HEFT: Homomorphically Encrypted Fusion of Biometric Templates (Supplementary Material)

Luke Sperling[†] Nalini Ratha[‡] Arun Ross[†] Vishnu Naresh Boddeti[†]
[†]Michigan State University [‡]University at Buffalo

This supplementary material includes the following:

1. **Section 1:** Two additional linear projection methods.
2. **Section 2:** Discussion on practical considerations to account for when approximating inverse square root in the encrypted domain.
3. **Section 3:** Additional experiments on fusing fingerprint and face templates.

1. Additional Linear Projection Methods

Naive: The naive method [2] aims to perform a matrix-vector multiplication similarly to how it would be computed in the unencrypted domain. The encoding scheme for the matrix encodes each row of the matrix as a plaintext. The query vector is presumed to be in the dense representation. The homomorphic inner product is taken between the vector and each row of the matrix. This results in γ ciphertexts. Each of these ciphertexts holds one dimension of the result replicated in every slot. To combine these results into a single ciphertext, each result ciphertext is multiplied by a mask plaintext containing one in the appropriate index and zeroes elsewhere. All of these ciphertexts are added together, yielding the final result. This method is considerably slowed down by the fact that it features two serial multiplications. This necessitates a slower ciphertext-ciphertext multiplication as well as requiring a larger coefficient modulus to be selected, which slows all our operations.

Diagonal: This method, devised by Halevi and Shoup [2], reduces the number of serial multiplications from two to one by not computing homomorphic inner products. Instead, by encoding the diagonals of a matrix as seen in Figure 2 and multiplying by rotated versions of the query vector (which is again presumed to be in the dense encoding scheme), the resultant vectors can simply be summed to yield the result. This diagonal encoding scheme and algorithm assumes that the matrix is square. Thus, we either need to zero-pad our rectangular matrix to use this method or not reduce the dimensionality of the concatenated representation (i.e., $\gamma = \delta$). Compressing the representation is important to enable efficient match score computation, however, so this approach is not desirable.

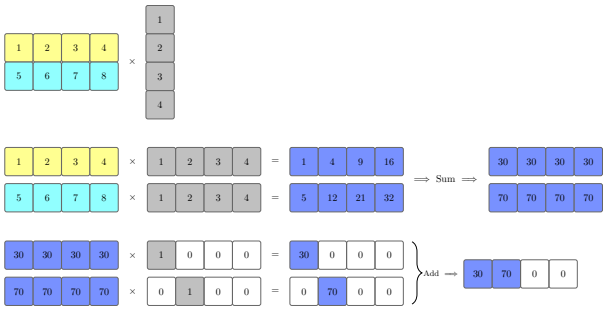

Figure 1: **Naive method:** A naive method for matrix-vector multiplication. Each row of the projection matrix is encoded as a plaintext. Through a series of homomorphic inner products, we obtain a ciphertext representing each dimension of the result. Through masking and adding, we arrive at the final result.

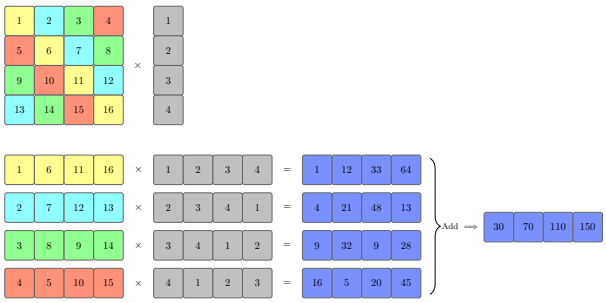

Figure 2: **Diagonal method:** Encoding over diagonals of a square matrix instead of over the rows results in a method that reduces the multiplicative depth from two to one. The query vector is rotated once and multiplied by each subsequent matrix plaintext. The sum of all these results yields the final result.

2. Practical Considerations for Approximating Inverse Square Root

Panda [3] showed that it is possible to normalize an encrypted vector using Goldschmidt's Algorithm to approx-

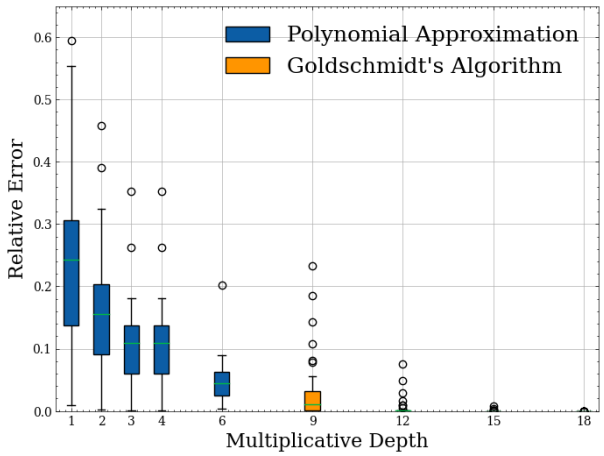

Figure 3: Comparison of multiplicative depth vs. relative error for different selections of inverse square root approximation. Polynomial approaches can achieve lower multiplicative depth, but incur more relative error than Goldschmidt's Algorithm.

imate inverse square root. This method of approximating inverse square root in the encrypted domain is not particularly suitable for practical applications due to its high multiplicative depth requirements, slowing every computation performed in the encrypted domain. We instead use a polynomial approximation constrained over an interval. Figure 3 shows the multiplicative depth of different parameters for polynomial and Goldschmidt's approximations and their relative errors. Care must be taken to ensure that an appropriate range is selected for the approximation, such that the results of the projection yield squared norms that fall in the valid range. Smaller ranges yield more precise polynomial approximations, but are more susceptible to samples not falling within the range. We select the range $[0.05 - 3.00]$ and constrain the norm of our projection matrix $\boldsymbol{P}$ such that the mean of the squared norms of the projections within the train set is in the middle of the range. If samples fall outside this valid range during training, we skip those samples during that epoch.

3. Additional Experiments

We explore the capacity of HEFT when used on a dataset with little room for performance improvement. NIST SD4 [4] is a dataset consisting of ink-rolled fingerprint images. It contains 4,000 samples over 2,000 identities. 192D features are extracted via DeepPrint [1]. We pair the well-performing NIST SD4 dataset with the more challenging CPLFW dataset to create a multimodal dataset of 7,696 samples over 1,924 classes.

The AUROC of HEFT compared to the baseline is shown in Table 1 and Fig. 4. The averaging method is not evaluated, since it only works on representations that are of

Data	Domain	Encrypted Normalization	Learning Normalization	Dimensionality	AUROC
CPLFW	Message	-	-	512	0.8253
NIST SD4	Message	-	-	192	0.99997
Concatenation	Message	-	-	704	0.9982
Learned	Message	-	Exact	32	0.99990
Learned	Encrypted	Poly (Deg=6)	Exact	32	0.9883
Learned	Encrypted	Poly (Deg=2)	Exact	32	0.9294
Learned	Encrypted	Goldschmidt's	Exact	32	0.9980
Learned	Message	-	HEFT (Deg=2)	32	0.9925
Learned	Encrypted	Poly (Deg=2)	HEFT (Deg=2)	32	0.9925

Table 1: AUROC comparison of HEFT versus baseline

the same dimensionality. NIST SD4 is a somewhat compact representation of 192D, but HEFT can compress it by a factor of 6 to 32D with only a 0.75% drop in AUROC. Notably, HEFT improves matching performance over exact learning by 6.31%. Additionally, HEFT outperforms the exact normalization learning method when a degree 6 polynomial is used at inference time by 0.42%. These results show HEFT's utility at compressing highly accurate representations via fusion.

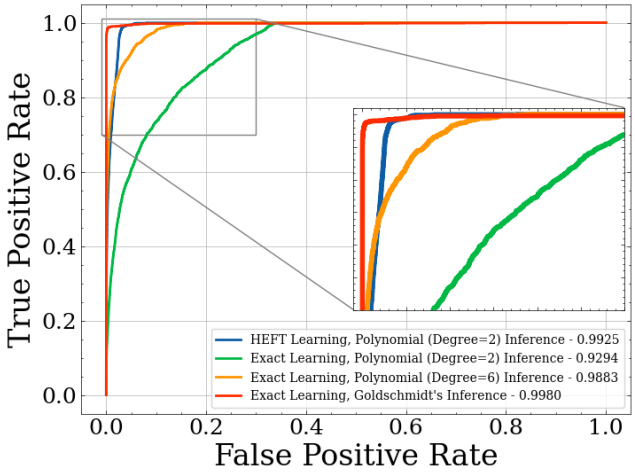

Figure 4: ROC comparison of HEFT against baseline.

References

[1] J. J. Engelsma, K. Cao, and A. K. Jain. Learning a fixed-length fingerprint representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):1981–1997, 2019. 2

[2] S. Halevi and V. Shoup. Algorithms in helib. In *Annual Cryptology Conference*. Springer, 2014. 1

[3] S. Panda. Principal component analysis using ckks homomorphic encryption scheme. *Cryptology ePrint Archive*, 2021. 1

[4] C. I. Watson and C. L. Wilson. Nist special database 4. *Fingerprint Database, National Institute of Standards and Technology*, 17(77):5, 1992. 2