Abhishek Jana, Bipin Paudel, Md Kamruzzaman Sarker, Monireh Ebrahimi, Pascal Hitzler, and George T Amariucai*

Neural Fuzzy Extractors: A Secure Way to Use Artificial Neural Networks for Biometric User Authentication

Abstract: Powered by new advances in sensor development and artificial intelligence, the decreasing cost of computation, and the pervasiveness of handheld computation devices, biometric user authentication (and identification) is rapidly becoming ubiquitous. Modern approaches to biometric authentication, based on sophisticated machine learning techniques, cannot avoid storing either trained-classifier details or explicit user biometric data, thus exposing users' credentials to falsification. In this paper, we introduce a secure way to handle user-specific information involved with the use of artificial neural networks for biometric authentication. Our proposed architecture, called a Neural Fuzzy Extractor (NFE), allows the coupling of pre-existing classifiers with fuzzy extractors, through an artificial-neural-network-based buffer called an expander, with minimal or no performance degradation. The NFE thus offers all the performance advantages of modern deep-learning-based classifiers and all the security of standard fuzzy extractors. We demonstrate the NFE retrofit of a few classic artificial neural networks, for simple biometric authentication scenarios.

Keywords: Security, Deep Learning, Fuzzy Extractor, Authentication, Artificial Neural Network.

DOI 10.56553/popets-2022-0100
Received 2022-02-28; revised 2022-06-15; accepted 2022-06-16.

Abhishek Jana: Kansas State University, E-mail: ajana@ksu.edu
Bipin Paudel: Kansas State University, E-mail: bipinp@ksu.edu
Md Kamruzzaman Sarker: Kansas State University, E-mail: mdkamruzzamansarker@ksu.edu
Monireh Ebrahimi: Kansas State University, E-mail: monireh@ksu.edu
Pascal Hitzler: Kansas State University, E-mail: hitzler@ksu.edu
***Corresponding Author: George T Amariucai:** Kansas State University, E-mail: amariucai@ksu.edu

1 Introduction

Secure architectures for password-based authentication avoid storing the passwords corresponding to each username, and resort instead to storing cryptographic hash digests of such passwords (along with salt, pepper, and other such auxiliary randomness). The rationale behind this widely-accepted paradigm is that not even a superuser, with complete access to the entire file system, should be able to fraudulently log in as one of the other, less privileged users of the system.

When using standard biometric authentication in place of (or in addition to) a password, similar functionality can be achieved through the use of fuzzy extractors [14]. However, despite their security guarantees, fuzzy extractor architectures are rarely deployed in practice. In particular, in the case of fingerprint-based authentication it turns out that similarity-score algorithms usually perform better than fuzzy extractors, and this has led to a preference for the former. For example, using fuzzy extractors, [2] reports an equal error rate of 15% on the FVC2000 database [56], while the top five FVC2000 competitors report equal error rates between 5% and 0.7% [38].

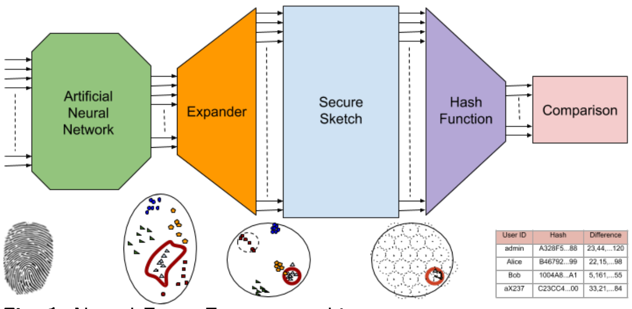
However, using similarity-score algorithms involves the storing of fingerprint databases, which are vulnerable to leakage. Notable examples are the Office of Personnel Management data breach of 2015 [19], in which 5.6 million sets of fingerprints were leaked, and the more recent Suprema Biostar leak of 2019 [54], which compromised the fingerprint and facial biometric information of more than one million people. An authentication method which has both the security of fuzzy extractors and the superior performance of similarity-score algorithms would clearly be prefered.

The reason behind the inferior performance of fuzzy extractors may be traced back to the very essence of fuzzy extractor functionality, the first stage of which consists of a channel decoding mechanism over a vector space. By nature, good channel coding implicitly assumes spherical (or close to spherical) decoding regions, which correspond to the decision regions of the vector-space-based classifier. In contrast to this, normal support-vector machines (SVMs) can learn highly irregular decision regions – hence their superiority. Similarly, artificial neural networks (ANNs) are known to outperform even the best of SVMs (at least in situations in which training data is abundant), and this is attributed to their ability to learn highly-irregular classification functions.

Unfortunately, both SVMs and ANNs (as well as the other frequently-used classifiers, like k-nearest neighbors (KNN), decision trees and random forests, etc.) rely on learned structures that have to be stored in non-volatile memory, similarly to a password file. A malicious user, with access to this information, could use the learned structure (for example, by back-tracking through an ANN, or by simply choosing a vector in the proper decision region, for an SVM) to produce synthetic inputs guaranteed to pass the authentication test.

The question that arises naturally is then how we can protect a user's biometric authentication information in a manner similar to the way in which we treat passwords, but without suffering from the spherical restrictions of the fuzzy extractors. In this paper, we propose a new (and severely overdue) such architecture, which we call *neural fuzzy extractors (NFEs)*.

NFEs are a concatenation of a classifier – such as an artificial neural network – with a fuzzy extractor, as illustrated in Figure 1. Nevertheless, as most ANNs are designed to output class labels or scores contained within some subspace of a certain vector space, and we want our NFE construction to be as close as possible to an add-on, to take advantage of many already-existing and well performing classification architectures, we have to retrofit NFEs to regular ANNs. We do this by constructing an interface between the ANN and the fuzzy extractor's decoding mechanism, which we call an *expander*. An expander will typically consist of an additional ANN with a few layers, that can be added to the end of any type of ANN, and the sole purpose of which is to re-cast the output embedding of the original ANN to a vector space in which representatives from each class cluster together in sphere-like clusters. The expander may be trained independently, based on labeled embedings from the original ANN, or together with the original ANN.



User ID	Hash	Difference
admin	A328F5...88	23,44,...120
Alice	B46792...99	22,15,...98
Bob	1004A8...A1	5,161,...55
aX237	C23CC4...00	33,21,...84

Fig. 1. Neural Fuzzy Extractor architecture.

Of course, some already-existing ANN architectures may be naturally suited to concatenation with fuzzy extractors, in the sense that their output embeddings corresponding to different classes are already in a vector space and already cluster in spheres. In these cases, no additional expander is necessary.

We should also note that the above-mentioned vector spaces need not be defined over $\mathbb{R}^n$ – as they are for the particular architecture discussed in this paper – but can also be defined over any other field, like $GF(2^n)$. The latter choice is easier to deal with from the perspective of the secure sketch, mainly due to the wide availability of binary channel capacity-achieving error correction codes (ECCs) – by contrast, additive white Gaussian noise (AWGN) channel capacity-achieving ECCs are few and suffering from very complex decoding mechanisms. On the other hand, forcing the output of the expander into binary vectors may incur unacceptable performance losses, which is why we leave the investigation of binary-ECC-based NFEs to future work.

The contributions of this paper are as follows.

1. We introduce NFEs, a first secure architecture for handling ANN-based biometric user authentication.
2. We show that NFEs can be retro-fitted to work with most of the already-existing ANN-based biometric authentication architectures.
3. We demonstrate our construction on three already-existing fingerprint-based authentication architectures, and we show that the NFE retrofit has marginal, if any, effects on performance, while introducing non-negligible but acceptable overhead to the training and running times.

The remainder of the paper is organized as follows. Section 2 provides a brief survey of already-existing biometric authentication protocols, focusing mostly on recent results involving artificial neural networks, and techniques that use fuzzy extractors. Section 3 introduces the NFE architecture and provides some general implementation insights. Section 4 presents the design decisions involved in our instantiation of the NFE architecture, namely the types of classifiers considered, the architecture of the expander and the choice of the error-correction code. Section 5 describes the decisions involved in the design of our evaluation framework, such as the datasets and the experimental setup, and presents our results in terms of authentication performance, training and running times. Finally, conclusions are drawn in Section 6.

2 Related Work

2.1 Biometric Authentication and Identification

The literature contains many early attempts at leveraging the ANN capabilities for biometric verification and authentication. [6] has filed a patent on biometric recognition using a classification neural network. The patented biometric recognition system involves two phases: creation of a master pattern set of authorized users' biometric identifications and authentications using a classification neural network. [1] developed a new supervised recurrent neural network for fingerprint authentication. Their approaches used similarity measures of features for clustering and ranking of the fingerprint representations stored in their database. [62] used both artificial neural networks and k-nearest neighbors as possible classifiers for typing pattern identification. [67] investigates the implementation of Weightless Neural Networks (WNNs) as a pattern recognition tool to classify users' typing patterns and thus attempts to separate the real users from impostors. [42] used artificial neural networks for face representation learning and recognition.

With the recent resurgence of interest in Deep Learning models, in recent years we have witnessed significant progress in representation learning for biometric identifiers by deep neural networks. [53] has proposed FingerNet, a unified deep network for fingerprint minutiae extraction. They propose a new way to design a deep convolutional network combining domain knowledge and the representation ability of deep learning. In terms of orientation estimation, segmentation, enhancement and minutiae extraction, several typical traditional methods that performed well on rolled/slap fingerprints are transformed into a convolutional approach and integrated as a unified plain network. [10] posed minutiae extraction as a machine learning problem and proposed a deep neural network – MENet, for Minutiae Extraction Network – to learn a data-driven representation of minutiae points. [40] used deep representations for Iris, Face, and Fingerprint Spoofing Detection. Similarly, [17] learned fingerprint representations. [29] proposed three variations of the VGGNet structure for fingerprint classification.

Similarly, [21] proposed a secure multimodal biometric system that uses a convolutional neural network (CNN) and a Q-Gaussian multi support vector machine (QG-MSVM) based on different level fusion. They developed two authentication systems with two different level fusion algorithms: a feature level fusion and a decision level fusion. The feature extraction for individual modalities is performed using a CNN. In this step, they selected two layers from the CNN that achieved the highest accuracy, in which each layer is regarded as a separate feature descriptor. After that, they combined them using the proposed internal fusion to generate the biometric templates. In the next step, they applied one of the cancelable biometric techniques to protect these templates and increase the security of the proposed system.

Likewise, [41] conducted multimodal biometric face and fingerprint recognition using neural networks based on adaptive principal component analysis and multilayer perceptrons. [50] proposed a novel latent overlapped fingerprints separation algorithm based on neural networks. [43] used convolutional neural networks (CNNs) for fingerprint liveness detection.

[44] leveraged neural networks to both identify QRS complex segments of ECG signals and then performed user authentication on these segments. [37] used multilayer perceptrons and radial basis function neural networks for electrocardiogram (ECG) biometric authentication. [46] proposed the use of various recurrent neural network (RNN) architectures (including vanilla, long short-term memory (LSTM), gated recurrent unit (GRU), unidirectional, and bidirectional networks) for ECG-based biometrics identification/classification and authentication. [31] presents Deep-ECG, a CNN-based biometric approach for ECG signals identification, verification and periodic re-authentication. Deep-ECG extracts prominent features from one or more leads using a deep CNN and compares biometric templates by computing simple and fast distance functions for verification or identification. [16] showed the novel use and effectiveness of deep learning CNN architectures for automatic rather than hand-crafted feature extraction for robust face recognition across time lapses. They show CNNs using the VGG-Face deep networks produce highly discriminative and interoperable features that are robust to aging variations even across a mix of biometric datasets.

2.2 Fuzzy Extractors

Fuzzy extractors were introduced in [14] as a secure way of coping with user biometrics – for which every new entry is slightly different from previous ones, but all entries share some common main features. The idea was that, instead of storing representative entries, for direct comparison to the new entries upon authentication request, the system should only store digests obtained through cryptographic hash functions – thus preventing biometric falsification.

The idea was quickly adapted to various types of biometric authentication mechanisms, like those based on fingerprints [2, 32, 58, 63, 64], iris scans [7, 25, 39], face [51] or gait [26, 27].

More recently, fuzzy extractors were used in the context of more sophisticated and specialized secure authentication mechanisms, like the one in [11], designed specifically for wireless sensor networks (like body-area networks), or the ones in [13, 33], which deal with the outputs of physically-unclonable functions (PUFs).

However, to the best of our knowledge, at the time of this writing, no works exist on the application of fuzzy extractors on biometric data pre-processed by sophisticated classifiers like artificial neural networks.

2.3 Privacy-Preserving Biometrics

Our problem is related to the problem of privacy-preserving biometric authentication [3, 59], to the extent that our NFE solution helps with some of the common objectives of privacy-preserving biometrics, such as biometric template protection [3]. Nevertheless, the problem of privacy-preserving biometric authentication is much broader, dealing with the privacy of the entire biometric authentication process, from the security of the channel between sensor and authentication server, to the security of the authentication database, and from anonymous biometric verification [18] to biometric cancellability [55]. The techniques employed in the literature to achieve these diverse goals range from the use of fuzzy extractors [12] to secure multi-party computation [8], and from zero-knowledge proofs of knowledge [4] to fully-homomorphic encryption [18, 30].

3 Preliminary Concepts: the Architecture of a Neural Fuzzy Extractor

Most classifiers – whether neural networks or vector-space-based – will output a vector representation of the input data. In some cases, the output vectors are already appropriate for direct input to the secure sketch (see Figure 1). To satisfy this property, the classifier outputs corresponding to each class of interest have to cluster in a somewhat spherical region of the vector space. This is because the secure sketch will use codes designed for error correction on either white Gaussian noise (AWGN) channels, or on binary symmetric channels (BSCs), where the decoding region is spherical by construction. So if the classifier's decision regions are not already spherical, imposing spherical decoding regions on top of them will invariably degrade the classification performance.

Unfortunately, in most cases, the classifiers are solely designed for good accuracy, without extra constraints on the spherical shape of their decision regions. In such cases, we propose the use of an *expander*, as illustrated in Figure 1, to further shape the classifier's decision regions into spherical ones. We call this procedure *retrofitting the NFE to pre-existing classifiers*. Intuitively, we expect that in order to avoid reducing the overall accuracy, we need to preserve as much of the information content of the classifier output as possible. Intuitively, this means that the expander should generally project the vector representations of the classifier's output to a larger-dimensional space (hence the term "expander"), in which the decision regions can be made spherical without significant accuracy penalties. We should note here that in cases in which the classifier is a neural network, the last layer often reduces the dimension of the data – for instance, to fit the number of relevant classes. In such cases, we propose to remove the last (low-dimensional) layer of the neural network before attaching the expander. In the cases in which the neural network presents an extremely wide output, the expander will in fact need to act as a compression mechanism, in order to reduce the dimension of the network's output layer to one that is tractable by the secure sketch – as we shall see shortly, the decoding procedure employed by the secure sketch can be quite time consuming [36] (albeit running in time linear in the code length).

3.1 The Classifier

As mentioned above, with the option of retrofitting the NFE to pre-existing classifiers, the only requirement for the classifier is to process the users' biometric readings into vectors, in a vector space in which classification is possible with reasonable accuracy. We do not expect that retrofitting with the NFE will improve this accuracy in any way – nor is that the purpose of the retrofit. The best we can hope for is that the original classifier's accuracy is maintained, while the security of the system is greatly improved. As such, our NFE scheme can work with multiple types of already-existing classifiers, for example K-nearest-neighbors, support-vector machines, or neural networks. The "*neural*" part of "NFE" refers not to the retrofitted classifier, but rather to the expander, which is invariably implemented as an artificial neural network.

3.2 The Expander

The expander will be constructed as a neural network, will take as input the output of the (trimmed or intact) original classifier, and will be trained using a cost function that penalizes deviations form a spherical shape. If the original classifier is also a neural network, the training of the expander can be done at the same time as that of the original classifier – in essence, the procedure consists of simply adding a constraint on the shape of the decision regions. However, when retrofitting the NFE to an already-trained neural network, separate training of the expander can be accomplished by feeding it with labeled outputs of the (trimmed or intact) original network, corresponding to some (original or novel) training dataset.

3.3 The Secure Sketch

The secure sketch, as defined in [14], consists of (1) a mechanism for mapping the fuzzy biometric of a user to a fixed point in the vector space of the biometric representation, coupled with (2) a secure method for storing such identifying user information. If the fuzzy biometric data of our user is situated in a roughly spherical region of the vector space, then the first part can be accomplished by defining an error-correction code over the vector space, and shifting it so that the user's decision sphere overlaps with one of the decoding regions, corresponding to one of the codewords.

This construction is described intuitively in Figure 2. In the left-most part of the figure, we can see that the authentic user's (AU's) biometric data-points (represented as light blue triangles) register (mostly) inside the bottom-right red sphere – which is the user's *decision region.* The radius of this sphere is user-specific, and has to be chosen to provide a good compromise between false positives and false negatives. The large circle is chosen to contain all available embeddings, from all users; in the general multi-dimensional case, this circle corresponds to the hyper-sphere that represents the support of the expander's output – basically, the region of the vector space in which one would expect to find data points corresponding to the embeddings of any user's biometric data.

Registration Phase

For the registration phase, a codebook is defined over the vector space, and restricted to the support of the expander's output. Different methods can be used to define such a code, but an optimal choice would be a capacity-achieving code (for an additive White Gaussian noise (AWGN) channel, if the vector space is of the form $\mathbb{R}^n$, and for a binary symmetric channel if the vector space is defined over $GF(2^n)$). For our specific example, in which embeddings are defined over $\mathbb{R}^{128}$, we choose a low-density lattice code (LDLC) [49] in 128 dimensions. This codebook is made up of a set of codewords, which are represented by the black dots in the middle portion of Figure 2. Each such black dot is surrounded by its decoding sphere (or *Voronoi* region). Any data point (represented as a vector) falling inside a particular codeword's Voronoi region can be decoded to this particular codeword (which is that codeword in the codebook that is closest to the data point).

Next, we identify the center of the AU's decision region – denote it as $\mathbf{r}_i$ – and "decode" it to the closest codeword in the codebook – denote this codeword as $\mathbf{c}_i$. We then calculate the difference vector (DV) $\mathbf{d}_i = \mathbf{r}_i - \mathbf{c}_i$ between the center of the AU's decision region and the closest codeword (a translated copy of $\mathbf{d}_i$ is shown as the small yellow arrow in the bottom-right part of the middle part of Figure 2). The DV $\mathbf{d}_i$ is stored as part of the AU's authentication record, along with the hash $h(\mathbf{r}_i)$ of the center of the AU's decision region.

Verification Phase

For each of registered user i, the verifier has access to their tuple $(\mathbf{d}_i, h(\mathbf{r}_i))$. Upon verifying a user's claim to be user i, the current biometric reading – denote it as $\mathbf{b}$ is placed (by virtue of the expander-enhanced classifier) into the vector space, and the DV is subtracted from it. Let the result be $\mathbf{f}_i = \mathbf{b} - \mathbf{d}_i$. In the right part of Figure 2, $\mathbf{b}$ is represented as a light-blue triangle, while $\mathbf{f}_i$ is the tip of the upper yellow arrow. The vector $\mathbf{f}_i$ is then decoded to the closest codeword – denote this codeword as $\mathbf{c}_j$ and note that if $\mathbf{b}$ is indeed the biometric of user i, then we should have $\mathbf{c}_j = \mathbf{c}_i$. Finally, the DV $\mathbf{d}_i$ is added to this codeword $\mathbf{c}_j$ (in an attempt to recover the center of the AU's decision region), and the hash of the result $h(\mathbf{c}_j + \mathbf{d}_i)$ is compared to the one in user i's record $h(\mathbf{r}_i)$. The user is authenticated if $h(\mathbf{c}_j + \mathbf{d}_i) = h(\mathbf{r}_i)$.

NOTE 1: In general, instead of *decoding* the center of the AU's decision region to the closest codeword, we could simply choose a random codeword, and calculate the DV between the center of the AU's decision region and this codeword. But in this case, additional steps have to be taken to ensure that the DV does not leak any information about the center of the AU's decision region. For example, if we choose a codeword in the upper-left of the large sphere in Figure 2, the DV has a large amplitude, and an attacker could infer that the center of the AU's decision region is in the lower right. To avoid such leakage, we would need to add to DV an additional random vector, the effect of which is neutral when wrapped around the large sphere (zero modulo the large sphere).

NOTE 2: It may appear at a first glance that the system can be further simplified by hashing directly the (closest, or randomly-chosen) codeword. However, with such an implementation, care must be taken to ensure that different users are assigned different codewords, which results in a net increase of the system complexity. Using the extra steps required to store the hash of the center of the AU's decision region as above will naturally ensure that different users have different hashes in the authentication table.

3.4 The Hash

The hash component of the NFE architecture is implemented as a simple cryptographic hash function, to be chosen according to the most recent NIST recommendations. At the time of this writing, hash functions from the SHA-2 and SHA-3 families would be perfectly adequate. The user authentication database stores, along with user names, the following user identifying components: (1) codebook parameters (or just a decoding algorithm), (2) difference vector (DV), as explained in Section 3.3 above; (3) any non-secret randomness such as salt and pepper used during the hashing process, and (4) the hash value (with salt, pepper, etc.) of the center of AU's decision region, as explained in Section 3.3 above.

3.5 Security Considerations

Threat Model

We assume a biometric-based authentication system, implemented with the help of an artificial neural network on an authentication server (which, as a particular case, and in a broad sense, can be hosted on the local machine to which users attempt to log in). Each of the legitimate users of the system undergo the registration phase, in which multiple readings of their biometric feature(s) are taken and used to train the authentication system. The trained authentication system, as well as the users' authentication records, are stored (possibly encrypted) on the authentication server. We assume no malicious interference during, or eavesdropping of, the registration phase – this phase can be completed in a protected environment, just like the one required when users set up their new accounts and passwords on any password-based authentication system. Further, we assume that the attacker cannot observe an authentication procedure in real time, and hence does not have direct access to the biometric information entered by the user. In reality, such access is possible if the attacker eavesdrops the connection between authentication server and biometric sensor, or if the attacker has admin privileges and scans the memory during the authentication process. We should note that all of these assumptions are standard in biometric authentication, and any efforts to strengthen the authentication system for the event when any of these assumptions fail are compatible with, but outside the scope of, our work.

The threat model consists of an attacker who can recover the weights of the ANN-based authentication system, as well as all users' biometric records. The attacker has access to this information only after the registration phase is complete, and has no further access to the authentication server, except potentially as a regular legitimate user of the system. Further, the attacker has ready access to technology that can produce exactly one biometric reading that is fully under the control of the attacker, and cannot be distinguished from a real biometric reading from a real person. The restriction to exactly one biometric reading is required to avoid the situation when the attacker brute-forces the biometric

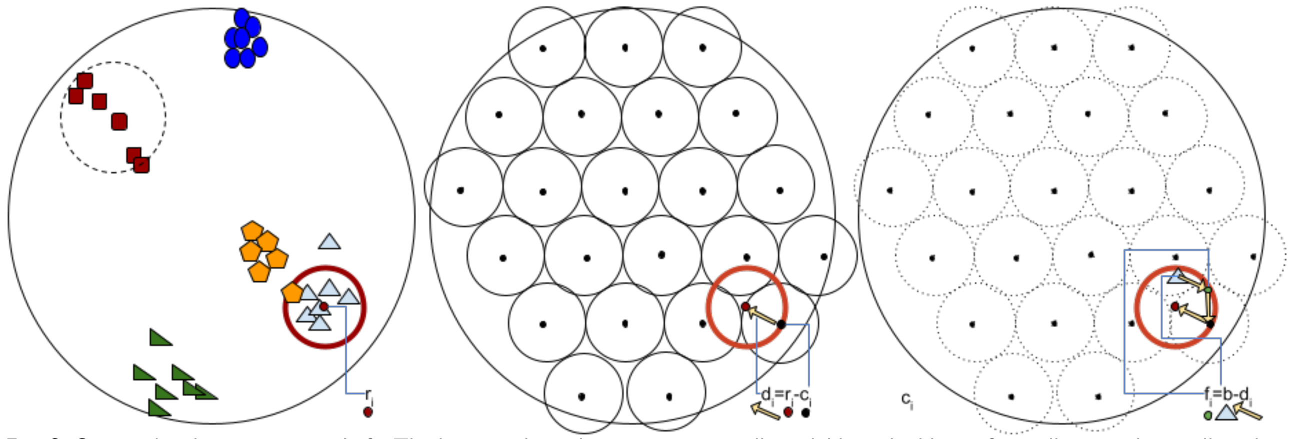


Fig. 2. Secure sketch construction. Left: The large circle is chosen to contain all available embeddings, from all users; the small circle for the authentic user (AU) is chosen to yield a favorable false positive-false negative compromise (different radii may be chosen for different AUs). Middle: A codebook is constructed, with Voronoi region congruent to AU's decision region; the center of AU's decision region (r_i) is decoded to the closest codeword (c_i), and the difference between the center of AU's decision region and this codeword (d_i) is saved to the AU's record along with the hash of the center of AU's decision region. Right: The AU submits a new sample for authentication (b); by subtracting the difference vector (d_i) and decoding to the nearest codeword (c_i), the previously-identified codeword is recovered. We then add the difference to the recovered codeword (again d_i), and obtain the center of AU's decision region (r_i); its hash is compared to AU's record.

authentication system – the effort required by a biometric brute-force attack depends only on the entropy of the biometric, and has little to do with the authentication mechanism that is using it, and the security of which this paper aims to improve.

The attacker's goal is to complete a successful login attempt as any one of the system's legitimate users – of course, with the exception of the user that the attacker already owns or has compromised. The attacker wins if they can log in as a (different) legitimate user of the system, with probability significantly higher than a non-legitimate random person attempting to log in by properly providing biometric readings to the authentication system.

Security Evaluation of the NFE

In order to log in as a (different) legitimate user of the system, the attacker would have to input the user name of a user of their choice (the attacker has a list of all user names from the authentication records), and to produce a biometric reading which, when passed through the authentication mechanism, yields the same output as the one in the authentication record associated to the chosen user name.

In the case of a standard ANN-based authentication mechanism, this is feasible, as the output corresponding to each user is merely a well defined set (usually a region of the output vector space) of scores – for instance, the authentication mechanism decides that the input belongs to user 1 if the first component of the output exceeds a certain threshold. The attacker can then choose an output vector in this set, and work backwards towards the input of the ANN classifier. The recovered input of the classifier (note that usually a continuum of such valid inputs exist) can then be chosen as the biometric reading. Recall that under our threat model above, the attacker can force the authentication mechanism to consume the biometric reading of their choice.

In the case of the NFE-enhanced classifier, the chosen user's authentication record consists of a user name, a difference vector and a hash value. The difference vector contains no information about the pre-image of the hash value. Neither can such a pre-image be found from the hash value, as long as the hash function is pre-image resistant. Therefore, the attacker has no idea about what the output of the expander should be for the chosen user. The best the attacker can do is to choose randomly one of the codewords, and add to it the difference vector located in the chosen user's authentication record, in the hope that the result is actually the center of the chosen user's decision region. The attacker can then work back through the ANN+expander to produce a biometric reading corresponding to this result. Recall however that the codebook is designed such that the number of codewords is the same as the number of possible distinguishable user profiles. Therefore, choosing a codeword from the codebook at random is no better than choosing a person at random, and asking them to provide a biometric reading. Now the probability that the attacker can successfully log in as a different user is equal to the probability that a non-legitimate random person can log in as that user by providing their own biometric readings, plus the probability that the attacker can find a pre-image for the user's hash value. Hence, within our threat model, and assuming a pre-image-resistant hash function, the attacker cannot win.

NFE Codebooks, Sphere Packings, and the Biometric Entropy

The purpose of the NFE is to cast the embeddings of different users' biometric readings in quasi-spherical regions in a certain vector space. In the left portion of Figure 2 we show multiple such quasi-spherical clusters, such that each cluster (represented by markers with a specific shape and color) corresponds to the embeddings of the available biometric readings for a single user. In this figure, we decide to consider the support of the expander output as a sphere that includes all the available biometric data points. Taking the smallest such sphere may artificially reduce the perceived security of the biometric authentication system – because it will result in a smaller codebook – so, to be safe, consider a sphere of radius 10% larger than that of the smallest outer sphere.

The radius of the user-specific decoding region (the small sphere corresponding to the user of interest in Figure 2) is usually chosen to provide a certain false-positive-false-negative tradeoff. For simplicity of implementation, we choose the same radius (technically, the same error-correction code) for all users.

As an example, for the VGG-16 architecture retrofitted with the NFE, the FVC2006 database yields outer sphere of radius 1.0153, and a small sphere of radius 0.7 (this radius is chosen such that the distance-based decoder achieves 95.36% accuracy on the training data set), meaning that at most $(1.0153/0.7)^{128} \simeq 4.7 \cdot 10^{20}$ small spheres can be packed inside the large 128-dimensional sphere. We could say that the maximum number of distinct individuals whose biometric reading embeddings in our new vector space fit within non-overlapping small spheres is thus about $4.7 \cdot 10^{20} \simeq 2^{68.67}$, or that the entropy of the secret biometric information, as represented in this space, is about 68.67 bits. In other words, the proposed methodology for the expansion of biometric data provides – as a byproduct – a way of evaluating the authentication potential of various types of biometric data. It would be possible in this framework to decide (at least approximately, based on the upper and lower bounds on the entropy of a user's biometric data) whether fingerprint-based biometrics, for instance, are more or less secure than, say retina-scan-based biometrics. This evaluation is related to the average (across multiple possible AUs) accuracy of the associated classifier, but is not straightforward to derive from it, and would not be feasible in the absence of the expander.

We should note here that previous efforts to quantify the entropy of various types of biometrics take different approaches. For example, [65] produces empirical distributions of fingerprint data, and yields a measure of entropy per pixel for different fingerprint databases. Their FVC2002 and FVC2004 databases are comparable to our FVC2006 database. However, their entropy upper bounds – slightly above 0.25 bits per pixel – would yield entropies in the order of 50,000 bits for each image. Note that this is the entropy of the fingerpint image, not the entropy of the fingerprint-based biometric modality that we calculate. To approximate the latter, [65] uses the mutual information between fingerprint images, and ends up with a maximum number of distinct individual representations of 10^{28} – making our estimate of $4.7 \cdot 10^{20}$ somewhat conservative. Other methods for evaluating biometric entropy are introduced in [34] and [52].

4 An Instantiation of NFE Retrofit

To instantiate our NFE retrofit architecture described above, we had to make several design choices. First, we chose three pre-existing classifier architectures, as described in Section 4.1 below, as the basis for the retrofit. We had to slightly modify each one of the architectures. Next, we used an expander architecture consisting of multiple fully-connected layers, of decreasing sizes, to reduce the original classifier output size to a fixed size of 128 neurons. This was done to accommodate a tractable decoding mechanism. The error-correcting code used to provide the decoding mechanism was chosen to be a 128-dimensional low-density lattice code taken directly from [49]. The rest of this section contains additional details regarding our design choices.

4.1 Classifier and Expander Architecture

VGG16

The VGG16 [22, 48] weights are pre-trained on ImageNet using Keras. To adapt this classifier to fingerprint-based user authentication, we removed the final softmax layer, rendering an output of size 4608. The size of the vector space (4608) is way too large to match to any efficient decoding mechanism. Therefore, to retrofit the NFE to this classifier, we constructed an expander, by adding three more fully connected layers of 512, 256 and 128 neurons respectively.

ResNet50

The ResNet50 [23] weights are pre-trained on ImageNet using Keras. After removing the final softmax layer, we were left with an output of size 2048 by average pooling the last layer. Then we constructed an expander by adding 4 fully connected layers of size 1024, 512, 256, and 128, respectively.

MobileNet

We used a modified MobileNet v1 model architecture [28] with weights pre-trained on ImageNet using Keras. After removing the final softmax layer, we were left with the output of 1024 by average pooling the last layer. Then we constructed an expander by adding 3 fully connected layers of size 512, 256 and 128, respectively.

One Shot Classification and Siamese Network

In the case of standard identification methods, a set of images are fed into an ANN to get an output probability for each one of the different classes. For example, if we want to distinguish between a cat and a dog we want to collect a lot of images (possibly more than 500 images per class) to improve model accuracy. The drawback of this type of network in fingerprint identification is **first**, it is nearly impossible to get a lot of images and **second**, if we want to include a new user in our database, we need to retrain the model to identify the new user as well. It is for these reasons that we choose to train our classifier in a Siamese network configuration, as explained below.

It should be noted that for this specific application, our “expander” is in fact not expanding at all, but rather contracting the ANN’s output. This is to reduce the complexity of the decoding involved in the secure sketch. Nevertheless, the same exact principles apply in situations in which the expander actually expands the output size.

A siamese network (sometimes called a twin neural network) is an ANN which learns to differentiate between two inputs instead of classifying. It takes two input images, runs through the same network simultaneously, and generates two vector embeddings of the images which are run through a logistic loss to calculate a similarity score between the two images [9]. This is very useful as it does not require many data points to train the model. For training purposes, we only need to store one image belonging to the legitimate user as a reference image, and calculate the similarity for every new instance presented to the network.

For our implementation, we used a triplet loss function with the Siamese networks. The benefit of using a triplet loss function (explained in the next subsection) in conjunction with a Siamese network is twofold [15]:

1. It extracts more features by learning to maximize the similarity between two similar images (Anchor-Positive) and the distance between two different images (Anchor-Negative) at the same time.
2. It generates more training samples than logistic loss. If we have P similar pairs and N dissimilar pairs then for logistic loss we will have $P + N$ total training samples. Whereas, we will have PN triplets for training. This will impove the model accuracy.

Our Siamese network architecture is depicted Figure 3. It is interesting to note that Siamese network training will naturally encourage the embeddings of the data points to cluster in spheres. This is a direct consequence of its loss function that causes embeddings from the same class to be close together, and embeddings from different classes to be well separated in terms of Euclidean distance. Nevertheless, under different circumstances, with more available training data, other expander architectures can be used, as long as their loss functions are adjusted to include similar distance-based penalties.

Triplet Loss Function

Triplet loss functions are widely used in various applications in computer vision, such as face recognition [47], person re-identification [24] and image retrieval [20]. Taking inspiration from that we used this for fingerprint verification. We will further explain how triplet loss functions work.

If we use a CNN to convert an image x into a d-dimensional Euclidean space, then the embedding is represented by $f(x) \in \mathbb{R}^d$. Here f is the function computed by the CNN. For training we used triplets of fingerprint images, as shown in Figure 4:

– A is an "anchor" image – a fingerprint image of a user.
– P is a "positive" image – a fingerprint image of the same user.

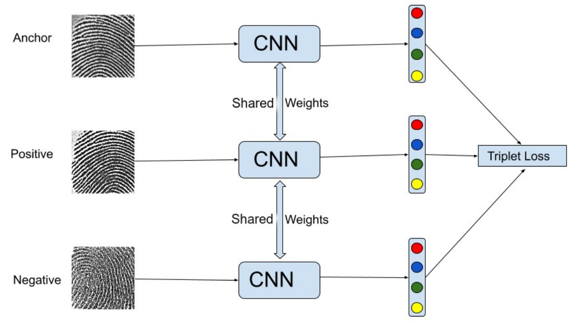


Fig. 3. Triplet Loss architecture with Siamese network: Three images ("Anchor","Positive" and "Negative") are passed through the same CNN simultaneously to generate a final layer of 128 dimensional vector. Then all three vectors are passed through the triplet loss function to minimize the distance between "Anchor" and "Positive" as well as maximizing the distance between "Anchor" and "Negative".

– N is a "negative" image – a fingerprint image of a different user.

We write triplets as $(A^{(i)}, P^{(i)}, N^{(i)})$ where i denotes the i^{th} training example. We want to make sure that P is closer to A than N. Thus, we want

$$\|f(A^{(i)}) - f(P^{(i)})\|_2^2 + \alpha < \|f(A^{(i)}) - f(N^{(i)})\|_2^2$$

for all $\{f(A^{(i)}), f(P^{(i)}), f(N^{(i)})\} \in T$, where T is the set of all possible triplets. Here, we want to make sure that the positive pair $(A^{(i)} - P^{(i)})$ has at least a margin difference of α over the negative pair $(A^{(i)} - N^{(i)})$. So the "triplet cost" function for the CNN becomes

$$\sum_{i=1}^{n} \left[\|f(A^{(i)}) - f(P^{(i)})\|_2^2 - \|f(A^{(i)}) - f(N^{(i)})\|_2^2 + \alpha \right]_+ .$$

Generating all possible triplets for training will result in slower convergence, so it is important to select a combination of "hard" and "easy" batches for the improvement of the model.

4.2 The Error-Correction Code

As mentioned in Section 3.3, for our error-correction code we used a low-density lattice code (LDLC), picked directly from [49], in $n = 128$ dimensions. As explained in [49], the LDLC is completely defined by its sparse parity-check matrix H. To find an appropriate parity-check matrix, we used the *Latin Square* construction

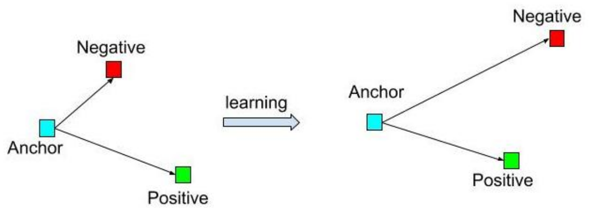


Fig. 4. Triplet Loss architecture: The architecture tries to minimize the distance between "Anchor" and "Positive" and maximize distance between "Anchor" and "Negative"

of [49], in which every row and every column of H contains the same d nonzero values. The d nonzero values were selected as the first d values in the sequence $\{1/2.31, 1/3.17, 1/5.11, 1.7.33, 1/11.71\}$. The allocation of these values to the rows and columns of H was done in accordance with the algorithms presented in Appendix VII of [49]. We experimented with $d = 3$ and $d = 5$. The error-correction performance of the two codes (for $d = 3$ and $d = 5$) are shown in Figure 5, where every point is obtained by applying the decoding algorithm 20 times to a randomly-chosen codeword distorted by additive white Gaussian noise of standard deviation σ (shown on the horizontal axis). We can see from the figures that the performance of the two codes is very similar in this artificial setting, with the $d = 3$ code performing slightly better. This is also the case in practice, as illustrated next by the performance of the NFEs using the two codes.

We should note here the difference between binary codes and lattice codes. Binary error-correction codes use a codebook formed by a select few, sufficiently distant (usually in the sense of Hamming distance) binary sequences of a given length n, and all other binary sequences can be decoded to the closest sequence in the codebook. The rate of the binary code is controlled by the ratio k/n, where k is the length of the message to be encoded (with $k < n$). The generator matrix of the code is thus a rectangular one, of size $n \times k$. One can therefore adjust the density of the codebook in the space $\{0, 1\}^n$ by adjusting k or n or both. This helps when trying to fit a good codebook to a constellation of binary fingerprint representations from different users – one can simply find the centroids of the users' fingerprint clouds, and produce a code with roughly the same density.

By contrast, LDLCs considered in this paper represent messages as vectors of n integers, that are then mapped to codewords by a square $n \times n$ generator matrix, which is the inverse of the sparse (low-density) parity-check matrix. Decoding involves finding the vec-

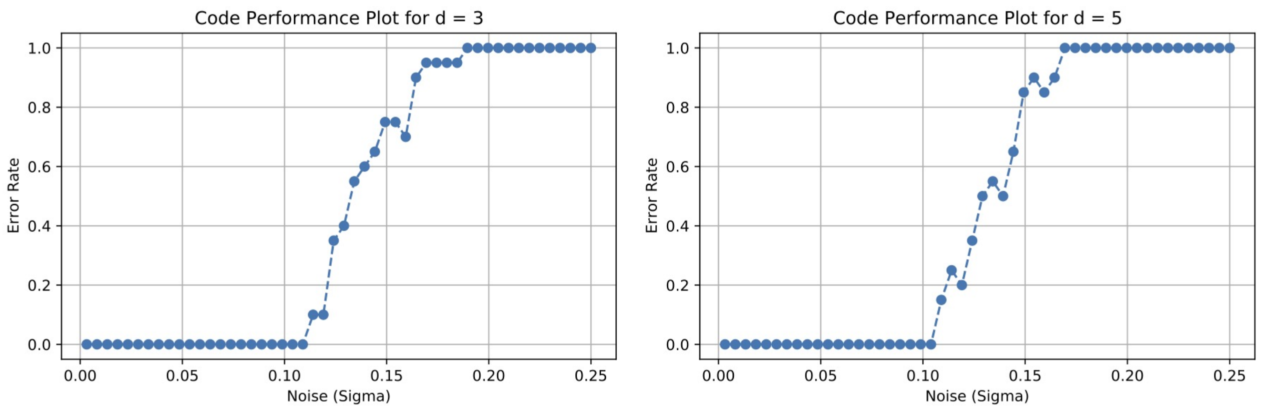


Fig. 5. Error rate vs. injected noise level for the Latin-Square LDLC with row/column degree d of the parity check matrix.

tor of integers that is most likely to have generated the received sequence. Therefore, the information rate of LDLCs is only meaningful when an average power constraint is imposed on the codewords. This is not the case for our application. Instead, it was shown in [45] that more meaningful performance metrics for infinite-constellation codes (like LDLCs), and ones that do not depend on power constraints, are directly related to the *constellation density*. For our LDLC code, the constellation density is fixed. Thus, we cannot fit the codebook to the constellation of fingerprint representations – instead, we have to fit the fingerprint constellation to the codebook. We do this by scaling all fingerprint representations by the same *scale factor* γ. Choosing different scaling factors produces different false-positive rate (FPR) to false-negative rate (FNR) tradeoffs, thus enabling us to construct the receiver operating characteristic (ROC) curves.

Interestingly, a very similar scaling by γ also enables us to produce different FPR-FNR tradeoffs when distance (norm) based decoding is used, by keeping the decoding distance threshold constant and varying γ.

4.3 Some Guidelines for System Configuration

Finally, before moving on to the performance and cost evaluation of our NFE implementation, we summarize several system configuration aspects of our NFE-retrofitted fingerprint-based biometric authentication, to serve as a quick reference for implementation.

Number of Users

There is no limit on the number of users in practice. Recall our (conservative, compared to [65]) estimate of a maximum of $4.7 \cdot 10^{20}$ differentiable users. Clearly, if this many users were registered with the authentication server, then any input biometric would most likely correspond to at least one legitimate user. Fortunately, the world's population is still far below this number, and an authentication attempt would not only have to provide the biometric reading, but also the associated user name, making a successful authentication attempt very unlikely.

Original Classifier Preparation

To prepare the original neural-network-based classifier for the NFE retrofit, we recommend removing the final softmax layer, if such a layer exists. The size of the output after this trimming procedure should be large enough to prevent the loss of biometric input information due to excessive compression.

Codeword Size

The appropriate codeword size n depends on the efficiency of the decoding algorithm and on the acceptable computation overhead during user authentication. Based on the capabilities of consumer-grade hardware at the time of this writing, we recommend to choose n in the range of $[100, 500]$. See [61] for more detailed results on the computational overhead as a function of n, for an efficient LDLC decoder. Also, as noted above, setting $d = 3$ should provide very acceptable performance in this range.

Setting the FPR-FNR Tradeoff

As mentioned above, controlling the FPR-FNR tradeoff of the LDLC code is done not by modifying the code (whose constellation is fixed), but by scaling the NFE's output by a scalar factor γ. When implementing our NFE, we recommend trying multiple values of γ to build an approximation of the ROC curve, and then choosing an operating point on this curve.

5 NFE Evaluation

To evaluate our NFE retrofit instance, we had to make several experimental design choices. First, we selected two databases of fingerprints, of different sizes. Next, we defined four different experiments, to allow us a comparison between the various retrofitted architectures, and two different baseline architectures, for both databases, and all three classifiers. Finally, we experimented with multiple ways of tuning the pre-trained (and slightly modified, as explained in Section 4 above) classifiers to our databases, and chose the best-performing tuning modalities. The remainder of this section provides additional details on these design choices, as well as the experimental results.

5.1 Datasets

We used two datasets for our evaluation process: the 2006 Fingerprint Verification Competition (FVC2006) Database [57] and the PolyU fingerprint database provided by Hong Kong Polytechnic University [60].

The FVC2006 Database

This database consists of 4 distinct subsets DB1, DB2, DB3 and DB4. Each database consists of 150 fingers and 12 impressions per finger. Each subset is further divided into "set A" and "set B" where "set A" contains 140x12 images and "set B" contains 10x12 images At the time of this writing we only used DB1, which has an image size of 96x96 but we expect similar results with the other databases. The images in the FVC2006 database were collected as part of the European Project BioSec, from anonymous volunteers, and was released publicly as part of the 2006 Fingerprint Verification Competition, organized by The Biometric System Laboratory (University of Bologna), the Pattern Recognition and Image Processing Laboratory (Michigan State University), the Biometric Test Center (San Jose State University) and the Biometrics Research Lab - ATVS (Universidad Autonoma de Madrid) [57]. At the time of this writing, we have no reason to doubt the ethical aspects of the data collection process.

The PolyU Database

This database contains 2016 fingerprint images from 336 different users, i.e., 6 impressions per user. The size of the image is 328x356 initially. To make it consistent with our initial dataset, we augmented the images using several data augmentation techniques like the addition of random noise and random rotation. In this way, we generated 6 additional impressions per user, bringing our dataset to 12 fingerprint images per 336 users and converted the size of these images to 224x224. The PolyU database was released in 2017 for public use by the The Hong Kong Polytechnic University. At the time of this writing, we have no reason to doubt the ethical aspects of the data collection process.

Training Data

For either one of the databases, training data consisted of 10 impressions per finger (total of 140x10 images for the FVC2006 database, and a total of 336x10 images for the PolyU database). Out of these images, we generated the triplet pairs. In this paper, we used 50% "hard" and 50% "easy" triplet pairs.

Testing Data

For either of the two databases, testing data consisted of 2 impressions per finger.

5.2 Experimental Setup

We conducted four experiments using ResNet50, MobileNet and VGG16 architectures with two different datasets FVC2006 and PolyU.

Baseline Classifier (BC)

Our first experiment is meant to provide a baseline for the evaluation of the expander-enhanced architectures. For this purpose, we trained the three classifier architectures (ResNet50, MobileNet and VGG16) separately. Since we used two datasets – FVC2006 with 140 users and PolyU with 336 users – our classification model's top layer was a softmax layer with 140 nodes in the case of FVC2006 and 336 nodes in the case of the PolyU dataset, respectively.

The final layer of the classification model gives us the probabilities (or, more generally, some scores for the events) that the input image belongs to each possible user. We transform the model into a binary classifier by establishing a score threshold for each one of the users. Hence, to calculate the false positive rate for a single class, we took 20 different fingerprint images from other classes fed them through the network. If the score output value for the image is greater than the probability threshold for that particular class we are interested in, then that's the condition of false positive.

Similarly, to calculate the FNR for a single class, we took 2 images for that class from the test set, augmented them using several data augmentation techniques like addition of random noise and rotation to generate another 20 images and fed them through the network. If the output probability value of the node of the class that we are concerned with is less than the probability threshold, then that's the condition of false negative.

Classifier Trained in a Siamese-Network Configuration (CSN):
In the second experiment, we trained the three classifiers individually, using a Triplet-based Siamese-network configuration. No expander was used in this experiment, and thus the size of the final embeddings vary according to the architecture: ResNet50 uses an embedding of size 2048 (the size of the final layer before the softmax layer), MobileNet uses an embedding size of 1024, and VGG-16 uses a size of 4608. This experiment was designed to provide a fair baseline comparison for the expander-enhanced architectures, just in case that this Siamese-network configuration training proved overall superior to the standard training of experiment BC above – it turns out that its superiority is in fact classifier-dependent.

Classifier Trained in a Siamese-Network Configuration Plus Expander (CSN+ESN)
In the third experiment, we use the classifiers already-trained under the second (CSN) experiment, and retrofit them with expanders. Then, only the expanders are trained, using the classifier-plus-expander architectures in the standard Siamese-network configuration. This experiment emulates a scenario in which an existing neural network-based classification architecture is already trained and provides good performance by itself. In such a situation, training only the expander while keeping the original classifier's weights fixed can provide significant computational savings.

Jointly Trained Classifier and Expander ((C+E)SN)
In this last experiment, we first retrofitted all three classification architectures with an expander, and trained the classifier-plus-expander architectures from scratch, using the standard Siamese-network configuration. Our initial expectation was that such joint training of the classifier and expander would provide the best performance. In reality, the results show that this is in fact classifier-dependent.

For each of the CSN+ESN and (C+E)SN experiments, we execute, and report the results of, two types of error-correction decoding: (a) using simple fixed-distance-based decoding, and (b) using the LDLC decoder. The purpose of the distance-based decoder is solely to provide a baseline for evaluating the performance degradation due to the LDLC decoder (however, in reality, as our results will show, the LDLC decoder usually out-performs the distance-based decoder). The distance-based decoder is not a viable option in practice, as it would require that the center of the AU's decision region is saved as part of the AU's authentication record. This would defeat the purpose of the NFE.

5.3 Training of the Classifiers

VGG-16
For VGG-16 architecture, we trained all its layers for the both baseline and the Siamese-network-based configurations, to get the better generalization of the model.

ResNet50
For ResNet50 architecture, in the Siamese-network-based configuration, we froze the first 143 layers and trained the remaining 32 layers to perform the transfer learning. However, in the case of the baseline model, we trained the whole architecture without freezing any layers.

MobileNet
For the MobileNet architecture in the Siamese-network-based configuration, we trained just the last 23 layers. However, in the case of the baseline model, we trained

all the layers of the architecture to get a better generalization of the model.

5.4 Results

We expect that our NFE retrofit contains two sources of performance degradation, and we proceed to systematically investigate each. The first potential source of performance degradation is the injection of the expander into the system's architecture. This should be especially critical when the expander in fact reduces the output size of the original architecture, rather than expanding it. This is exactly the case with our implementation, where the output size is reduced from 4608, 2048 or 1024 to 128.

The second source of performance degradation is the use of an error-correction code in place of distance-based decoding. This step is essential to the security of the NFE architecture, as it enables automatic decoding in the absence of a class representative (or centroid). Such a representative is required by distance-based decoding, and constitutes a severe security vulnerability.

The Effect of the Expander

To evaluate the impact of the expander, we compared the performance of the three classifier architectures in the BC and CSN experiments, to the NFE-retrofitted architectures – experiments CSN+ESN and (C+E)SN – but performing distance-based decoding instead of LDLC-based decoding. For our two datasets, the results are given in Tables 1 and 2.

Experiment	ResNet		MobileNet		VGG-16	
	EER	AUROC	EER	AUROC	EER	AUROC
BC	**0.032**	**0.972**	**0.022**	**0.969**	**0.040**	**0.940**
CSN	**0.065**	**0.982**	**0.055**	**0.989**	**0.014**	**0.999**
CSN+ESN distance	**0.054**	**0.986**	**0.044**	**0.994**	**0.013**	**0.990**
CSN+ESN LDLC	**0.053**	**0.987**	**0.025**	**0.995**	**0.013**	**0.999**
(C+E)SN distance	**0.066**	**0.984**	**0.042**	**0.993**	**0.020**	**0.997**
(C+E)SN LDLC	**0.065**	**0.984**	**0.039**	**0.993**	**0.020**	**0.995**

Table 1. Equal error rates (EER) and areas under the receiver operating characteristics (ROC) curve (AUROC) for our four experiments, under each of the three architectures, when using the FVC2006 database.

We notice that for the FVC2006 database (Table 1), with small exceptions, all three architectures exhibit comparable EER values under the *CSN* experiment, with the *CSN+ESN distance* and *(C+E)SN distance* experiments. However, it appears that the ResNet50 and MobileNet architectures in the *BC* experiment exhibit significantly lower EER values, albeit also smaller AUROC values, than in the other two experiments. This shows that some particular performance degradation around the EER point may be caused by the training configuration, i.e. training under the baseline configuration outperforms training under the Siamese-network-based configuration. However, considering the AUROC values, such performance degradation is localized around the EER point.

The opposite holds true for the VGG-16 architecture, where the BC experiment shows higher EER – and this time also lower AUROC – than all the other experiments.

Experiment	ResNet		MobileNet		VGG-16	
	EER	AUROC	EER	AUROC	EER	AUROC
BC	**0.031**	**0.958**	**0.017**	**0.979**	**0.062**	**0.972**
CSN	**0.035**	**0.995**	**0.025**	**0.996**	**0.013**	**0.999**
CSN+ESN distance	**0.037**	**0.992**	**0.026**	**0.995**	**0.007**	**0.999**
CSN+ESN LDLC	**0.037**	**0.992**	**0.025**	**0.995**	**0.007**	**0.999**
(C+E)SN distance	**0.059**	**0.985**	**0.054**	**0.986**	**0.033**	**0.993**
(C+E)SN LDLC	**0.058**	**0.985**	**0.054**	**0.981**	**0.034**	**0.981**

Table 2. Equal error rates (EER) and areas under the receiver operating characteristics (ROC) curve (AUROC) for our four experiments, under each of the three architectures, when using the PolyU database.

The same difference in performance between the *BC* and *CSN* experiments is observed for the PolyU dataset (Table 2), where again it appears that ResNet50 and MobileNet do significantly better in the BC experiment, while VGG-16 does significantly worse.

Interestingly, on both datasets, all architectures do much better in the *CSN+ESN distance* experiment than in the *(C+E)SN distance* experiment, suggesting that training of the expander alone, rather than joint training of the expander and original classifier, is the better choice.

The Effect of the Decoder

To evaluate the impact of the error-correction-code on the NFE performance, we compare the results in the experiments *CSN+ESN distance* and *(C+E)SN distance* to the experiments *CSN+ESN LDLC* and *(C+E)SN LDLC*, respectively. We notice that for both datasets, and for each one of the classifiers, the EER and AUROC values appear very similar between the distance-based decoding and the LDLC-based decoding. If anything, the LDLC-based decoder seems to perform very slighly better. One notable exception is the case of the MobileNet architecture, on the FVC2006 dataset (Table 1), where the LDLC decoder appears to help a lot, lowering ERR from 4.4% (for the case of *CSN+ESN* distance-based decoding) to 2.5%. Such improvements in performance due to LDLC decoding may be explained by the fact that the LDLC decoding regions are not perfectly spherical (even in 128 dimensions), and fill up the vector space better than perfectly spherical regions (which are implicit with distance-based decoding).

Overall Effect of the NFE Retrofit

Connsidering the best performance among the *BC* and *CSN* experiments, and the best performance among the *CSN+ESN LDLC* and *(C+E)SN LDLC* experiments, we draw the conclusion that there appears to be no, or very slight, performance degradation due to the use of the NFE. This is an encouraging result, and should motivate the future evaluation of the application of NFEs to other architectures, and to other biometric authentication modalities.

We can also notice that overall, the PolyU Dataset performed better than the FVC2006 dataset in most scenarios. This might be because PolyU images are of bigger resolution and have less noise than FVC2006 images.

Training and Running Times

We evaluated the training times and the running times of the three different architectures, under each one of our four experiments and two datasets. All our models were trained on a GeForce RTX 2080 TI GPU with 4 cores, with the maximum memory size at 25GB. For studying the runtime performance, we ran our user authentication mechanisms on a more realistic work station – specifically, a personal laptop with Apple's M1 silicon chip, and with 16GB of memory.

The execution times are listed in Tables 3 and 4 for the FVC2006 and the PolyU datasets, respectively. For training, we list in parentheses the number of training epochs that were necessary for each one of the experiments to observe the convergence of the training process. We note that for the BC experiment, convergence is observed a lot sooner than for the other experiments (fewer than 80 epochs, compared to over 4000 epochs). This is most probably due to the Siamese-Network configuration, which is employed by all the other experiments.

Experiment		ResNet50	MobileNet	VGG-16
BC	training	7m11s (60 ep)	7m14s (80 ep)	8m48s (70 ep)
	runtime	70ms	48ms	55ms
CSN	training	1h53m22s (5000 ep)	1h06m12s (5000 ep)	1h05m39s (5000 ep)
	runtime	529ms	264ms	173ms
CSN+ESN	training	1h50m20s (5000 ep)	1h08m13s (5000 ep)	1h00m41s (5000 ep)
	runtime	645ms	343ms	134ms
(C+E)SN	training	1h57m00s (5000 ep)	1h06m30s (5000 ep)	1h08m48s (5000 ep)
	runtime	589ms	325ms	189ms

Table 3. Training and running times for our four experiments, under each of the three architectures, when using the FVC2006 database.

We observe that under the BC experiment, both training and running times are significantly smaller than in all the other experiments. We also note that there are only small differences between the *CSN+ESN* and *(C+E)SN* experiments, and the *CSN* experiment, in both training and running times, and with all three classifier architectures. As expected, training and running for the *CSN+ESN* and *(C+E)SN* experiments in general takes slightly longer, but not by much.

We can therefore infer that the addition of the expander incurs minimal overhead. However, switching from a standard baseline architecture to a Siamese-network-based architecture does incur significant penalties – in the order of a few hours for training, and in the order of hundreds of milliseconds for runtime.

We should note here that the running times reported in Tables 3 and 4 do not include the running times of the decoder for the *CSN+ESN* and *(C+E)SN* experiments. Our implementation of the LDLC decoder is based directly on the method proposed in [49], which is known to be rather inefficient. Many subsequent works have proposed much more efficient techniques for decoding LDLCs, see for example [5, 35, 61, 66]. For example, when considering decoding parameters similar to

Experiment		ResNet50	MobileNet	VGG-16
BC	training	27m50s (50 ep)	21m38s (40 ep)	22m35s (50 ep)
	runtime	81ms	55ms	63ms
CSN	training	3h12m21s (4000 ep)	2h10m08s (4000 ep)	3h26m43s (4000 ep)
	runtime	579ms	214ms	151ms
CSN+ESN	training	3h17m23s (4000 ep)	2h08m06s (4000 ep)	3h09m32s (4000 ep)
	runtime	459ms	227ms	160ms
(C+E)SN	training	3h12m45s (4000 ep)	2h08m46s (4000 ep)	3h22m00s (4000 ep)
	runtime	755ms	240ms	156ms

Table 4. Training and running times for our four experiments, under each of the three architectures, when using the PolyU database.

ours ($d = 3$ and $n \in [100, 1000]$), running on consumer-grade hardware, [61] reports running times of around $4.2ms$ per iteration for $n = 100$ and $36ms$ per iteration for $n = 1000$, leading to total decoding times of between $0.42s$ and $3.6s$ (corresponding to 100 iterations). These decoding times are well within the acceptable range for user authentication. Nevertheless, the integration of efficient LDLC decoders with NFEs is outside the scope of the current paper, and is the subject of future work.

Comparison with Plain Fuzzy Extractors

A performance comparison with the plain fuzzy extractors of [14] is not fair – it has already been established that they suffer from low performance [2, 38]. However, in terms of security and cost comparisons, plain fuzzy extractors appear similar to our proposed NFEs. The security provided by NFEs relies on the secure sketch construction – a concept borrowed from the plain fuzzy extractors, and hence the security guarantees are identical. Since both types of extractors have to store only hash values and difference vectors for each of the registered users, the storage requirements are also similar – of course, NFEs have to also store the weights of the associated neural network. In terms of runtime performance, both NFEs and plain fuzzy extractors have to implement a complex decoding procedure, which accounts for the bulk of the user authentication computation in NFEs, and for almost the entire computation in the case of plain fuzzy extractors.

6 Conclusions

In this paper, we presented a secure architecture for biometric user authentication (or identification), which avoids the storage of information (such as neural network weights or specific biometric data) that could be used by malicious entities for constructing artificial biometric inputs able to pass the authentication tests. To that extent, the proposed architecture aligns with the current paradigm for handling users' passwords. The proposed architecture – which we call a *neural fuzzy extractor* – works by coupling the classifier with a fuzzy extractor – this is possible by the use of an *expander*, which is a neural network that can be trained at the same time as, or after, the classifier. The NFE architecture can be retrofitted to any already-existing well-performing classifier, and should combine the classification performance of artificial neural networks with the security of fuzzy extractors. Our instantiation of the NFE for fingerprint-based authentication demonstrates how a pre-existing classifier can be retrofitted for NFE implementation, and how two different types of training can be conducted for NFE-specific output-space sphere clustering. Our experimental results show that the addition of the NFE does not incur significant performance degradation (in terms of equal-error rates and areas under the receiver-operating-characteristics curves), but does incur higher training and runtime overhead, mainly due to the Siamese-network-based configuration. Nevertheless, we believe that the 1 to 3 hour increases in training times is well worth the security advantage provided by NFEs, while the 100 to 700 millisecond increases in running times is well within the acceptable range for applications. Future work will focus on (1) studies of the application of NFEs to other types of biometrics and (2) the construction and training of expanders suited to coding over binary extension fields.

7 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1619201.

References

[1] Mohamed Mostafa Abd Allah. 2005. Artificial neural networks based fingerprint authentication with clusters algo-

rithm. *Informatica* 29, 3 (2005).

[2] Arathi Arakala, Jason Jeffers, and Kathy J Horadam. 2007. Fuzzy extractors for minutiae-based fingerprint authentication. In *International conference on biometrics*. Springer, 760–769.

[3] Rima Belguechi, Vincent Alimi, Estelle Cherrier, Patrick Lacharme, Christophe Rosenberger, et al. 2011. An overview on privacy preserving biometrics. *Recent Application in Biometrics* (2011), 65–84.

[4] Abhilasha Bhargav-Spantzel, Anna C Squicciarini, Shimon Modi, Matthew Young, Elisa Bertino, and Stephen J Elliott. 2007. Privacy preserving multi-factor authentication with biometrics. *Journal of Computer Security* 15, 5 (2007), 529–560.

[5] Danny Bickson, Alexander T Ihler, Harel Avissar, and Danny Dolev. 2009. A low density lattice decoder via non-parametric belief propagation. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 439–446.

[6] Mark J Brady. 1999. Biometric recognition using a classification neural network. US Patent 5,892,838.

[7] Julien Bringer, Hervé Chabanne, Gérard Cohen, Bruno Kindarji, and Gilles Zémor. 2007. Optimal iris fuzzy sketches. In *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*. IEEE, 1–6.

[8] Julien Bringer, Hervé Chabanne, and Alain Patey. 2013. Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends. *IEEE Signal Processing Magazine* 30, 2 (2013), 42–52.

[9] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*. 737–744.

[10] Luke Nicholas Darlow and Benjamin Rosman. 2017. Fingerprint minutiae extraction using deep learning. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 22–30.

[11] Ashok Kumar Das. 2017. A secure and effective biometric-based user authentication scheme for wireless sensor networks using smart card and fuzzy extractor. *International Journal of Communication Systems* 30, 1 (2017), e2933.

[12] Ashok Kumar Das, Mohammad Wazid, Neeraj Kumar, Athanasios V Vasilakos, and Joel JPC Rodrigues. 2018. Biometrics-based privacy-preserving user authentication scheme for cloud-based industrial Internet of Things deployment. *IEEE Internet of Things Journal* 5, 6 (2018), 4900–4913.

[13] Jeroen Delvaux, Dawu Gu, Ingrid Verbauwhede, Matthias Hiller, and Meng-Day Mandel Yu. 2016. Efficient fuzzy extraction of PUF-induced secrets: Theory and applications. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 412–431.

[14] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. 2004. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *International conference on the theory and applications of cryptographic techniques*. Springer, 523–540.

[15] Xingping Dong and Jianbing Shen. 2018. Triplet loss in siamese network for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 459–474.

[16] Hachim El Khiyari and Harry Wechsler. 2016. Face recognition across time lapse using convolutional neural networks. *Journal of Information Security* 7, 3 (2016), 141–151.

[17] Joshua J Engelsma, Kai Cao, and Anil K Jain. 2019. Fingerprints: Fixed length representation via deep networks and domain knowledge. *arXiv preprint arXiv:1904.01099* (2019).

[18] David Evans, Yan Huang, Jonathan Katz, and Lior Malka. 2011. Efficient privacy-preserving biometric identification. In *Proceedings of the 17th conference Network and Distributed System Security Symposium, NDSS*, Vol. 68. 90–98.

[19] Stephanie Gootman. 2016. OPM hack: The most dangerous threat to the federal government today. *Journal of Applied Security Research* 11, 4 (2016), 517–525.

[20] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. 2016. Deep image retrieval: Learning global representations for image search. In *European conference on computer vision*. Springer, 241–257.

[21] Mohamed Hammad, Yashu Liu, and Kuanquan Wang. 2018. Multimodal biometric authentication systems using convolution neural network based on different level fusion of ECG and fingerprint. *IEEE Access* 7 (2018), 26527–26542.

[22] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[24] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017).

[25] Fernando Hernández Álvarez, Luis Hernández Encinas, and Carmen Sánchez Ávila. 2009. Biometric fuzzy extractor scheme for iris templates. (2009).

[26] Thang Hoang and Deokjai Choi. 2014. Secure and privacy enhanced gait authentication on smart phone. *The Scientific World Journal* 2014 (2014).

[27] Thang Hoang, Deokjai Choi, and Thuc Nguyen. 2015. Gait authentication on mobile phone using biometric cryptosystem and fuzzy commitment scheme. *International Journal of Information Security* 14, 6 (2015), 549–560.

[28] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[29] Wang-Su Jeon and Sang-Yong Rhee. 2017. Fingerprint pattern classification using convolution neural network. *International Journal of Fuzzy Logic and Intelligent Systems* 17, 3 (2017), 170–176.

[30] Taeyun Kim, Yongwoo Oh, and Hyoungshick Kim. 2020. Efficient privacy-preserving fingerprint-based authentication system using fully homomorphic encryption. *Security and Communication Networks* 2020 (2020).

[31] Ruggero Donida Labati, Enrique Muñoz, Vincenzo Piuri, Roberto Sassi, and Fabio Scotti. 2019. Deep-ECG: convolutional neural networks for ECG biometric recognition. *Pattern Recognition Letters* 126 (2019), 78–85.

[32] Qiming Li, Muchuan Guo, and Ee-Chien Chang. 2008. Fuzzy extractors for asymmetric biometric representations. In *2008 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE, 1–6.

[33] Zhonghao Liao, George T Amariucai, Raymond KW Wong, and Yong Guan. 2017. The impact of discharge inversion effect on learning sram power-up statistics. In *2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 31–36.

[34] Meng-Hui Lim and Pong C Yuen. 2015. Entropy measurement for biometric verification systems. *IEEE transactions on cybernetics* 46, 5 (2015), 1065–1077.

[35] Shuiyin Liu, Yi Hong, Emanuele Viterbo, Alessia Marelli, and Rino Micheloni. 2019. Efficient decoding of low density lattice codes. *IEEE Wireless Communications Letters* 8, 4 (2019), 1195–1199.

[36] Zhenyu Liu and Dimitrios A Pados. 2005. A decoding algorithm for finite-geometry LDPC codes. *IEEE Transactions on Communications* 53, 3 (2005), 415–421.

[37] Vu Mai, Ibrahim Khalil, and Christopher Meli. 2011. ECG biometric using multilayer perceptron and radial basis function neural networks. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2745–2748.

[38] Dario Maio, Davide Maltoni, Raffaele Cappelli, James L. Wayman, and Anil K. Jain. 2002. FVC2000: Fingerprint verification competition. *IEEE transactions on pattern analysis and machine intelligence* 24, 3 (2002), 402–412.

[39] R Álvarez Mariño, F Hernández Álvarez, and L Hernández Encinas. 2012. A crypto-biometric scheme based on iris-templates with fuzzy extractors. *Information Sciences* 195 (2012), 91–102.

[40] David Menotti, Giovani Chiachia, Allan Pinto, William Robson Schwartz, Helio Pedrini, Alexandre Xavier Falcao, and Anderson Rocha. 2015. Deep representations for iris, face, and fingerprint spoofing detection. *IEEE Transactions on Information Forensics and Security* 10, 4 (2015), 864–879.

[41] Praveen Kumar Nayak and Devesh Narayan. 2012. Multimodal biometric face and fingerprint recognition using neural network. *International Journal of Engineering* 1, 10 (2012).

[42] Shahrin Azuan Nazeer, Nazaruddin Omar, and Marzuki Khalid. 2007. Face recognition system using artificial neural networks approach. In *2007 International Conference on Signal Processing, Communications and Networking*. IEEE, 420–425.

[43] Rodrigo Frassetto Nogueira, Roberto de Alencar Lotufo, and Rubens Campos Machado. 2016. Fingerprint liveness detection using convolutional neural networks. *IEEE transactions on information forensics and security* 11, 6 (2016), 1206–1213.

[44] Adam Page, Amey Kulkarni, and Tinoosh Mohsenin. 2015. Utilizing deep neural nets for an embedded ECG-based biometric authentication system. In *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 1–4.

[45] Gregory Poltyrev. 1994. On coding without restrictions for the AWGN channel. *IEEE Transactions on Information Theory* 40, 2 (1994), 409–417.

[46] Ronald Salloum and C-C Jay Kuo. 2017. ECG-based biometrics using recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2062–2066.

[47] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. *CoRR* abs/1503.03832 (2015). arXiv:1503.03832 http://arxiv.org/abs/1503.03832

[48] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[49] Naftali Sommer, Meir Feder, and Ofir Shalvi. 2008. Low-density lattice codes. *IEEE Transactions on Information Theory* 54, 4 (2008), 1561–1585.

[50] Branka Stojanović, Aleksandar Nešković, and Oge Marques. 2017. A novel neural network based approach to latent overlapped fingerprints separation. *Multimedia Tools and Applications* 76, 10 (2017), 12775–12799.

[51] Yagiz Sutcu, Qiming Li, and Nasir Memon. 2009. Design and analysis of fuzzy extractors for faces. In *Optics and Photonics in Global Homeland Security V and Biometric Technology for Human Identification VI*, Vol. 7306. International Society for Optics and Photonics, 73061X.

[52] Yagiz Sutcu, Elham Tabassi, Husrev T Sencar, and Nasir Memon. 2013. What is biometric information and how to measure it?. In *2013 IEEE international conference on technologies for homeland security (HST)*. IEEE, 67–72.

[53] Yao Tang, Fei Gao, Jufu Feng, and Yuhang Liu. 2017. Fingernet: An unified deep network for fingerprint minutiae extraction. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 108–116.

[54] Josh Taylor. 2019. Major breach found in biometrics system used by banks, UK police and defence firms. *The Guardian* (August 2019).

[55] Andrew BJ Teoh, Yip Wai Kuan, and Sangyoun Lee. 2008. Cancellable biometrics and annotations on biohash. *Pattern recognition* 41, 6 (2008), 2034–2044.

[56] FVC2000: the First International Competition for Fingerprint Verification Algorithms. 2000. . http://bias.csr.unibo.it/fvc2000/. [Online; accessed 10-February-2022].

[57] FVC2006: the Fourth International Fingerprint Verification Competition. 2006. . http://bias.csr.unibo.it/fvc2006/. [Online; accessed 10-February-2022].

[58] Valérie Viet Triem Tong, Hervé Sibert, Jérémy Lecoeur, and Marc Girault. 2007. Biometric fuzzy extractors made practical: a proposal based on fingercodes. In *International conference on Biometrics*. Springer, 604–613.

[59] Quang Nhat Tran, Benjamin P Turnbull, and Jiankun Hu. 2021. Biometrics and privacy-preservation: How do they evolve? *IEEE Open Journal of the Computer Society* 2 (2021), 179–191.

[60] The Hong Kong Polytechnic University. 2017. Contactless 2D to Contact-based 2D Fingerprint Images Database Version 1.0. http://www4.comp.polyu.edu.hk/~csajaykr/fingerprint.htm. [Online; accessed 10-February-2022].

[61] Xuebo Wang and Wai Ho Mow. 2019. Efficient LDLC decoder design from the lattice viewpoint. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.

[62] Fadhli Wong Mohd Hasan Wong, Ainil Sufreena Mohd Supian, Ahmad F Ismail, Lai Weng Kin, and Ong Cheng Soon. 2001. Enhanced user authentication through typing biometrics with artificial neural networks and k-nearest neighbor

algorithm. In *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat. No. 01CH37256)*, Vol. 2. IEEE, 911–915.

[63] Kai Xi, Jiankun Hu, and Fengling Han. 2011. An alignment free fingerprint fuzzy extractor using near-equivalent dual layer structure check (NeDLSC) algorithm. In *2011 6th IEEE Conference on Industrial Electronics and Applications*. IEEE, 1040–1045.

[64] Wencheng Yang, Jiankun Hu, and Song Wang. 2012. A delaunay triangle-based fuzzy extractor for fingerprint authentication. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 66–70.

[65] Metodi P Yankov, Martin A Olsen, Mikkel B Stegmann, Søren Sk Christensen, and Søren Forchhammer. 2019. Fingerprint entropy and identification capacity estimation based on pixel-level generative modelling. *IEEE Transactions on Information Forensics and Security* 15 (2019), 56–65.

[66] Yair Yona and Meir Feder. 2009. Efficient parametric decoder of low density lattice codes. In *2009 IEEE International Symposium on Information Theory*. IEEE, 744–748.

[67] Shereen Yong, Weng Kin Lai, and George Goghill. 2004. Weightless neural networks for typing biometrics authentication. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 284–293.