{R}_i^\top \mathbf{v}^{(i)} (\mathbf{v}^{(i)})^\top \mathbf{R}_i$.
- Compute the eigenvalues $\hat{\lambda}_1, \dots, \hat{\lambda}_n$ of $\hat{\Sigma}$.
- Output 1 iff there is a pair of eigenvalues $\hat{\lambda}_i, \hat{\lambda}_j$ with $i \neq j$ such that

$$|\hat{\lambda}_i - \hat{\lambda}_j| > 4\sqrt{\frac{n}{m}}.$$

Recall the oracle $\mathbf{R}_i O(\mathbf{R}_i^\top \cdot, \cdot)$ is the oracle that rotates the first input and the output of the sample from O , as in the proof of Lemma 7.1. Let $\mathbf{R} \leftarrow \text{SO}(n), \mathbf{v} \leftarrow \mathcal{A}^{\mathbf{R} O(\mathbf{R}^\top \cdot, \cdot)}(1^\lambda)$ be random variables. Consider the two cases.

Null case: When the oracle $O(\cdot, \cdot)$ ignores its inputs and outputs standard Gaussian vectors, each query to $\mathbf{R} O(\mathbf{R}^\top \cdot, \cdot)$ returns $\mathbf{R}\mathbf{g}$ where $\mathbf{g} \leftarrow D_{\mathbb{R}^n, 1}$. As in the proof of the previous reduction, this is distributed exactly the same as a query to $O(\cdot, \cdot)$. We can analyze the covariance matrix for the random variable $\mathbf{u} := \mathbf{R}^\top \mathbf{v}$,

$$\begin{aligned} \mathbb{E}[\mathbf{u}\mathbf{u}^\top] &= \mathbb{E}_{\mathbf{R}, \mathcal{A}, O}[\mathbf{R}^\top \mathcal{A}^{\mathbf{R} O(\mathbf{R}^\top \cdot, \cdot)}(1^\lambda) \mathcal{A}^{\mathbf{R} O(\mathbf{R}^\top \cdot, \cdot)}(1^\lambda)^\top \mathbf{R}] \\ &= \mathbb{E}[\mathbf{R}^\top \mathcal{A}^{O(\cdot, \cdot)}(1^\lambda) \mathcal{A}^{O(\cdot, \cdot)}(1^\lambda)^\top \mathbf{R}] \\ &= \mathbb{E}_{\mathbf{s} \leftarrow \mathbb{S}^{n-1}}[\mathbf{s}\mathbf{s}^\top] \end{aligned}$$

which has all eigenvalues equal, $\lambda_1 = \lambda_2 = \dots = \lambda_n = \frac{1}{n}$. Since the random variables $\mathbf{R}_i^\top \mathbf{v}^{(i)}$ are independent copies of $\mathbf{u}$, Corollary 7.7.2 tells us that,

$$|\hat{\lambda}_i - \lambda_i| \leq 2\sqrt{\frac{n}{m}}$$

for all $i \in [n]$ simultaneously with probability $1 - 2ne^{-n/2}$. Therefore, the absolute difference between any two $\hat{\lambda}_i, \hat{\lambda}_j$ is at most $4\sqrt{\frac{n}{m}}$, meaning $\mathcal{B}$ will output 0 with probability at least $1 - 2ne^{-n/2}$.

Pancake case: Consider a *fixed* $\mathbf{s} \in \mathbb{S}^{n-1}$, and consider the oracle $O(\cdot, \cdot)$ that outputs Gaussian pancake vectors with secret $\mathbf{s}$. Then $\mathbf{R} O(\mathbf{R}^\top \cdot, \cdot)$ has the same distribution as an oracle for $\text{GP}(\mathbf{R}\mathbf{s}, \cdot, \cdot)$. Recall from the proof of Lemma 7.1 we can rewrite

$$\mathbf{u} = \mathbf{R}^\top \mathbf{v} = \sqrt{1 - \delta^2} \tilde{\mathbf{a}} + \delta \mathbf{s}$$

where $\delta := \mathbf{s}^\top \mathbf{u}$, and the direction $\tilde{\mathbf{a}}$ is distributed as uniform over $\mathbb{S}^{n-1} \cap \mathbf{s}^\perp$ independent of δ , so $\sqrt{1 - \delta^2} \tilde{\mathbf{a}}$ has mean $\mathbf{0}$. This tells us that

$$\mathbb{E}_{\mathbf{R}, \mathcal{A}, O}[\mathbf{u}\mathbf{u}^\top] = \mathbb{E}[(1 - \delta^2)\tilde{\mathbf{a}}\tilde{\mathbf{a}}^\top] + \mathbb{E}[\delta^2]\mathbf{s}\mathbf{s}^\top = \frac{1 - \mathbb{E}[\delta^2]}{n-1}(\mathbf{I} - \mathbf{s}\mathbf{s}^\top) + \mathbb{E}[\delta^2]\mathbf{s}\mathbf{s}^\top,$$

The coefficient $\frac{1 - \mathbb{E}[\delta^2]}{n-1}$ follows from the fact that $\sqrt{1 - \delta^2} \tilde{\mathbf{a}}$ is rotationally invariant over the subspace $\mathbf{s}^\perp$. Therefore, the covariance matrix restricted to $\mathbf{s}^\perp$ is isotropic, with all $n - 1$ eigenvalues being equal. Since,

$$\text{tr}(\mathbb{E}[(1 - \mathbb{E}[\delta^2])\tilde{\mathbf{a}}\tilde{\mathbf{a}}^\top]) = 1 - \mathbb{E}[\delta^2],$$

each eigenvalue in the orthogonal subspace is $\frac{1 - \mathbb{E}[\delta^2]}{n-1}$.

WLOG, let $\hat{\lambda}_1$ denote the eigenvalue corresponding to $\mathbb{E}[\delta^2]$. By Corollary 7.7.2, with probability at least $1 - 2ne^{-n/2}$, $\hat{\lambda}_1$ is within absolute distance $2\sqrt{\frac{n}{m}}$ of $\mathbb{E}[\delta^2]$ and $\hat{\lambda}_2$ is within absolute distance $2\sqrt{\frac{n}{m}}$ of $\frac{1 - \mathbb{E}[\delta^2]}{n-1}$. Let β_1, β_2 denote their errors respectively. Conditioned on this event, we have,

$$\begin{aligned} \hat{\lambda}_1 - \hat{\lambda}_2 &\geq \frac{(n-1)\mathbb{E}[\delta^2] - (1 - \mathbb{E}[\delta^2])}{n-1} + \beta_1 - \beta_2 \\ &\geq \frac{n\mathbb{E}[\delta^2] - 1}{n-1} - 4\sqrt{\frac{n}{m}} \\ &\geq \frac{t\epsilon - 1}{n-1} - 4\sqrt{\frac{n}{m}}, \end{aligned}$$

where we use the fact that $\mathbb{E}[\delta^2] = \mathbb{E}[(\mathbf{R}\mathbf{s})^\top \mathcal{A}^{\text{GP}(\mathbf{R}\mathbf{s}, \cdot, \cdot)}]^2 \geq \frac{t}{n}(\Pr[\mathcal{A} \text{ succeeds}]) \geq t\epsilon/n$. Now, so long as

$$t > \frac{1}{\epsilon} \left(1 + 8(n-1)\sqrt{\frac{n}{m}} \right) = \frac{1}{\epsilon}(1 + o(1)),$$

the difference between $\hat{\lambda}_1$ and $\hat{\lambda}_2$ is more than $4\sqrt{\frac{n}{m}}$, meaning $\mathcal{B}$ will output 1 with probability at least $1 - 2ne^{-n/2}$. Because this holds for every secret $\mathbf{s} \in \mathbb{S}^{n-1}$, it holds on average over $\mathbf{s} \leftarrow \mathbb{S}^{n-1}$. $\square$

7.3.1 Covariance estimation

In prior section, we use the following facts that allow us to estimate the covariance of random unit vectors. For self-adjoint matrices $\mathbf{A}$ and $\mathbf{B}$, we write $\mathbf{A} \preceq \mathbf{B}$ to denote that $\mathbf{B} - \mathbf{A}$ is positive semidefinite. We write $\lambda_i(\mathbf{A})$ for the i th eigenvalue of $\mathbf{A}$.

Lemma 7.7 ([Tro12] Matrix Hoeffding). *Let $\mathbf{X}_1, \dots, \mathbf{X}_m$ be random self-adjoint matrices with dimension n , and let $\mathbf{A}_1, \dots, \mathbf{A}_m$ be fixed self-adjoint matrices with dimension n . Assume each $\mathbf{X}_i$ satisfies*

$$\mathbb{E}[\mathbf{X}_i] = \mathbf{0} \text{ and } \mathbf{X}_i^2 \preceq \mathbf{A}_i^2 \text{ almost surely}$$

Then for all $t \geq 0$,

$$\Pr[\lambda_{\max}(\sum_{i=1}^m \mathbf{X}_i) \geq t] \leq n \exp(-\frac{t^2}{8\sigma^2})$$

where $\sigma^2 = \|m\mathbf{A}^2\|_{\text{op}}$.

Fact 7.7.1. *Let $\mathbf{u}$ be a random vector from a distribution over $\mathbb{S}^{n-1}$. Then almost surely,*

$$-\mathbf{I} \preceq \mathbf{u}\mathbf{u}^\top - \mathbb{E}[\mathbf{u}\mathbf{u}^\top] \preceq \mathbf{I}$$

Proof. $\mathbf{u}$ is supported on unit vectors, so $0 \preceq \mathbf{u}\mathbf{u}^\top \preceq \mathbf{I}$ with probability 1, which implies the same about its expectation, $0 \preceq \mathbb{E}[\mathbf{u}\mathbf{u}^\top] \preceq \mathbf{I}$. $\square$

Setting $\mathbf{X}_i = \pm(\mathbf{u}^{(i)}(\mathbf{u}^{(i)})^\top - \mathbb{E}[\mathbf{u}\mathbf{u}^\top])$ and $\mathbf{A}_i = \mathbf{I}$, we get the following.

Corollary 7.7.1. Let $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(m)}$ be i.i.d. from a distribution $\mathcal{U}$ over the unit sphere. Then

$$\Pr\left[\left\|\sum_{i=1}^m \mathbf{u}^{(i)}(\mathbf{u}^{(i)})^\top - m\mathbb{E}_{\mathbf{u} \leftarrow \mathcal{U}}[\mathbf{u}\mathbf{u}^\top]\right\|_{op} \leq t\right] \geq 1 - 2n \exp\left(-\frac{t^2}{8m}\right)$$

Proof. We use the fact that for a symmetric matrix $\mathbf{x} = \sum_{i=1}^m \mathbf{u}^{(i)}(\mathbf{u}^{(i)})^\top - m\mathbb{E}_{\mathbf{u} \leftarrow \mathcal{U}}[\mathbf{u}\mathbf{u}^\top]$, $\|\mathbf{x}\|_{op} = \max_i |\lambda_i|$. Since $\lambda_{\max}(\mathbf{x})$ and $\lambda_{\max}(-\mathbf{x})$ are simultaneously smaller than t with the stated probability, it follows that the operator norm is bounded by t with the same probability. $\square$

Corollary 7.7.2. Let $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(m)}$ be i.i.d. from a distribution $\mathcal{U}$ over the unit sphere. Let $\epsilon \in (0, 1)$. Then with probability at least $1 - \epsilon$, for every $i \in [n]$,

$$\left|\lambda_i\left(\frac{1}{m} \sum_{i=1}^m \mathbf{u}^{(i)}(\mathbf{u}^{(i)})^\top\right) - \lambda_i(\mathbb{E}[\mathbf{u}\mathbf{u}^\top])\right| \leq 2\sqrt{\frac{2 \ln(\frac{2n}{\epsilon})}{m}}$$

Proof. Denote $\mathbf{A} := \frac{1}{m} \sum_{i=1}^m \mathbf{u}^{(i)}(\mathbf{u}^{(i)})^\top$ and $\mathbf{B} := \mathbb{E}[\mathbf{u}\mathbf{u}^\top]$. From Corollary 7.7.1,

$$\|m\mathbf{A} - m\mathbf{B}\|_{op} \leq 2\sqrt{2m \ln(\frac{2n}{\epsilon})}$$

with probability at least $1 - \epsilon$. Therefore, the operator norm between $\mathbf{A}$ and $\mathbf{B}$ is at most $\frac{2}{\sqrt{m}}\sqrt{2 \ln(\frac{2n}{\epsilon})}$. The result follows from Weyl's inequality, which states the absolute difference between corresponding eigenvalues of $\mathbf{A}$ and $\mathbf{B}$ is upper bounded by the operator norm of their difference. $\square$

8 Hardness of Key-Dependent Shifted Gaussian Pancake

Here, we recall the proposition, that the multi-secret, KDS Gaussian pancake distinguishing problem is hard assuming CLWE with normalized binary secrets is hard. We prove the statement through a series of auxiliary lemmas.

Proposition 8.1. Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\gamma = \gamma(\lambda) \geq \sqrt{3\lambda\omega(\log \lambda)}$ and $\beta = 1/\text{poly}(\lambda)$ be parameters. Assume no PPT adversary solves $\text{CLWE}(\lambda, \beta/\sqrt{2}, \gamma, U(\frac{1}{\sqrt{\lambda}}\{\pm 1\}^\lambda))$ with advantage $\delta = \delta(\lambda)$. Then for every $n = \text{poly}(\lambda) > \lambda$ and $\ell = \text{poly}(\lambda)$ such that $n - \ell \geq \lambda$, there is no PPT adversary that solves $\text{GP}_{\text{shift}}(n, \ell, \beta, \gamma)$ with advantage $\ell(\delta + \text{negl}(\lambda))$.

Proof. The condition $\gamma \geq \sqrt{3\lambda\omega(\log \lambda)}$ implies that, for every $m = \text{poly}(\lambda)$, $\gamma \geq \sqrt{\lambda(\ln m + \ln \lambda + \omega(\log \lambda))}$. Hence, by Lemma 8.1 with $\mathcal{S} := U(\{\pm 1\}^\lambda)$, no PPT adversary solves $\text{CLWE}_{\text{shift}}(\lambda, \beta/\sqrt{2}, \gamma, \frac{1}{\sqrt{\lambda}}\mathcal{S})$ with advantage $\delta + \text{negl}(\lambda)$. By Lemma 8.2, this implies that no PPT adversary solves $\text{CLWE}_{\text{shift}}(\lambda, \beta/\sqrt{2}, \gamma, U(\mathbb{S}^{\lambda-1}))$ with advantage $\delta + \text{negl}(\lambda)$. By Lemmas 8.3 and 8.4, and the fact that $n - \ell + 1 > \lambda$, no PPT adversary succeeds at $\text{GP}_{\text{shift}}(n - \ell + 1, 1, \beta, \gamma)$ with advantage $\delta + \text{negl}(\lambda)$. Finally, Lemma 8.5 implies that no PPT adversary solves $\text{GP}_{\text{shift}}(n, \ell, \beta, \gamma)$ with advantage $\ell(\delta + \text{negl}(\lambda))$. $\square$

8.1 CLWE with key-dependent shift

We define an intermediate problem, the decisional continuous LWE problem with a secret-dependent shift.

Definition 8.1 (CLWE with KDS distribution). Let $n \in \mathbb{N}$ and $\gamma, \beta \in \mathbb{R}_{>0}$. For a secret $\mathbf{s} \in \mathbb{R}^n$ and a shift $\mathbf{u} \in \mathbb{R}^n$, we define the $\mathbf{u}$ -shifted CLWE distribution $\mathbf{u}\text{-CLWE}_{\mathbf{s}, \text{shift}}(n, \beta, \gamma)$ as the distribution that generates samples

$$(\mathbf{y}, \gamma \mathbf{s}^\top \mathbf{y} + \gamma \mathbf{s}^\top \mathbf{u} + e \bmod 1),$$

where $\mathbf{y} \leftarrow D_{\mathbb{R}^n, 1}$ and $e \leftarrow D_{\mathbb{R}, \beta}$.

Definition 8.2 (Decisional CLWE with KDS). Let $n = n(\lambda)$, $\gamma = \gamma(\lambda)$, $\beta = \beta(\lambda)$ be parameter families, and let $W = \{W_\lambda\}$ be a family of distributions over $\mathbb{S}^{n-1}$. The CLWE with key-dependent shift problem asks to distinguish between an oracle $\text{CLWE}_{\mathbf{s}, \text{shift}}(\cdot)$ for a random $\mathbf{s} \leftarrow W_\lambda$, which takes as input $\mathbf{u} \in \mathbb{R}^n$ and returns a fresh sample from $\mathbf{u}\text{-CLWE}_{\mathbf{s}, \text{shift}}(n, \beta, \gamma)$, and a null oracle Null which ignores its input and outputs fresh null samples.

8.2 Reducing normalized binary secret CLWE to CLWE with KDS

The key idea behind the reduction (inspired by [GVV22]) is that a sufficiently wide Gaussian sample $\mathbf{y} \leftarrow D_{\mathbb{R}^\lambda, \tau}$ can be viewed as consisting of an integer part and a fractional part $\mathbf{a} \in [0, 1)^\lambda$, where the fractional part is uniformly random. Given such a sample, we can first compute its fractional part $\mathbf{a}$. We can then sample a fresh Gaussian whose fractional part is $\mathbf{a} - \mathbf{u} \pmod{1}$, which is still uniform. Thus, while in the discrete LWE analogue of this reduction, one subtracts $\mathbf{u}$ directly from the uniform sample, here we subtract $\mathbf{u}$ only from the fractional part of the Gaussian sample.

Lemma 8.1. *Let $\lambda \in \mathbb{N}$ be the security parameter (and dimension), let $m = \text{poly}(\lambda)$ be the number of samples, and let $\mathcal{S} = \{\mathcal{S}_\lambda\}$ be a family of distributions over $\mathbb{Z}^\lambda$ with fixed norm $r = r(\lambda)$. Assume no PPT adversary making m queries to its oracle solves $\text{CLWE}(\lambda, \beta, \gamma, \frac{1}{r}\mathcal{S})$ with advantage $\delta = \delta(\lambda)$, where $\beta = \beta(\lambda)$, $\gamma = \gamma(\lambda)$ are parameters satisfying*

$$\gamma \geq r \sqrt{\ln m + \ln \lambda + \omega(\log \lambda)}.$$

Then there is no PPT adversary making m queries to its oracle that solves $\text{CLWE}_{\text{shift}}(\lambda, \beta, \gamma, \frac{1}{r}\mathcal{S})$ with advantage $\delta + \text{negl}(\lambda)$.

Proof. Denote $\tau = \gamma/r$. We know $\tau \geq \sqrt{\ln m + \ln \lambda + \omega(\log \lambda)}$. We show that for any $\mathbf{s} \in \text{supp}(\frac{1}{r}\mathcal{S})$, one can simulate a single query to a $\text{CLWE}_{\mathbf{s}, \text{shift}}(\cdot)$ oracle using a single query to its $\text{CLWE}_{\mathbf{s}}$ oracle, incurring only $\text{negl}(\lambda)$ loss in advantage.

In particular, given input $\mathbf{u}^* \in \mathbb{R}^\lambda$, an adversary $\mathcal{A}^O$ can generate samples as follows:

- Sample $(\mathbf{y}, b) \leftarrow O$.
- Let $\mathbf{a} = (\tau \mathbf{y}) \pmod{1}$.
- Let $\mathbf{u} = \tau \mathbf{u}^*$.
- For $i \in [\lambda]$, sample $y'_i \leftarrow D_{\mathbb{Z} + (a_i - u_i \pmod{1}), \tau}$. Denote $\mathbf{y}' = (y'_1, \dots, y'_\lambda)$.
- Output $(\frac{\mathbf{y}'}{\tau}, b)$.

For the case where O is the $\text{CLWE}_{\mathbf{s}}$ oracle, we want to argue that $(\frac{\mathbf{y}'}{\tau}, b)$ is negligibly close to a $\mathbf{u}^*$ -shifted sample when $\mathbf{y}$ is standard Gaussian and $b = \gamma \mathbf{s}^\top \mathbf{y} + e \pmod{1}$. Note that

$$\mathbf{y}' = \mathbf{a} - \mathbf{u} = \tau \mathbf{y} - \tau \mathbf{u}^* \pmod{1},$$

which implies that

$$\tau \mathbf{y} = \mathbf{y}' + \tau \mathbf{u}^* \pmod{1}.$$

Since $r\mathbf{s} \in \mathbb{Z}^\lambda$ and $\gamma = r\tau$, $\mathcal{A}$'s simulated samples satisfy

$$\begin{aligned} \left(\frac{\mathbf{y}'}{\tau}, (r\tau)\mathbf{s}^\top \mathbf{y} + e \pmod{1}\right) &= \left(\frac{\mathbf{y}'}{\tau}, (r\mathbf{s})^\top (\tau \mathbf{y}) + e \pmod{1}\right) \\ &= \left(\frac{\mathbf{y}'}{\tau}, (r\mathbf{s})^\top (\mathbf{y}' + \tau \mathbf{u}^*) + e \pmod{1}\right) \\ &= \left(\frac{\mathbf{y}'}{\tau}, (r\tau)\mathbf{s}^\top \left(\frac{\mathbf{y}'}{\tau} + \mathbf{u}^*\right) + e \pmod{1}\right) \\ &= \left(\frac{\mathbf{y}'}{\tau}, \gamma \mathbf{s}^\top \left(\frac{\mathbf{y}'}{\tau}\right) + \gamma \mathbf{s}^\top \mathbf{u}^* + e \pmod{1}\right). \end{aligned}$$

It remains to show that $\mathbf{y}'$ looks like $D_{\mathbb{R}, \tau}$ when $\mathbf{y}$ is standard Gaussian, which implies that the above is distributed as a $\mathbf{u}^*$ -shifted $\text{CLWE}_{\mathbf{s}, \text{shift}}$ sample. From Lemma 3.7, for $i \in [\lambda]$, $a_i = (\tau y_i \pmod{1})$ is within $\text{negl}(\lambda)/(\lambda m)$ in statistical distance of uniform over $[0, 1)$. Therefore the shifted $(a_i - u_i \pmod{1})$ is also $\text{negl}(\lambda)/(\lambda m)$ close to uniform over $[0, 1)$. By the data processing and triangle inequalities, as well as Lemma 3.6,

$$y'_i \leftarrow D_{\mathbb{Z} + (a_i - u_i \pmod{1}), \tau}$$

is $2\text{negl}(\lambda)/(\lambda m)$ close to $D_{\mathbb{R},\tau}$. In total, there are λm coordinates (λ coordinates per query) sampled like this, so by triangle inequality the m generated samples are $2\text{negl}(\lambda)$ close to m $\text{CLWE}_{\mathbf{s},\text{shift}}$ samples. Note in the case where O is the null oracle, this also implies that the m generated samples are $2\text{negl}(\lambda)$ close to m null samples. Therefore, the advantage of $\mathcal{A}$ is at least the advantage of any $\text{CLWE}_{\text{shift}}(\lambda, \beta, \gamma \frac{1}{r} \mathcal{S})$ distinguisher making m queries (with a negligible loss). $\square$

8.2.1 Making secrets uniform over the unit sphere

In the prior section, we showed $\text{CLWE}_{\text{shift}}$ is hard when the secret distribution $\frac{1}{r} \mathcal{S}$ is discrete. In this section, we will show that this implies its hardness when secrets come uniformly from the unit sphere. This is a common idea from the literature [BRST21; GVV22], here we just apply it to our key-dependent shifted problem.

Lemma 8.2. *Let $\lambda \in \mathbb{N}$ be the security parameter (and dimension), and let $\mathcal{S}$ be a distribution over $\mathbb{Z}^\lambda$ with fixed norm $r = r(\lambda)$. Let $\gamma = \gamma(\lambda), \beta = \beta(\lambda)$ be parameters. If no PPT adversary can solve $\text{CLWE}_{\text{shift}}(\lambda, \beta, \gamma, \frac{1}{r} \mathcal{S})$ with advantage $\delta = \delta(\lambda)$, then no PPT adversary can solve $\text{CLWE}_{\text{shift}}(\lambda, \beta, \gamma, U(\mathbb{S}^{\lambda-1}))$ with advantage δ .*

We show that for any $\mathbf{s} \in \text{supp}(\frac{1}{r} \mathcal{S}) \subseteq \mathbb{S}^{\lambda-1}$ an adversary $\mathcal{A}$ with an oracle for $\text{CLWE}_{\mathbf{s},\text{shift}}$ can modify their oracle to one for $\text{CLWE}_{\mathbf{s}',\text{shift}}$ where $\mathbf{s}'$ is uniform on the sphere. At the beginning, $\mathcal{A}^{O(\cdot)}$ samples a uniformly random rotation $\mathbf{R} \in \mathbb{R}^{\lambda \times \lambda}$. On a query $\mathbf{u} \in \mathbb{R}^\lambda$, they generate

$$(\mathbf{R}\mathbf{y}, b), \quad \text{where } (\mathbf{y}, b) \leftarrow O(\mathbf{R}^\top \mathbf{u}).$$

By rotational invariance, when O is the null oracle, the distribution remains unchanged. When O is the $\text{CLWE}_{\mathbf{s},\text{shift}}$ oracle,

$$\begin{aligned} (\mathbf{R}\mathbf{y}, \gamma \mathbf{s}^\top \mathbf{y} + \gamma \mathbf{s}^\top (\mathbf{R}^\top \mathbf{u}) + e \bmod 1) &= (\mathbf{R}\mathbf{y}, \gamma (\mathbf{R}\mathbf{s})^\top \mathbf{R}\mathbf{y} + \gamma (\mathbf{R}\mathbf{s})^\top \mathbf{u} + e \bmod 1) \\ &\stackrel{d}{=} (\mathbf{y}', \gamma (\mathbf{R}\mathbf{s})^\top \mathbf{y}' + \gamma (\mathbf{R}\mathbf{s})^\top \mathbf{u} + e \bmod 1) \\ &\stackrel{d}{=} (\mathbf{y}', \gamma \mathbf{s}'^\top \mathbf{y}' + \gamma \mathbf{s}'^\top \mathbf{u} + e \bmod 1), \end{aligned}$$

where $\mathbf{y}' \leftarrow D_{\mathbb{R}^\lambda,1}$ is standard Gaussian by rotational invariance and $\mathbf{s}' \leftarrow \mathbb{S}^{\lambda-1}$ is uniform from the unit sphere because it is a randomly rotated unit vector.

8.3 Reducing CLWE with KDS to KDS Gaussian Pancake

The same reductions used in [BRST21; BNHR22] to reduce the CLWE distinguishing problem to the Gaussian pancake distinguishing problem extend naturally to the shifted setting. In particular, [BRST21] shows that if we perform rejection sampling on the first component $\mathbf{y}$ of a CLWE sample conditioned on the second component $\gamma \mathbf{s}^\top \mathbf{y} + e \bmod 1$ being close to some $z \in [0, 1]$, then we can generate a sample from $\text{GP}(\mathbf{s}, \mathbf{0}, z)$ in polynomial time. In the shifted setting, we can instead query for $\frac{\mathbf{u}}{\gamma}$ -shifted CLWE samples

$$(\mathbf{y}, \gamma \mathbf{s}^\top \mathbf{y} + \mathbf{s}^\top \mathbf{u} + e \bmod 1),$$

then apply the same rejection-sampling procedure, with the additional shift accounted for by the term $\mathbf{s}^\top \mathbf{u}$.

Lemma 8.3. *Let $\lambda \in \mathbb{N}$ be the security parameter (and dimension), let $\beta = 1/\text{poly}(\lambda)$ and $\gamma = \gamma(\lambda)$ be parameters. If no PPT adversary can solve $\text{CLWE}_{\text{shift}}(\lambda, \beta/\sqrt{2}, \gamma, U(\mathbb{S}^{\lambda-1}))$ with advantage $\delta = \delta(\lambda)$, then no PPT adversary can solve $\text{GP}_{\text{shift}}(\lambda, 1, \beta, \gamma)$ with advantage δ .*

Proof. The proof is identical to the rejection-sampling argument of [BRST21, Lemma 4.1], except that, when generating a sample from $\text{GP}(\mathbf{s}, \mathbf{u}, z)$, we do rejection sampling using $g_0(b) = \sum_{k \in \mathbb{Z}} \rho_{\beta/\sqrt{2}}(b + z + k)$ on freshly queried samples $(\mathbf{y}, b) \leftarrow \text{CLWE}_{\mathbf{s},\text{shift}}(\frac{\mathbf{u}}{\gamma})$. $\square$

8.3.1 Separating the parameters and the dimension

Earlier, in Proposition 3.1, we showed that when the dimension is sufficiently larger than the security parameter, we can separate the parameters β and γ from the dimension n . Here, we use the same idea to establish an analogous parameter separation for the GP_{shift} problem.

Lemma 8.4. *Let $\lambda \in \mathbb{N}$ be the security parameter and let $\beta = \beta(\lambda), \gamma = \gamma(\lambda)$ be parameters. If no PPT adversary can solve $\text{GP}_{\text{shift}}(\lambda, 1, \beta, \gamma)$ with advantage $\delta = \delta(\lambda)$, then for any $n = \text{poly}(\lambda) > \lambda$, no PPT adversary can solve $\text{GP}_{\text{shift}}(n, 1, \beta, \gamma)$ with advantage δ .*

Proof. For contradiction, assume there is some $n > \lambda$ and some PPT adversary $\mathcal{A}$ that can distinguish an oracle $\text{GP}(\mathbf{s}, \cdot, \cdot)$ where $\mathbf{s}' \leftarrow \mathbb{S}^{n-1}$ from a null oracle with advantage δ . Then we construct $\mathcal{B}$ against the lower dimensional problem. At the beginning $\mathcal{B}^{O(\cdot, \cdot)}(1^\lambda)$ samples a uniformly random basis for a λ -dimensional subspace of $\mathbb{R}^n$, $\mathbf{U} \leftarrow V_{\lambda, n}$. Let $\mathbf{U}^\perp \in \mathbb{R}^{n \times n-\lambda}$ arbitrarily complete the basis. $\mathcal{B}$ then implements the oracle O' as follows. On input $z \in [0, 1)$, $\mathbf{u} \in \mathbb{R}^n$,

- Sample $\mathbf{g} \leftarrow D_{\mathbb{R}^{n-\lambda}, 1}$.
- Sample $\mathbf{y} \leftarrow O(\mathbf{U}^\top \mathbf{u}, z)$.
- Output $(\mathbf{U}\mathbf{y} + \mathbf{U}^\perp \mathbf{g})$.

$\mathcal{B}$ then simulates $\mathcal{A}^{O'(\cdot, \cdot)}(1^\lambda)$ and outputs the bit returned by $\mathcal{A}$. Clearly, when $O(\cdot, \cdot)$ outputs standard Gaussian vectors, $O'(\cdot, \cdot)$ also outputs standard Gaussian vectors.

Now suppose $O(\cdot, \cdot)$ is an oracle for $\text{GP}(\mathbf{s}, \cdot, \cdot)$ for $\mathbf{s} \leftarrow \mathbb{S}^{\lambda-1}$. Recall that $\mathbf{y} \leftarrow O(\mathbf{U}^\top \mathbf{u}, z)$ can be written as

$$\mathbf{y} = (\mathbf{I} - \mathbf{s}\mathbf{s}^\top)\mathbf{h} + \left(\frac{k}{\gamma'} + e\right)\mathbf{s},$$

where $\mathbf{h} \leftarrow D_{\mathbb{R}^\lambda, 1}$, and $k \in \mathbb{Z} + (-(\mathbf{U}\mathbf{s})^\top \mathbf{u} + z \bmod 1)$. Then we have that

$$\mathbf{U}\mathbf{y} + \mathbf{U}^\perp \mathbf{g} = \mathbf{U}(\mathbf{I} - \mathbf{s}\mathbf{s}^\top)\mathbf{h} + \left(\frac{k}{\gamma'} + e\right)(\mathbf{U}\mathbf{s}) + \mathbf{U}^\perp \mathbf{g},$$

The Gaussian component is n -dimensional with covariance

$$\mathbf{U}(\mathbf{I} - \mathbf{s}\mathbf{s}^\top)\mathbf{U}^\top + \mathbf{U}^\perp(\mathbf{U}^\perp)^\top = \mathbf{U}\mathbf{U}^\top - (\mathbf{U}\mathbf{s})(\mathbf{U}\mathbf{s})^\top + \mathbf{I} - \mathbf{U}\mathbf{U}^\top = \mathbf{I} - (\mathbf{U}\mathbf{s})(\mathbf{U}\mathbf{s})^\top.$$

Therefore,

$$\mathbf{U}\mathbf{y} + \mathbf{U}^\perp \mathbf{g}_1 \stackrel{d}{=} (\mathbf{I} - (\mathbf{U}\mathbf{s})(\mathbf{U}\mathbf{s})^\top)\mathbf{g}' + \left(\frac{k}{\gamma'} + e\right)(\mathbf{U}\mathbf{s}),$$

where $\mathbf{g}' \leftarrow D_{\mathbb{R}^n, 1}$, which is exactly a sample from $\text{GP}(\mathbf{U}\mathbf{s}, \mathbf{u}, z)$.

Finally, since $\mathbf{U} \leftarrow V_{\lambda, n}$ and $\mathbf{s} \leftarrow \mathbb{S}^{\lambda-1}$ are uniform, the vector $\mathbf{U}\mathbf{s}$ is uniformly distributed over $\mathbb{S}^{n-1}$. Hence $O'(\cdot, \cdot)$ is an oracle for n -dimensional Gaussian pancake samples oracle with uniformly random secret, and $\mathcal{B}$ inherits the distinguishing advantage of $\mathcal{A}$. $\square$

8.3.2 Extension to multiple secrets

We can additionally use the same ideas as in [BRST21, Section 9] and the proof of Lemma 7.3 to see that the shifted problem remains hard when there are $\ell \leq n - \Omega(\lambda^c)$ secret directions.

Lemma 8.5. *Let $\lambda \in \mathbb{N}$ be the security parameter. Let $\beta = \beta(\lambda), \gamma = \gamma(\lambda)$ be parameters. Let $n = n(\lambda)$ and $\ell = \ell(\lambda)$ such that $n > \lambda$ and $n - \ell = \Omega(\lambda^c)$ for some constant $c > 0$ be the dimension and number of secrets, respectively. If no PPT algorithm can solve $\text{GP}_{\text{shift}}(n - \ell + 1, 1, \beta, \gamma)$ with advantage δ , then no PPT algorithm can solve $\text{GP}_{\text{shift}}(n, \ell, \beta, \gamma)$ with advantage $\ell\delta$.*

Proof. The proof is identical to that of [BRST21, Lemma 9.3], with one difference. In particular, the adversary, given a lower-dimensional single secret oracle $\text{GP}(\mathbf{s}, \cdot, \cdot)$, must adapt the procedure from the proof of Lemma 7.3 in order to simulate *shifted* multi-secret samples $\text{GP}(\mathbf{S}, \mathbf{u}, \mathbf{z})$ for $\mathbf{S} \in V_{i+1, n}$ as needed for the reduction. $\square$

References

- [AA58] Tom M Apostol and CM Ablow. *Mathematical analysis*. 1958 (page 62).
- [AACD+25] Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. “Ideal pseudorandom codes”. In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*. 2025 (pages 12, 33).
- [ABCP+18] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. “Turning your weakness into a strength: Watermarking deep neural networks by backdooring”. In: *27th USENIX security symposium (USENIX Security 18)*. 2018 (page 10).
- [ABNN+92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M Roth. “Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs”. In: *IEEE Transactions on information theory* 38.2 (1992) (page 34).
- [ACGM+16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. “Deep learning with differential privacy”. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016 (pages 6, 24, 63).
- [Ach03] Dimitris Achlioptas. “Database-friendly random projections: Johnson-Lindenstrauss with binary coins”. In: *Journal of computer and System Sciences* 66.4 (2003) (pages 33, 41, 42).
- [AJVZ26] Reyhaneh Abdolazimi, Shengmin Jin, Pramod Varshney, and Reza Zafarani. “Harnessing the power of noise: A survey of techniques and applications”. In: *ACM Computing Surveys* 58.13 (2026) (pages 25, 33).
- [APS15] Martin R Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of Learning with Errors.” In: *J. Math. Cryptol.* 9.3 (2015) (page 29).
- [BGIR+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. “On the (im) possibility of obfuscating programs”. In: *Journal of the ACM (JACM)* 59.2 (2012) (page 10).
- [Bis95] Chris M Bishop. “Training with noise is equivalent to Tikhonov regularization”. In: *Neural computation* 7.1 (1995) (pages 6, 24).
- [BNHR22] Andrej Bogdanov, Miguel Cueto Noval, Charlotte Hoffmann, and Alon Rosen. “Public-Key Encryption from Continuous LWE.” In: *IACR Cryptol. ePrint Arch.* 2022 (2022) (pages 8, 22, 55).
- [BRST21] Joan Bruna, Oded Regev, Min Jae Song, and Yi Tang. “Continuous lwe”. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021 (pages 1, 5, 7, 8, 18, 22, 55, 56).
- [BRV26] Andrej Bogdanov, Alon Rosen, and Neekon Vafa. “Statistically Undetectable Backdoors in Deep Neural Networks”. In: *arXiv preprint arXiv:2607.09532* (2026) (page 10).
- [BRVV26] Andrej Bogdanov, Alon Rosen, Neekon Vafa, and Vinod Vaikuntanathan. “Adaptive Robustness of Hypergrid Johnson-Lindenstrauss”. In: *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*. 2026 (page 13).
- [BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. “Private empirical risk minimization: Efficient algorithms and tight error bounds”. In: *2014 IEEE 55th annual symposium on foundations of computer science*. IEEE. 2014 (page 63).
- [CG24] Miranda Christ and Sam Gunn. “Pseudorandom error-correcting codes”. In: *Annual International Cryptology Conference*. Springer. 2024 (pages 4, 8, 10).
- [CGMR24] Miranda Christ, Sam Gunn, Tal Malkin, and Mariana Raykova. “Provably robust watermarks for open-source language models”. In: *arXiv preprint arXiv:2410.18861* (2024) (pages 3, 10).
- [CGZ24] Miranda Christ, Sam Gunn, and Or Zamir. “Undetectable watermarks for language models”. In: *The Thirty Seventh Annual Conference on Learning Theory*. PMLR. 2024 (pages 4, 10).
- [CGZ26] Ran Canetti, Shafi Goldwasser, and Or Zamir. “Proofs of Ownership for Machine Learning Models”. In: *arXiv preprint arXiv:2606.30423* (2026) (page 10).

- [CHNV+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. “Watermarking cryptographic capabilities”. In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016 (page 10).
- [CHS25] Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. “Watermarking language models for many adaptive users”. In: *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2025 (page 10).
- [CRK19] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. “Certified adversarial robustness via randomized smoothing”. In: *international conference on machine learning*. PMLR. 2019 (pages 6, 24).
- [CVS23] Jennifer Cobbe, Michael Veale, and Jatinder Singh. “Understanding accountability in algorithmic supply chains”. In: *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*. 2023 (page 3).
- [DD18] Daniel Dadush and Léo Ducas. *Periodic Gaussian, Discrete Gaussian and Transference*. Lecture notes for Mastermath, Intro to Lattice Algorithms & Cryptography. Lecture 7, scribe: S. Huiberts. 2018 (page 16).
- [Eur24] European Parliament and Council of the European Union. “Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence”. In: *Official Journal of the European Union L* (2024) (page 4).
- [FGJM+23] Jaiden Fairuze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. “Publicly-detectable watermarking for language models”. In: *arXiv preprint arXiv:2310.18491* (2023) (page 10).
- [GDJP+24] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024) (page 7).
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. “How to construct random functions”. In: *Journal of the ACM (JACM)* 33.4 (1986) (page 17).
- [GKVZ22] Shafi Goldwasser, Michael P Kim, Vinod Vaikuntanathan, and Or Zamir. “Planting undetectable backdoors in machine learning models”. In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022 (page 10).
- [GM24] Noah Golowich and Ankur Moitra. “Edit distance robust watermarks via indexing pseudo-random codes”. In: *Advances in neural information processing systems* 37 (2024) (page 10).
- [GSVV25] Shafi Goldwasser, Jonathan Shafer, Neekon Vafa, and Vinod Vaikuntanathan. “Oblivious defense in ml models: Backdoor removal without detection”. In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*. 2025 (page 10).
- [GTNP26] Greg Gluch, Berkant Turan, Sai Ganesh Nagarajan, and Sebastian Pokutta. “The Good, the Bad and the Ugly: Meta-Analysis of Watermarks, Transferable Attacks and Adversarial Defenses”. In: *Advances in Neural Information Processing Systems* 38 (2026) (page 10).
- [GVV22] Aparna Gupte, Neekon Vafa, and Vinod Vaikuntanathan. “Continuous lwe is as hard as lwe & applications to learning gaussian mixtures”. In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022 (pages 7, 16, 18, 29, 54, 55).
- [GZS25] Samuel Gunn, Xuandong Zhao, and Dawn Song. “An undetectable watermark for generative image models”. In: *International Conference on Learning Representations*. Vol. 2025. 2025 (page 10).
- [HPTD15] Song Han, Jeff Pool, John Tran, and William Dally. “Learning both weights and connections for efficient neural network”. In: *Advances in neural information processing systems* 28 (2015) (page 33).
- [HRF19] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. “Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2019 (page 6).

- [HWA+21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. “Lora: Low-rank adaptation of large language models”. In: *arXiv preprint arXiv:2106.09685* (2021) (page 33).
- [JL+84] William B Johnson, Joram Lindenstrauss, et al. “Extensions of Lipschitz mappings into a Hilbert space”. In: *Contemporary mathematics* 26.189-206 (1984) (pages 13, 33).
- [KGWK+23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. “A watermark for large language models”. In: *International conference on machine learning*. PMLR. 2023 (page 10).
- [KW17] Sam Kim and David J Wu. “Watermarking cryptographic functionalities from standard lattice assumptions”. In: *Annual International Cryptology Conference*. Springer. 2017 (page 10).
- [KW19] Sam Kim and David J Wu. “Watermarking PRFs from lattices: stronger security via extractable PRFs”. In: *Annual International Cryptology Conference*. Springer. 2019 (page 10).
- [LGH24] Jiedong Lang, Zhehao Guo, and Shuyu Huang. “A comprehensive study on quantization techniques for large language models”. In: *2024 4th International conference on artificial intelligence, robotics, and communication (ICAIRC)*. IEEE. 2024 (pages 7, 33).
- [LM00] Beatrice Laurent and Pascal Massart. “Adaptive estimation of a quadratic functional by model selection”. In: *Annals of statistics* (2000) (page 16).
- [LSS26] Huijia Lin, Kameron Shahabi, and Min Jae Song. “Unforgeable watermarks for language models via robust signatures”. In: *Annual International Cryptology Conference*. Springer. 2026 (pages 4, 6, 10).
- [Mic18] Daniele Micciancio. “On the hardness of learning with errors with binary secrets”. In: *Theory of Computing* 14.1 (2018) (pages 7, 18).
- [MMML+25] Emanuele Mezzi, Asimina Mertzani, Michael P Manis, Siyanna Lilova, Nicholas Vadivoulis, Stamatis Gatirdakis, Styliani Roussou, and Rodayna Hmede. “Who owns the output? bridging law and technology in llms attribution”. In: *arXiv preprint arXiv:2504.01032* (2025) (page 3).
- [MMP26] Noam Mazor, Andrew Morgan, and Rafael Pass. “Can We Watermark Low-Entropy LLM Outputs?” In: *arXiv preprint arXiv:2604.12051* (2026) (page 10).
- [MP11] Daniele Micciancio and Chris Peikert. “Trapdoors for lattices: Simpler, tighter, faster, smaller”. In: *Cryptology ePrint Archive* (2011) (page 44).
- [MR07] Daniele Micciancio and Oded Regev. “Worst-case to average-case reductions based on Gaussian measures”. In: *SIAM journal on computing* 37.1 (2007) (pages 15, 16).
- [NHBS+26] Anshul Nasery, Jonathan Hayase, Creston Brooks, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, and Sewoong Oh. “Scalable fingerprinting of large language models”. In: *Advances in Neural Information Processing Systems* 38 (2026) (pages 3, 10).
- [NN90] Joseph Naor and Moni Naor. “Small-bias probability spaces: Efficient constructions and applications”. In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990 (page 34).
- [NS26] Anna Neumann and Jatinder Singh. “AI Safety Evaluations Need To Consider Cascading Effects”. In: *Proceedings of IASEAI Conference*. Vol. 2. 1. 2026 (page 3).
- [Pei10] Chris Peikert. “An efficient and parallel Gaussian sampler for lattices”. In: *Annual Cryptology Conference*. Springer. 2010 (pages 11, 16, 25, 26).
- [PHDS+17] Matthias Plappert, Rein Houthoofd, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. “Parameter space noise for exploration”. In: *arXiv preprint arXiv:1706.01905* (2017) (page 6).
- [PKA25] Dario Pasquini, Evgenios M Kornaropoulos, and Giuseppe Ateniese. “{LLMmap}: Fingerprinting for large language models”. In: *34th USENIX Security Symposium (USENIX Security 25)*. 2025 (pages 3, 10).

- [PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. “Pseudorandomness of ring-LWE for any ring and modulus”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017 (page 15).
- [QWZ18] Willy Quach, Daniel Wichs, and Giorgos Zirdelis. “Watermarking PRFs under standard assumptions: public marking and security with extraction queries”. In: *Theory of Cryptography Conference*. Springer. 2018 (page 10).
- [RCS26] Mark Russinovich, Yanan Cai, and Ahmed Salem. “Hey, that’s my model! introducing chain & hash, an llm fingerprinting technique”. In: *International Conference on Learning Representations*. Vol. 2026. 2026 (pages 3, 10).
- [Reg09] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Journal of the ACM (JACM)* 56.6 (2009) (page 18).
- [RR07] Ali Rahimi and Benjamin Recht. “Random features for large-scale kernel machines”. In: *Advances in neural information processing systems* 20 (2007) (page 24).
- [SH26] Haochen Sun and Xi He. “GPM: The Gaussian Pancake Mechanism for Planting Undetectable Backdoors in Differential Privacy”. In: *Proceedings of the ACM on Management of Data* 4.3 (SIGMOD (2026) (page 24).
- [Sko23] Maciej Skorski. “Bernstein-type bounds for beta distribution”. In: *Modern Stochastics: Theory and Applications* 10.2 (2023) (pages 43, 47).
- [SKPS+22] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. “Learning from noisy labels with deep neural networks: A survey”. In: *IEEE transactions on neural networks and learning systems* 34.11 (2022) (page 24).
- [Ta-17] Amnon Ta-Shma. “Explicit, almost optimal, epsilon-balanced codes”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 2017 (page 34).
- [Tro12] Joel A Tropp. “User-friendly tail bounds for sums of random matrices”. In: *Foundations of computational mathematics* 12.4 (2012) (page 52).
- [UNSS17] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin’ichi Satoh. “Embedding watermarks into deep neural networks”. In: *Proceedings of the 2017 ACM on international conference on multimedia retrieval*. 2017 (pages 3, 10).
- [Ver18] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. 2018 (page 16).
- [WK19] Tianhao Wang and Florian Kerschbaum. “Attacks on digital watermarks for deep neural networks”. In: *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2019 (pages 3, 10).
- [WN23] David Gray Widder and Dawn Nafus. “Dislocated accountabilities in the “AI supply chain”: Modularity and developers’ notions of responsibility”. In: *Big data & society* 10.1 (2023) (page 3).
- [WWZY20] Jiangfeng Wang, Hanzhou Wu, Xinpeng Zhang, and Yuwei Yao. “Watermarking in deep neural networks via error back-propagation”. In: *Electronic Imaging* 32 (2020) (pages 3, 10).
- [WXLX+25] Zhebo Wang, Zhenhua Xu, Maike Li, Wenpeng Xing, Chunqiang Hu, Chen Zhi, and Meng Han. “Sraf: Stealthy and robust adversarial fingerprint for copyright verification of large language models”. In: *arXiv preprint arXiv:2505.06304* (2025) (page 3).
- [XWMK+24] Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. “Instructional fingerprinting of large language models”. In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 2024 (pages 3, 10).
- [Zam24] Or Zamir. “Undetectable steganography for language models”. In: *Transactions on Machine Learning Research* (2024) (page 8).

- [ZGCF+25] Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramer, et al. “Sok: Watermarking for ai-generated content”. In: *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2025 (page 10).
- [ZWHX+24] Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. “Huref: Human-readable fingerprint for large language models”. In: *Advances in Neural Information Processing Systems* 37 (2024) (pages 3, 10).

A Acknowledgments and AI Usage

This work was supported in part by the Simons Collaboration on the Theory of Algorithmic Fairness and an Amazon Research Award. KS thanks Rachel Hong for discussions on litigation in AI supply chains.

The authors acknowledge the use of AI tools in preparing this manuscript. Anthropic’s Sonnet 4.6 was used to generate code for Figures 1 and 2 based on hand-drawn representations provided by the authors. OpenAI’s GPT-5.6 suggested an approach that led to the reduction in Section 7.3, after being given the formal definitions of pancake alignment and CLWE.

B Noise Level of Natural Training Algorithms

Throughout the paper, we informally motivated noisy training algorithms as those that inject noise either into the final parameters or into each gradient step during training. In the latter case, a trivial decomposition still places the algorithm within our noisy training class: we can regard the noise injected at the final gradient step as the noise added to the final parameters, while absorbing all earlier noise into the base weights. However, this decomposition fails to capture the contribution of noise injected during earlier iterations of gradient descent. Recall for robustness, we want the algorithm’s noise level to be as large as possible.

In this section, we show that for many natural training algorithms, we can instead accumulate the noise injected across all gradient steps and decompose the algorithm into learning base weights followed by adding a larger noise vector at the end. This decomposition then fits naturally into our fingerprinting framework, because we can learn the base weights and use embedded, pseudogaussian noise for the final noise vector, thereby increasing the noise level σ compared to the trivial decomposition.

B.1 Accumulating noise during training

Let $L : \mathbb{R}^n \rightarrow \mathbb{R}$ be a twice-differentiable loss function that potentially depends on a dataset $(\mathbf{X}, \mathbf{y})$, and let $\{\eta^{(t)}\}_{t \in [T]}$ and $\{\sigma^{(t)}\}_{t \in [T]}$ be sequences of learning rates and noise levels, respectively. Consider a learning procedure that injects noise at each step of training. That is, weights $\mathbf{w}^{(0)} \in \mathbb{R}^n$ are initialized at the start, and at each step $t \in [T]$ the algorithm computes

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta^{(t)} (\nabla L(\mathbf{w}^{(t-1)}) + \mathbf{g}^{(t)}), \quad \text{where } \mathbf{g}^{(t)} \leftarrow D_{\mathbb{R}^n, \sigma^{(t)}}.$$

The algorithm then outputs $\mathbf{w}^{(T)}$ as the final weights. Trivially, any algorithm of this form is in the class $(\eta^{(T)}\sigma^{(T)}, n)$ -NoisyTrain because of the additive noise at the final step. We show that such an algorithm can additionally be approximated (and, in many cases, exactly represented) by a learning algorithm in (σ, n) -NoisyTrain, where the noise parameter σ accumulates all the noise $\sigma^{(t)}$ added throughout training.

Decomposing into clean weights plus noise. Let $\mathbf{w}_{\text{clean}}^{(t)} \in \mathbb{R}^n$ denote the weights at step t if they were generated by the algorithm *without* noise:

$$\mathbf{w}_{\text{clean}}^{(t)} = \mathbf{w}_{\text{clean}}^{(t-1)} - \eta^{(t)} \nabla L(\mathbf{w}_{\text{clean}}^{(t-1)}).$$

We show that the final noisy weights $\mathbf{w}^{(T)}$ can equivalently be computed as the clean weights $\mathbf{w}_{\text{clean}}^{(T)}$ plus a single (larger) Gaussian perturbation. Define $\mathbf{v}^{(t)} := \mathbf{w}^{(t)} - \mathbf{w}_{\text{clean}}^{(t)}$, and write $\mathbf{w}^{(t)}$ as base weights plus a perturbation

$$\mathbf{w}^{(t)} = \mathbf{w}_{\text{clean}}^{(t)} + \mathbf{v}^{(t)}.$$

We will show that in many settings $\mathbf{v}^{(T)}$ is distributed as noise term that can be sampled *after* computing the base weights $\mathbf{w}_{\text{clean}}^{(T)}$. We have $\mathbf{v}^{(0)} = \mathbf{0}$, and expanding the recurrence,

$$\begin{aligned}\mathbf{v}^{(t)} &= \mathbf{w}^{(t-1)} - \mathbf{w}_{\text{clean}}^{(t-1)} + \eta^{(t)} \left(\nabla L(\mathbf{w}_{\text{clean}}^{(t-1)}) - \nabla L(\mathbf{w}^{(t-1)}) - \mathbf{g}^{(t)} \right) \\ &= \mathbf{v}^{(t-1)} + \eta^{(t)} \left(\nabla L(\mathbf{w}_{\text{clean}}^{(t-1)}) - \nabla L(\mathbf{w}^{(t-1)}) - \mathbf{g}^{(t)} \right).\end{aligned}$$

By the multivariate mean value theorem [AA58, Theorem 12.9], for each $t \in [T]$ there exists some $\hat{\mathbf{w}}^{(t)} \in \mathbb{R}^n$ (on the line between $\mathbf{w}_{\text{clean}}^{(t)}$ and $\mathbf{w}^{(t)}$) such that

$$\nabla L(\mathbf{w}_{\text{clean}}^{(t)}) - \nabla L(\mathbf{w}^{(t)}) = \nabla^2 L(\hat{\mathbf{w}}^{(t)}) (\mathbf{w}_{\text{clean}}^{(t)} - \mathbf{w}^{(t)}) = -\nabla^2 L(\hat{\mathbf{w}}^{(t)}) \mathbf{v}^{(t)}.$$

Substituting this into the equation from above gives us

$$\mathbf{v}^{(t)} = \left(\mathbf{I} - \eta^{(t)} \nabla^2 L(\hat{\mathbf{w}}^{(t-1)}) \right) \mathbf{v}^{(t-1)} - \eta^{(t)} \mathbf{g}^{(t)},$$

and expanding this recurrence down to $\mathbf{v}^{(0)} = \mathbf{0}$ yields

$$\mathbf{v}^{(T)} = \sum_{t=1}^T \eta^{(t)} \left(\prod_{j=t+1}^T \left(\mathbf{I} - \eta^{(j)} \nabla^2 L(\hat{\mathbf{w}}^{(j-1)}) \right) \right) \mathbf{g}^{(t)}.$$

Sampling a single noise vector. In some cases, we can compute $\mathbf{v}^{(T)}$ by sampling a single noise vector. Define

$$\Sigma := \sum_{t=1}^T (\eta^{(t)} \sigma^{(t)})^2 \mathbf{A}^{(t)} (\mathbf{A}^{(t)})^\top, \quad \text{where } \mathbf{A}^{(t)} = \left(\prod_{j=t+1}^T \left(\mathbf{I} - \eta^{(j)} \nabla^2 L(\hat{\mathbf{w}}^{(j-1)}) \right) \right).$$

Since the $\mathbf{g}^{(t)}$ are independent isotropic Gaussian vectors, $\mathbf{v}^{(T)}$ is distributed as Gaussian with covariance Σ . Thus, letting $\Sigma^{1/2}$ denote any matrix square root of Σ , we can equivalently generate $\mathbf{v}^{(T)}$ by sampling $\mathbf{h} \leftarrow D_{\mathbb{R}^n, 1}$ once and outputting $\mathbf{v}^{(T)} = \Sigma^{1/2} \mathbf{h}$.

If the standard deviation $\Sigma^{1/2}$ can be computed in advance, then this lets us decompose the training algorithm as: (1) train $\mathbf{w}_{\text{clean}}^{(T)}$ with no noise, (2) compute $\Sigma^{1/2}$, (3) sample a noise vector $\mathbf{g} = \Sigma^{1/2} \mathbf{h}$, and (4) output $\mathbf{w}^{(T)} = \mathbf{w}_{\text{clean}}^{(T)} + \mathbf{g}$. When $\Sigma^{1/2} = \sigma \mathbf{I}$ is isotropic, this algorithm falls in the class (σ, n) -Train.

How to compute $\Sigma^{1/2}$? Note that computing $\Sigma^{1/2}$ requires computing $\nabla^2 L(\hat{\mathbf{w}}^{(t)})$ at the unknown intermediate points $\hat{\mathbf{w}}^{(t)}$. There are two ways to handle this:

1. *Heuristic approximation.* Replace $\nabla^2 L(\hat{\mathbf{w}}^{(t)})$ with $\nabla^2 L(\mathbf{w}_{\text{clean}}^{(t)})$, which is computable along the clean training trajectory. This yields an algorithm that is heuristically close to, but not exactly equal to, the original noisy procedure.
2. *Exact computation when the Hessian is constant.* For some natural loss functions, $\nabla^2 L(\mathbf{w})$ does not depend on the input $\mathbf{w}$ at all, so it can be computed exactly (and in advance, before training). For example, for the least-squares loss induced by a dataset $(\mathbf{X}, \mathbf{y})$,

$$L(\mathbf{w}) = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2,$$

the gradient and Hessian are

$$\nabla L(\mathbf{w}) = \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y}), \quad \nabla^2 L(\mathbf{w}) = \mathbf{X}^\top \mathbf{X},$$

which is independent of $\mathbf{w}$. In this case $\Sigma^{1/2}$ can be computed exactly, with $\mathbf{X}^\top \mathbf{X}$ substituted directly for every $\nabla^2 L(\hat{\mathbf{w}}^{(j-1)})$ term in the product above. In addition, when the loss is piecewise linear, for example,

$$L(\mathbf{w}) = |\mathbf{X}\mathbf{w} - \mathbf{y}|$$

then the Hessian is $\mathbf{0}$ almost everywhere.

When is the noise $\Sigma^{1/2}$ isotropic? Although throughout the main body we embed the fingerprint into isotropic Gaussian noise, this is primarily for ease of presentation, and our constructions extend naturally to noisy training algorithms whose Gaussian noise has arbitrary full rank covariance matrix Σ . Rather than replacing the noise with $\sigma \mathbf{y}$ for a Gaussian pancake vector $\mathbf{y}$, we can use $\Sigma^{1/2} \mathbf{y}$. In this case, the σ term appearing in our robustness thresholds is replaced by the appropriate noise level $1/\|\Sigma^{-1/2}\|_{op}$. Thus, the ideas shown above already allow us to accumulate the noise injected throughout a training algorithm into a single Gaussian vector with covariance Σ , which can then be fingerprinted. This can yield better robustness than the trivial decomposition, which uses only the noise $\eta^{(T)} \sigma^{(T)}$ injected at the final training step.

However, since we focus on the isotropic setting in the main body, it is useful to characterize when the accumulated noise is isotropic, allowing us to determine the effective noise level σ . This occurs, for example when the Hessian is $\mathbf{0}$, or, in the case of the least-squares loss, when the data matrix $\mathbf{X}^\top \mathbf{X} = \lambda \mathbf{I}$ for some $\lambda \in \mathbb{R}$ ¹⁶. Then we have that $\Sigma^{1/2} = \sigma \mathbf{I}$ for σ satisfying

$$\sigma = \sqrt{\sum_{t=1}^T \left(\eta^{(t)} \sigma^{(t)} \prod_{j=t+1}^T (1 - \eta^{(j)} \lambda) \right)^2},$$

which can be substantially larger than $\eta^{(T)} \sigma^{(T)}$.

B.2 Noisy training algorithms

Here, we give settings where accumulating the noise $\sigma^{(t)}$ may be useful. We are slightly informal because the specific choice of noise is heavily dependent on the application.

Differential privacy. In the differential privacy literature, the noise level and learning rates are set appropriately to ensure privacy while preserving utility. In particular, [ACGM+16, Theorem 1] says that to obtain (ϵ, δ) -differential privacy, we require noise

$$\sigma^{(t)} = \Omega\left(\frac{C\sqrt{T\log(1/\delta)}}{\epsilon d}\right) \quad \forall t \in [T],$$

where $d \in \mathbb{N}$ is the number of datapoints used for training, $T \in \mathbb{N}$ is the total number of training steps, and $C \in \mathbb{R}$ is a bound on the gradient of the loss function, $\|\nabla L(\mathbf{w})\|_2 \leq C$ for all $\mathbf{w} \in \mathbb{R}^n$. A reasonable setting that additionally guarantees utility is, given d private datapoints, run the training algorithm for $T = d^2$ steps using step size $\eta^{(t)} = \Theta(\frac{1}{td})$ and the same noise $\sigma^{(t)} = \Theta(C\sqrt{\log(1/\delta)}/\epsilon)$ at each step [BST14, Theorem 2.4]¹⁷. The noise of this algorithm, when accumulated (ignoring the Hessian terms), is approximately

$$\sigma = \sum_{t \in [d^2]} \eta^{(t)} \sigma^{(t)} = \Omega\left(\frac{C\sqrt{\log(1/\delta)} \log d}{\epsilon d}\right).$$

Without accumulation, the noise is instead smaller, $\eta^{(d^2)} \sigma^{(d^2)} = \Theta(C\sqrt{\log(1/\delta)}/(\epsilon d^3))$. Informally, one can view this accumulated noise as a slightly lower order than the norm of the clean base weights $\mathbf{w}_{\text{clean}}^{(d^2)}$. This is because, loosely, $\|\mathbf{w}_{\text{clean}}^{(d^2)}\|_2 \leq Cd$, as it is the sum of d^2 gradients of norm at most C , and the accumulated noise σ is smaller by a factor of $\frac{\sqrt{\log(1/\delta)} \log d}{\epsilon d^2}$.

Noisy data. When the training data is noisy, we show that for certain loss functions, the noise *passes through* the gradient as an independent additive Gaussian term, allowing noise injected into the data at each iteration to be accumulated into a single noise vector as described above. Concretely, suppose the training

¹⁶For instance, when the data comes from a spherical distribution.

¹⁷This setting gives an excess risk bound of $O(\frac{C^2 n \log(1/\delta)}{d \epsilon^2})$ when the loss L is $\Theta(1)$ -strongly convex.

algorithm uses noisy labels $\mathbf{y}' \in \mathbb{R}^d$, distributed as clean labels $\mathbf{y} \in \mathbb{R}^d$ perturbed by independent Gaussian noise: $\mathbf{y}' \stackrel{d}{=} \mathbf{y} + \mathbf{g}$ where $\mathbf{g} \leftarrow D_{\mathbb{R}^d, \sigma}$. Consider again the quadratic loss $L_{\mathbf{X}, \mathbf{y}}(\mathbf{w}) = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$. We have

$$\begin{aligned} \nabla L_{\mathbf{X}, \mathbf{y}'}(\mathbf{w}) &= \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y}') \\ &= \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y} - \mathbf{g}) \\ &= \nabla L_{\mathbf{X}, \mathbf{y}}(\mathbf{w}) - \mathbf{X}^\top \mathbf{g}, \end{aligned}$$

which shows that computing a gradient on noisy data is equivalent to computing the gradient on clean data plus adding an independent Gaussian noise vector that one can compute with the clean data, $\mathbf{h} = -\mathbf{X}^\top \mathbf{g}$ (which has covariance $\sigma^2 \mathbf{X}^\top \mathbf{X}$). Consequently, gradient descent with fresh label noise at each iteration can be represented as gradient descent on the clean data with an additive Gaussian noise term at each iteration, which can then be accumulated as in the preceding section.

A similar argument holds for linear losses, for example $L_{\mathbf{X}, \mathbf{y}}(\mathbf{w}) = -\mathbf{y}^\top \mathbf{X}\mathbf{w}$, even when fresh noise is injected into the *features* at each iteration, $\mathbf{X}' = \mathbf{X} + \mathbf{G}$ where $\mathbf{G} \leftarrow D_{\mathbb{R}^{d \times n}, \sigma}$. Then

$$\begin{aligned} \nabla L_{\mathbf{X}', \mathbf{y}}(\mathbf{w}) &= -(\mathbf{X} + \mathbf{G})^\top (\mathbf{y}) \\ &= \nabla L_{\mathbf{X}, \mathbf{y}}(\mathbf{w}) - \mathbf{G}^\top \mathbf{y}. \end{aligned}$$

where noise $-\mathbf{G}^\top \mathbf{y}$ is distributed as Gaussian with covariance $\sigma^2 \|\mathbf{y}\|_2^2 \mathbf{I}_n$.