FingerFaker: Spoofing Attack on COTS Fingerprint Recognition Without Victim's Knowledge

Yijie Shen[†]
Zhejiang University
ZJU-Hangzhou Global Scientific and Technological Innovation Center
Hangzhou, China
shenyijie@zju.edu.cn

Zhe Ma[†]
Zhejiang University
ZJU-Hangzhou Global Scientific and Technological Innovation Center
Hangzhou, China
mazhe1013@zju.edu.cn

Feng Lin[*]
Zhejiang University
ZJU-Hangzhou Global Scientific and Technological Innovation Center
Hangzhou, China
flin@zju.edu.cn

Hao Yan
Zhejiang University
Hangzhou, China
yanhao0366@zju.edu.cn

Zhongjie Ba
Zhejiang University
Hangzhou, China
zhongjieba@zju.edu.cn

Li Lu
Zhejiang University
Hangzhou, China
li.lu@zju.edu.cn

Wenyao Xu
University at Buffalo
Buffalo, New York, USA
wenyaoxu@buffalo.edu

Kui Ren
Zhejiang University
Hangzhou, China
kuiren@zju.edu.cn

ABSTRACT

Fingerprint recognition has been a vital security guard for various applications whose vulnerability has been explored by different works. However, previous works on spoofing fingerprint recognition rely on prior knowledge (e.g., photos and minutiae) of the target fingerprint, which fails to implement in practical scenarios. In this paper, we design a fingerprint spoofing attack, namely *FingerFaker*, to explore the vulnerability of fingerprint recognition, which can spoof automated fingerprint recognition systems (AFRSs) without prior knowledge of target fingerprints. Specifically, we propose a novel concept of "pseudo-minutiae-set" as an effective optimization object and design a two-stage scheme to optimize "pseudo-minutiae-set" leveraging a two-factor evolutionary strategy. In addition, we use a GAN-based training strategy with a minutiae loss function to pre-train a fingerprint generator to map a "pseudo-minutiae-set" into a fingerprint. We use 6342 fingerprint images to verify the performance of *FingerFaker* on spoofing the open-source AFRS, which shows a high attack success rate (ASR) of 97.78%. Meanwhile, we conduct a realistic case study on commercial off-the-shelf (COTS) AFRS, where *FingerFaker* also shows 94.22% ASR. Finally, we explore the impact of different conditions to guide the attack and propose countermeasures to mitigate the harm.

CCS CONCEPTS

• **Security and privacy** → *Biometrics.*

KEYWORDS

Fingerprint recognition, spoofing attack, no prior-knowledge

ACM Reference Format:
Yijie Shen, Zhe Ma, Feng Lin, Hao Yan, Zhongjie Ba, Li Lu, Wenyao Xu, and Kui Ren. 2023. FingerFaker: Spoofing Attack on COTS Fingerprint Recognition Without Victim's Knowledge. In *ACM Conference on Embedded Networked Sensor Systems (SenSys '23), November 12–17, 2023, Istanbul, Turkiye.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3625687.3625783

1 INTRODUCTION

Automated fingerprint recognition systems (AFRS) have gained popularity in various applications such as mobile payments and access control, as they provide a secure means of user verification using their unique fingerprints. The security of fingerprinting is of utmost importance due to the use of commercial off-the-shelf (COTS) AFRS in safeguarding critical applications.

Previous works have shown the latent risk of AFRSs. One common method of attack involves stealing fingerprints to bypass AFRS security measures, which has been reported as successful in Apple's TouchID [61] sensors and Samsung S10 smartphones [38]. However, this type of attack has limitations since adversaries must physically follow victims and obtain traces of their fingerprints [40]. Additionally, fingerprints are more subtle biometrics than other types such as face and voice, making stealing them a high-cost and inefficient process. Another category of attack involves reconstructing fingerprints from minutiae (See Section 3) that can be obtained through invading fingerprint template databases. Nevertheless, these databases are protected by multiple safeguards, including malicious intrusion detection [4, 18, 50] and encryption circuits [7, 28], which limit such attacks.

[†]These authors contributed equally to this work.
[*]Feng Lin is the corresponding author.

SenSys '23, November 12–17, 2023, Istanbul, Turkiye

ACM ISBN 979-8-4007-0414-7/23/11…$15.00
https://doi.org/10.1145/3625687.3625783

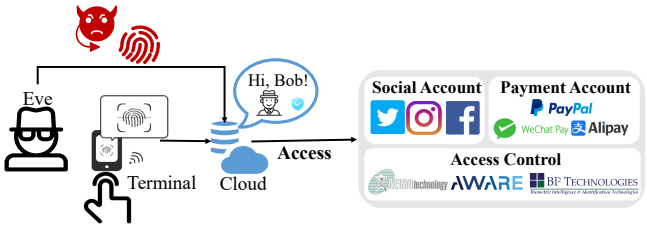


Figure 1: Eve utilizes the fake fingerprint to bypass fingerprint recognition on the cloud to execute high-risk operations with Bob's identity.

In this study, we aim to examine the security of fingerprinting and explore potential vulnerabilities through practical experimentation. Specifically, we pose the question: **How can synthetic fingerprints be generated without access to the target's fingerprint/minutiae?** Adversarial attacks using techniques such as adversarial examples [14] and generative adversarial networks (GANs) [54] have successfully generated synthetic biometrics for spoofing other biometric systems such as facial [63, 64] and vocal recognition [74]. As such, adversarial attacks hold promise in generating synthetic fingerprints without access to the target's fingerprint/minutiae. However, prior attempts [10] to use GANs for fake fingerprint generation failed to effectively constrain the generated fingerprint to a specific target, thereby rendering the attack unsuccessful in spoofing AFRSs.

Inspired by previous works, we attempt to generate fake fingerprints under a gray-box setting. However, we encounter three core challenges.Firstly, we must address the challenge of querying the AFRS using a feature set. Prior research [59] optimized the feature set by querying the template of the target fingerprint using evolutionary algorithms. However, this approach is impractical as the AFRS accepts a fingerprint as input, not a feature set. To overcome this, we propose a two-stage scheme that includes a feature-to-fingerprint mapping stage and a feature optimization stage. The former stage maps the feature set into a fingerprint to spoof the AFRS, while the latter optimizes the feature set by querying the target AFRS with the generated fingerprint. Secondly, we need to design an appropriate form of the optimization object (i.e., the feature set). Our experiments showed that using a random vector as the optimization object in the feature optimization stage [10] is not effective for generating fake fingerprints. Hence, we need a well-designed optimization object. Thirdly, it's essential to recognize that our feature set is inherently incomplete when compared to the feature set of the target fingerprint. This is primarily due to the fact that critical information, such as the quantity of minutiae, remains unknown during our attack. As demonstrated in Section 8.7, the state-of-the-art generator that synthesizes a fingerprint using whole minutiae performs poorly in our spoofing attack. Therefore, we need an advanced generator to achieve the mapping stage.

We have designed an end-to-end spoofing attack called *FingerFaker* that utilizes our two-stage scheme to create high-resolution fake fingerprints and spoof AFRSs. Our approach involves two main components. First, we introduce a novel concept of a "pseudo-minutiae-set" to replace random vectors in the optimization object for generating effective fake fingerprints. To optimize it, we use a two-factor evolutionary strategy based on non-dominated sorting [22] and genetic operators during the feature optimization stage. Second, we develop a fingerprint generator capable of mapping the "pseudo-minutiae-set" to the fingerprint and train it using a GAN-based strategy with a minutiae loss for pre-training. We summarize our contributions as follows:

- We conduct the investigation of spoofing COTS AFRSs without the victims' fingerprints and propose an end-to-end spoofing attack, namely *FingerFaker*. Specifically, we propose a two-stage scheme to generate fake fingerprints under the gray-box setting without any victims' information.
- We propose the concept of "pseudo-minutiae-set" as the optimization object in the two-stage scheme. Based on such an object, we design a two-factor evolving strategy and a GAN-based fingerprint generator with a well-designed loss function to work together to generate fake fingerprints.
- We conduct comprehensive experiments. The attack success rate (ASR) achieves 97.78%, which is competitive compared with the state-of-the-art works in generating fake fingerprints with victims' prior knowledge.

2 THREAT MODEL

We consider a scenario where an adversary, hereafter Eve, attempts to bypass AFRSs in the cloud to execute a high-risk operation (e.g., payment) with the identity of the victim, hereafter Bob. Eve utilizes *FingerFaker* to generate a high-resolution fake fingerprint of Bob. After that, Eve launches fingerprint recognition in her device and sends the fake fingerprint to the AFRS through its API to break the security guard and execute the operation. In the past, the reliance on Bob's fingerprint/minutiae is the main issue of a fingerprint spoofing attack. To overcome the obstacle, Eve uses the novel concept of "pseudo-minutiae-set" in *FingerFaker*. To clarify the capacity of Eve, we consider the following assumptions:

Gray-box setting: Eve's focus lies on the gray-box setting where she can only access the authentication results, namely the decisions and match scores. In this setting, the AFRS decides by comparing the match score against an empirical threshold. This approach is reasonable since various suppliers (e.g., VeriFinger [52], SecuGen [62], CloudABIS [19], Innovatrics [35], Bio-Key [5], and Aware [8]) of AFRS specify that their application programming interface (API) will return match scores. Additionally, it is estimated that the AFRS products distributed by these vendors have been deployed across over 170 countries and regions, serving a vast user base exceeding 100 million individuals [9, 20, 53]. In addition, we refrain from making any assumptions regarding the internal implementation of the AFRS system, whether it relies on traditional minutiae matching(e.g., Bozorth3 [55] for open-source AFRS) or deep learning(e.g., VeriFinger [52] for COTS AFRS)).

Ability to query: We assume that Eve can query the AFRS. As a countermeasure against security threats, AFRSs implement a tolerance value between two successful authentications. Specifically, to ensure a seamless user experience, the allowable number of attempts typically ranges from 5 to 50, depending on the manufacturer and application [32]. Furthermore, successful user authentication resets the count of failed attempts. Based on this setup, Eve can

implement attacks by spreading queries to different days. Additionally, the research [16] indicates that even in mature systems like Android, Eve can exploit system vulnerabilities to perform unrestricted queries. In our study, we concentrate on devising a method to generate high-resolution fake fingerprints while assuming that Eve has query capacity according to the aforesaid solutions. Exploring ways to further enhance the efficiency of querying AFRS is intriguing. In this paper, we only provide a brief statement regarding its feasibility without delving into further research.

No high-precision fingerprint/minutiae: Eve cannot obtain high-precision fingerprint/minutiae from Bob, as she has no prior knowledge of him. Bob's vigilance prevents Eve from accessing his fingerprint/minutiae directly, and he takes precautions to remove traces of his fingerprints from touched surfaces to prevent leakage. Also, fingerprints are a more subtle biometric modality than others, unlike face and voice, making them harder for Eve to access by taking photos.

Non-invasiveness: Eve is unable to breach the software or hardware of AFRSs to modify or steal Bob's fingerprint due to the implementation of multiple security measures ranging from software to hardware [4, 28].

3 FINGERPRINTING PRELIMINARIES

There are various approaches to implementing fingerprint recognition (i.e., AFRSs), which can be mainly divided into two types: (1) feature-matching approaches [36] and (2) image-matching approaches (by deep learning) [72]. Due to its lightweight and effectiveness, the former type is more widely used in real life. Therefore, we focus on feature-matching approaches to explore the end-to-end fingerprinting spoofing attack, which can achieve an attack with a practical impact.

3.1 Minutiae description

A fingerprint is regarded as the combination of ridges and valley patterns that can be divided into bifurcation and ending. Conventional approaches [36] consider a two-level model, as shown in Figure 2, which consists of level-1 (i.e., orientation map, ridge frequency map, and enhanced map) and level-2 (i.e., minutiae) features. Meanwhile, **minutiae** are the most common features used to describe fingerprints in COTS AFRSs, and other features are usually used to assist the AFRS in extracting minutiae. In practice, there are various methods (e.g., MINDTCT [55] and VeriFinger [52]) that can extract minutiae from fingerprints, and they, in general, involve two main modules: image enhancement and feature extraction, where the image enhancement module improves the clarity of the pattern of the fingerprint for stable feature extraction by different steps (e.g., orientation calculation, ridge frequency calculation, Gabor filtering), and the feature extraction module processes the enhanced image to obtain features (i.e., minutiae) to represent the fingerprint for matching [33].

3.2 Fingerprint matcher

The fingerprint matcher determines the effectiveness, robustness, and reliability of the recognition result, which is a significant distinction between AFRSs. To verify the performance of *FingerFaker*

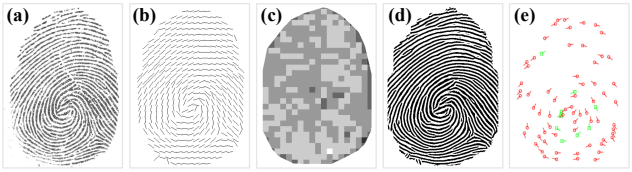


Figure 2: Fingerprint features: (a) orignal; (b) orientation; (c) ridge frequency; (d) enhanced; (e) minutiae.

in both research and commercial scenarios, we utilize an open-source matcher and a COTS matcher in this paper. We first introduce an open-source fingerprint matcher, namely Bozorth3, from the national institute of standards and technology (NIST) due to its widespread use in previous works [44, 48]. It compares two fingerprints in three main steps [68]: (1) intra-fingerprint comparison, (2) inter-fingerprint comparison, and (3) traverse. First, the matcher calculates the distance and angular difference between each pair of two minutiae in the same fingerprint and records the result in a table. Second, the matcher looks for compatible entries whose difference is within the threshold between the two tables (i.e., tables of the registered fingerprint and tested one). Finally, the matcher traverses and clusters compatible table entries into clusters to calculate the match score. A higher match score means a higher similarity between two fingerprints. In addition, we introduce a COTS fingerprint matcher, namely VeriFinger, due to its superior performance compared with others [52]. Based on our results, *FingerFaker* can effectively spoof the AFRS achieved by the open-source or the COTS fingerprint matcher, which shows the generalization of *FingerFaker*.

4 PSEUDO-MINUTIAE-SET

In this section, we defined a novel concept, i.e., pseudo-minutiae-set, which is formally consistent with the minutiae but only has partially matched features of the target fingerprint, which gets by optimization. Based on Section 8.7, it fails to optimize a random vector to generate effective fake fingerprints to spoof AFRSs. Inspired by previous works on the reconstruction of fingerprints, we attempt to optimize an object consisting of minutiae to approach the minutiae of the target fingerprint to generate effective fake fingerprints. However, it is impossible to totally recover the minutiae of the target fingerprint due to the large candidate space [76] caused by multidimensional values (e.g., the number of minutiae, the x-location, the y-location, and the orientation). In particular, the number of minutiae dramatically increases the size of the candidate space, where the candidate space will be searchable if we can limit the number of minutiae [15]. To this end, we proposed the concept of "pseudo-minutiae-set" to overcome the problem by the following characteristics of fingerprint recognition: it is a fact that the different scanning of the same fingerprint will have different minutiae, which forces the AFRS to have a tolerance for the testing fingerprint, i.e. when partial minutiae are consistent, the conclusion can be drawn that the testing and registered fingerprints are the same ones. Hence, it is possible to find a proper fixed size to construct a "pseudo-minutiae-set" to replace real minutiae, which can satisfy the requirement of generating effective fingerprints. Based

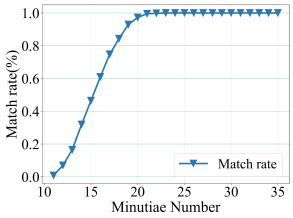


Figure 3: Match rate under different number of reversed minutiae.

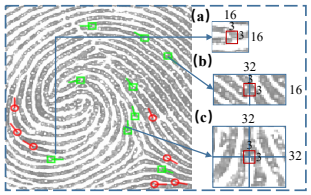


Figure 4: Related area of a minutia when it is in different locations.

on this insight, we conduct a verification experiment to prove the feasibility of finding a fixed size for the "pseudo-minutiae-set".

Setup: We randomly select 5800 fingerprints from our dataset(as shown in Table 1) and use MINDTCT to extract minutiae from such fingerprints. For each fingerprint, we randomly reserve n minutiae and match the reserved minutiae with the original one by Bozorth3 with a threshold corresponding to a 0.01% false match rate (details in Section 8), where we define the match rate as the following:

$$Match\ rate = \frac{SMA}{TA}, \tag{1}$$

where SMA is the number of successful attempts, and TA is the number of total attempts.

We set the number of reserved minutiae n as 11 and increase the number with the step of one until it reaches 35.

Insights: As shown in Figure 3, the match rate increases with the number of reserved minutiae and can achieve 100% when the number achieves 24. It indicates that AFRSs have a tolerance of loss for minutiae, and the lower limit of the number is 24 in our experiment. Based on this result, we design "pseudo-minutiae-set" that includes ***30 quadruple tuples in the form of minutia (i.e., x-y location, orientation, and type)*** to optimize during the attack. In fact, the number is a hyperparameter, which we set as 30 to offer a certain level of redundancy (See Section 10 for details).

5 MAPPING STRATEGY: DESIGN OF LOSS FUNCTION

Previous works [11–13, 44]focus on mapping whole minutiae into fingerprints. However, we can only acquire a "pseudo-minutiae-set" in the attacking, which leads to poor performance of conventional approaches (See Section 8.7). Thus, we will review the process of minutiae calculation from a fingerprint image and propose a mapping strategy to generate fake fingerprints leveraging a special minutiae loss.

5.1 Minutiae Calculation

A minutia is a quadruple tuple that consists of x-y locations, orientation, and type.

Calculating orientation: To obtain the orientation of the minutiae, AFRS needs to calculate the overall orientation field. First, a 3×3 Sobel operator [37] is constructed to calculate the gradient $G_x(x, y)$ and $G_y(x, y)$ in x and y directions of all points. Second, the fingerprint image will be partitioned into 16×16 blocks and calculate the ***second-order gradients*** of the centroid of the block by the following formulations:

$$G_{xy} = \sum_{u=1}^{16}\sum_{v=1}^{16} 2G_x(u,v)G_y(u,v), \tag{2}$$

$$G_{xx} = \sum_{u=1}^{16}\sum_{v=1}^{16} G_x(u,v)^2, \tag{3}$$

$$G_{yy} = \sum_{u=1}^{16}\sum_{v=1}^{16} G_y(u,v)^2, \tag{4}$$

where G_{xy}, G_{xx}, and G_{yy} are the second-order gradients of the centroid of the block in different directions. Finally, we can represent the ***orientation*** of the block θ with the following:

$$\theta = \frac{1}{2}tan^{-1}(\frac{2G_{xy}}{G_{xx} - G_{yy}}) + \frac{\pi}{2}. \tag{5}$$

The orientation of the minutia is equal to the orientation of the block that contains them.

Calculating x-y location and type: To obtain reliable minutiae, AFRSs need to remove noise in the fingerprint image. Hence, the Gabor filter is employed to ***denoise*** each 16×16 block in previous works by the following formulation [33]:

$$\begin{aligned} h(x,y,\theta,f) = exp(-\frac{1}{2}[\frac{(xcos\theta + ysin\theta)^2}{\delta_x^2} \\ + \frac{(-xsin\theta + ycos\theta)^2}{\delta_y^2}] \times exp[2\pi jf(xcos\theta + ysin\theta)], \end{aligned} \tag{6}$$

where the θ is the orientation of the block, f is the frequency of the block, i.e., the reciprocal of the average distance between adjacent extreme points of the block, δ_x^2 and δ_y^2 are empirical constants. After all 16×16 blocks are processed by Gabor filters, the enhanced fingerprint can be used to ***extract minutiae***. Each point of the enhanced fingerprint can be partitioned into a 3×3 matrix. The centroid of the matrix is the testing point with the location of (x, y) that is marked as A, and the surroundings are marked A_1 to A_8. We can formulate the sum of the differences of adjacent point-pairs in such neighborhoods as $cn(A)$ with the following formulation [6]:

$$cn(A) = \sum_{k=1}^{8} |I_{A_{(k+1)\ mod\ 8}} - I_{A_k}|, \tag{7}$$

where I_k is the intensity, which is 0 or 1, on the enhanced fingerprint image. When $cn(A) = 2\ or\ 6$, the point can be regarded as a minutia. In addition, when $cn(A) = 2$, the type of the minutia is ending, otherwise it is bifurcation.

5.2 Transformation between Minutiae and Fingerprint Image

A minutia is only related to a limited area (a 3×3 block) around itself and 16×16 blocks where points of the 3×3 block are located. Specifically, it, as shown in Figure 4 (a), is associated with a single 16×16 block when it is not positioned on the border of the block, and it is associated with two and four blocks when it is on the border and the corner of the block, as shown in Figure 4 (b) and (c), respectively. Thus, we attempt to constrain the 32×32 block (it is in) to design the loss function. Such a block's left-top (b_x, b_y) can be calculated by the followings:

$$b_x = (m_x - 1) - (m_x - 1)\ mod\ 16, \tag{8}$$

$$b_y = (m_y - 1) - (m_y - 1)\ mod\ 16, \tag{9}$$

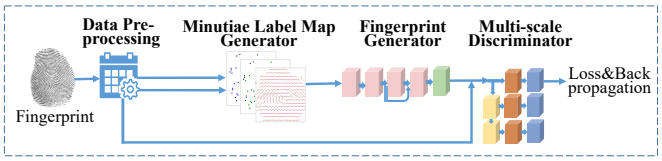


Figure 5: FingerFaker generator pre-training.

where the m_x and m_y are the x and y locations of the minutiae. The formulations above indicate that the optimization involves not only the original values of the block points, but also their first and second-order gradients. This insight prompts us to consider the three-level information of such blocks. Consequently, we design a loss function based on this idea to constrain the "pseudo-minutiae-set" blocks. We provide a detailed description of the implementation of this approach in Section 7.

6 ATTACK OVERVIEW

As shown in Figure 5 and Figure 6, *FingerFaker* focuses on spoofing COTS AFRSs, which can be divided into two phases: (1) fingerprint generator pre-training and (2) targeted attack. With the two phases, *FingerFaker* can generate high-resolution fake fingerprints to bypass the COTS AFRS.

Fingerprint generator pre-training phase: As shown in Figure 5, it includes four core modules: (1) data pre-processing, (2) minutiae label map generator, (3) fingerprint generator, and (4) fingerprint discriminator. First, *FingerFaker* pre-processes training data by a series of operations (i.e., screen and segment). Second, it employs MINDTCT to obtain the minutiae and orientation of pre-processed fingerprints and use the minutiae label map generator to represent them in a label map. Finally, GAN-based training is implemented on the fingerprint generator with the discriminator and well-designed loss functions to provide the generator.

Targeted attack phase: In this phase, the adversary attacks the targeted victim in the COTS AFRS, leveraging *FingerFaker* with the pre-trained fingerprint generator, as shown in Figure 6. It can be further divided into two stages: (1) feature-to-fingerprint mapping and (2) feature optimizing. The former stage generates fake fingerprints and determines whether to optimize the features. Based on non-dominated sorting and genetic operators, the latter stage uses a two-factor evolutionary strategy to promote the iteration of a "pseudo-minutiae-set" for generating a satisfactory fake fingerprint.

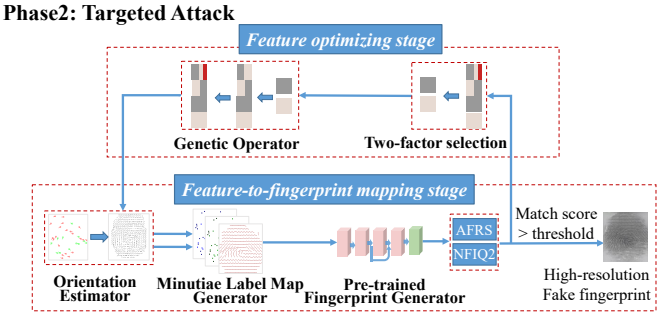


Figure 6: Targeted attack leveraging two-stage scheme.

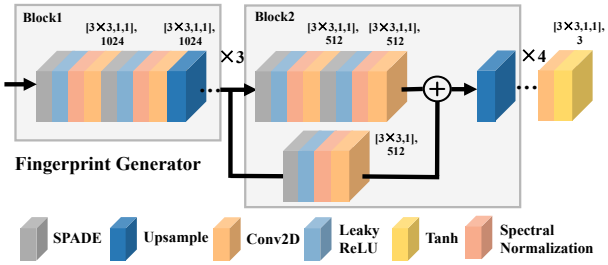


Figure 7: GAN-based fingerprint generator.

First, the adversary randomly initializes a population of "pseudo-minutiae-set" and sends it to the feature-to-fingerprint mapping stage after processing by the genetic operators. Second, an orientation estimator is employed to reconstruct the orientation of the "pseudo-minutiae-set" in the population. Third, the minutiae label map generator and pre-trained fingerprint generator work together to generate fake fingerprints. Next, *FingerFaker* queries the AFRS by the fake fingerprint and fetches its result and match score. If the result is a failure, *FingerFaker* sends the match score, the quality (calculated by a local quality evaluator, i.e., NIST finger image quality), and the current population to the two-factor selection module and repeats the above steps until the result is a success or the number of iterations exceeds the upper limit. Finally, *FingerFaker* outputs the fake fingerprint of the highest score in the population.

7 FINGERFAKER: ATTACK DESIGN

7.1 Fingerprint Generator Pre-Training

Data pre-processing: It is a fact that there is noise during fingerprint scanning. *FingerFaker* first uses a fingerprint quality evaluator, i.e., NIST fingerprint image quality (NFIQ) 2 [55], to remove fingerprint images whose quality is below 30. Second, *FingerFaker* crops high-quality fingerprint images leveraging NIST fingerprint segmentation (NFSEG) [55] and resizes them to 256×256. Finally, *FingerFaker* describes the minutiae and orientation of these fingerprint images by an extractor (i.e., MINDTCT) due to its accessibility.

Minutiae label map generator: A three-channel label map, namely a minutiae label map, is used to map the real minutiae set/"pseudo-minutiae-set" into a proper form for training/calling the GAN-based fingerprint generator. Specifically, it uses three channels to label the location of minutiae, the orientation of minutiae, and the orientation of the fingerprint. In the fingerprint generator pre-training phase, we provide the minutiae of the real minutiae set and the orientation calculated by the real fingerprint to generate a minutiae label map. In the targeted attack phase, we provide minutiae of the "pseudo-minutiae-set" and estimate the orientation of the fingerprint by such minutiae. A minutiae label map generator is designed in *FingerFaker* to translate pre-processed minutiae and orientation into the minutiae label map. Specifically, given a minutia set of the fingerprint image x, let $M_x \in \mathbb{R}^{H \times W \times 3}$ be a minutiae label map, where H, W and 3 are the height, width and number of channels of the image. In order to obtain M_x, we define V_x to be calculated as:

$$V_x^{i,j} = \begin{cases} O_{ij}, \theta_{ij}, t_{ij} & p_{ij} \text{ is a minutia} \\ O_{ij}, 0, 0 & \text{otherwise} \end{cases}, \tag{10}$$

where (i, j) is the coordinate in the map, and p_{ij} is the point of the corresponding position. O_{ij} is the orientation of the block, where

points of a 16×16 block share the same orientation. $\theta_{ij} \in [0, 180°)$ and t_{ij} (i.e., 128 or 255) are the orientation and type of the minutia at (i, j). To obtain M_x, we denote O_{ij} in M_x^0 as a line, i.e., when $O_{ij} < 90°$, the generator draws a line between the lower left corner of the 16×16 block and the boundary point (e.g., 11.25°), otherwise it draws between the lower right corner and the boundary point. If the θ_{ij} and t_{ij} are not 0, the 3×3 blocks centred of M_x^{1ij}/ M_x^{2ij} are filled with θ_{ij} and t_{ij}, respectively.

Fingerprint generator: Inspired by the previous work [56], we design the fingerprint generator based on spatially-adaptive denormalization (SPADE) layers. As shown in Figure 7, the generator consists of three block 1, four block 2, and a three-channel output layer with a spectral normalization layer, where block 1 and 2 are mainly based on the module consisting of a SPADE layer, a LeakyReLU layer, a spectral normalization layer and a two-dimensional convolutional layer, hereafter ***base module***. Block 1 contains two base modules and an upsample layer, where the convolutional layer of each base module has 3×3 convolution kernels with the stride and the padding of 1 and 1024 output channels. Block 2 contains two base modules in the main skeleton and a base module in the short connection, and an upsample layer at the end, where 3×3 convolution kernels with the step of 1 are deployed in each base module. Meanwhile, the output channel of the first block 2 is 512, and the number of output channels of the latter block will become half of the former one.

Multi-scale discriminators: Multi-scale discriminators contain three discriminators with the same structure but operate at different scales [73]. Specifically, the fingerprints are downsampled by a factor of 2 and 4 by an average pooling layer of 3x3 kernel size with stride 2 to create images of 3 scales. Each discriminator contains one block 1 and three block 2. Block 1 consists of a spectral normalization layer [49], a two-dimensional convolutional layer, and a LeakyReLU layer, where the convolutional layer has 4×4 convention kernels with the stride and padding of 2 and 64 output channels. Block 2 compares block 1, adding an instance normalization layer [71] before the LeakyReLU layer, where the convolutional layer has 4×4 convention kernels, whose strides are 2, 1 and 1, with the padding of 2. The output channel of the first block 2 is 128, and the output channels of the latter block will become double the former one.

Minutiae adversarial loss: Based on the transformation in Section 5, *FingerFaker* first divides the fingerprint area into 32×32 blocks that include minutiae. For each block, *FingerFaker* measures

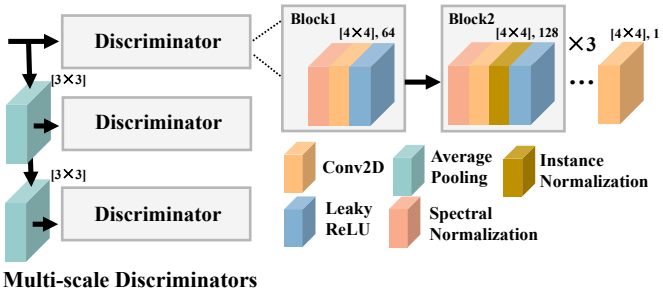


Figure 8: GAN-based fingerprint discriminator.

Algorithm 1 Targeted Attack

Input: Thresholds for AFRS t;
Output: The high-resolution fake fingerprint h;

Initialize $K \leftarrow 70$, $N \leftarrow 30$,$maxScore \leftarrow 0$
Initialize $curPopm \leftarrow RandomSampling(N, K)$
//*SBCrossover: Simulated Binary Crossover*
//*MLGenerator: Minutiae Label Generator*
while $maxScore <= t$ **do**
  //*Genetic operators*
  $offPopm \leftarrow SBCrossover(curPopm)$;
  $offPopm \leftarrow PolynomialMutation(offPopm)$;
  $popm \leftarrow MergePopm(curpopm, offPopm)$;
  for $i \leftarrow 1 : length(popm)$ **do**
    $o_i \leftarrow OrientationEstimator(popm_i)$;
    $l_i \leftarrow MLGenerator(popm_i, o_i)$;
    $f \leftarrow FingerprintGenerator(l_i)$;
    $maxScore \leftarrow Max(AFRS(f), maxScore)$;
    $fit \leftarrow AFRS(f), NFIQ2(f)$;
  end for
  $nextPopm \leftarrow TwoFactorSelection(popm, fit)$;
  $curPopm \leftarrow nextPopm$;
end while
return h

the difference between the real and generated fingerprint in zero-order gradients to ensure the geometry of the minutiae and in the first-order and second-order gradients to ensure the orientation. Specifically, given a minutiae label map s and its corresponding real fingerprint y, the minutiae adversarial loss is calculated as:

$$\mathcal{L}_{\text{MINUTIAE}}(G) = \sum_{i=1}^{N} \sum_{j=1}^{6} \left[\left\| F(y_j, i) - \left(F(G_j(s), i)\right) \right\|_2 \right], \quad (11)$$

where N is the number of minutiae, $F(\cdot, i)$ is a function that can extract the area of the block including the i^{th} minutia, and j is the mark that can represent the zero-order, the first-order (i.e., x-direction and y-direction) and the second-order (e.g., xx-direction, yy-direction, and xy-direction) gradients of the block of both real and generated fingerprints. In addition, *FingerFaker* employs another loss in previous works [56], and the final loss function is the following:

$$\mathcal{L}_{\text{OVERALL}} = \mathcal{L}_{\text{PIROR}} + \lambda \mathcal{L}_{\text{MINUTIAE}}(G). \quad (12)$$

7.2 Targeted Attack

This phase aims at generating a fake fingerprint to spoof the AFRS by the pre-trained fingerprint generator. The process is detailed in Algorithm 1. Specifically, we randomly sample an initial population, which consists of K "pseudo-minutiae-set" (here, $K = 70$). Then, we continuously iterate the population by the algorithm until occurring a "pseudo-minutiae-set" that can generate the fake fingerprint to spoof the AFRS. As discussed in Section 4, we fix the number of minutiae in the "pseudo-minutiae-set" as N (here, $N = 30$).

7.2.1 Feature-to-fingerprint mapping stage. In this stage, the population of "pseudo-minutiae-set" is mapped to the corresponding fingerprint images.

For each "pseudo-minutiae-set", *FingerFaker* first estimates the orientation from it by the orientation estimator and generates the minutiae label map by it and the estimated orientation. Second, the pre-trained fingerprint generator is used to generate fake fingerprints leveraging the minutiae label map. Next, *FingerFaker* fetches the match score and quality of the fake fingerprint by querying AFRS in the cloud and calling NFIQ2 in local, respectively. Finally, *FingerFaker* compares the match score with the threshold and outputs the fake fingerprint of the highest match score in the current population if the match score exceeds the threshold. Otherwise, after all "pseudo-minutiae-set" in the population are processed, the match scores and the qualities are sent to the feature optimizing stage to optimize the population of "pseudo-minutiae-set".

Orientation estimator: *FingerFaker* employs an approach in previous work [26] to estimate the orientation of the "pseudo-minutiae-set". Specifically, for each block, *FingerFaker* partitions the fingerprint into eight sectors centered on the block, i.e., one sector every 45°, and find the nearest minutiae in each sector, respectively. The orientation of the block is calculated by the following formulation:

$$O(m,n) = \frac{1}{2}\arctan\left(\frac{\sum_{j=1}^{J}\sin\left(\alpha_j\right)w_j}{\sum_{j=1}^{J}\cos\left(\alpha_j\right)w_j}\right), \tag{13}$$

where J is the number of minutiae ($J = 8$ here), (m, n) is the location of the block, w_j is the reciprocal of the Euclidean distance between the center of the block and the j^{th} minutia, and α_j is the orientation of the j^{th} minutia. In algorithm 1, o is the complete orientation consisting of blocks.

7.2.2 Feature optimizing stage. In this stage, *FingerFaker* optimizes the population of "pseudo-minutiae-set" by genetic operators and the two-factor selection.

Two-factor selection: It is based on the two-factor evolutionary strategy that takes the match scores and qualities of the fingerprints generated by the population of "pseudo-minutiae-set" as the fitness values to screen the population. For two "pseudo-minutiae-set" x_1 and x_2, x_1 dominates x_2, which means x_1 has the higher competitiveness, if all fitnesses of x_1 are greater than or equal to the fitness of x_2 and at least one fitness of x_1 is strictly greater than the fitness of x_2. For "pseudo-minutiae-set" that cannot be sorted by the above approach (i.e., they have no dominance relation), *FingerFaker* first sorts them as f by a single type of fitness and generates a lookup table T, whose i^{th} item records the sequence numbers of the i^{th} "pseudo-minutiae-set" in the f. The crowding distance of i^{th} "pseudo-minutiae-set" is calculated to sort them with the following:

$$d_i = \sum_{m=1}^{M}\frac{f_m(T_m(i)-1) - f_m(T_m(i)+1))}{Max(f_m) - Min(f_m)}, \tag{14}$$

where f_m is the m^{th} fitness (e.g., match score) and T_m is the m^{th} lookup table, and M is the number of fitness types ($M = 2$ here). If the i^{th} "pseudo-minutiae-set" has the highest or lowest fitness in both types, d_i is set to infinity. Then, we sort these "pseudo-minutiae-set" with no dominance relation by the crowding distances.Finally, K ($K = 70$ here) "pseudo-minutiae-set" will be selected through binary tournament [42] and elite strategy [22] for the next generation.

Genetic operators: The genetic operators contain a simulated binary crossover (SBX) [21] operator and a polynomial mutation [21] operator: (1) ***simulated binary crossover operator***: the operator generates the offspring population of "pseudo-minutiae-set" by swapping attributes of two "pseudo-minutiae-set" $x^{(I,cur)}$ and $x^{(j,cur)}$ in current population with a probability of 0.9 by the following formulation:

$$\begin{cases} x_k^{(i,off)} = 0.5\left[(1+\beta)\cdot x_k^{(i,cur)} + (1-\beta)\cdot x_k^{(j,cur)}\right] \\ x_k^{(j,off)} = 0.5\left[(1-\beta)\cdot x_k^{(i,cur)} + (1+\beta)\cdot x_k^{(j,cur)}\right] \end{cases}, \tag{15}$$

where i and j are marks of different "pseudo-minutiae-set" in both the current and offspring population, k is the mark of the k^{th} attribution (e.g., x location and orientation) in a "pseudo-minutiae-set", β is calculated by the following:

$$\beta = \begin{cases} (2u)^{\frac{1}{(1+\eta)}} & \mathrm{u} \le \frac{1}{2} \\ \left(\frac{1}{2-2u}\right)^{\frac{1}{(1+\eta)}} & \text{otherwise} \end{cases}, \tag{16}$$

where $u \in [0,1]$ is a random number, and η is a customized parameter (here $\eta = 28$). (2) ***polynomial mutation***: after the crossover, the offspring population will be mutated to promote evolution. For the "pseudo-minutiae-set" of the offspring population $x^{(i,off)}$, the each attribution of the "pseudo-minutiae-set" will be mutated with the probability of 0.08 when k^{th} attribution is selected, the mutated x_k^{off} is calculated as:

$$x_k^{off} = x_k^{off} + \delta\cdot(u_k - l_k), \tag{17}$$

where l_k and u_k are the lower and upper limitation of the attribution, δ can be defined as:

$$\delta = \begin{cases} 1 - \left[2(1-u) + 2(u-\frac{1}{2})\left(1-\delta_2\right)^{\eta+1}\right]^{\frac{1}{\eta+1}} & \mathrm{u} > \frac{1}{2} \\ \left[2u + (1-2u)\left(1-\delta_1\right)^{\eta+1}\right]^{\frac{1}{\eta+1}} - 1 & \text{otherwise} \end{cases}, \tag{18}$$

where $\delta_1 = \frac{x_k^t - l_k}{u_k - l_k}$ and $\delta_2 = \frac{u_k - x_k^t}{u_k - l_k}$, $u \in [0,1]$ is a random number, and η is a customized parameter (here $\eta = 20$). We round attributions of the offspring population.

8 EVALUATION

8.1 Implementation details

We deploy our *FingerFaker* on the server with Ubuntu 20.04, Intel Xeon CPU E5-2678 v3@ 2.50GHz with 125G RAM, and four NVIDIA GeForce GTX 3090 GPUs. The initial learning rates of the generator and discriminant are set to 0.0001 and 0.0004. We conduct the GAN-based training for 140 epochs, where we keep the learning rate constant in the first 80 epochs and decay the learning rate linearly to 0 for in latter 60 epochs. We use the ADAM optimizer [39] with β_1=0 and β_2 = 0.999. For the minutiae adversarial loss function, we set the weights λ=50 in Eq. 12.

8.2 Dataset

We use the LivDet [29] datasets to evaluate the performance of *FingerFaker*. LivDet is widely used as a dataset for presentation attack detection, and we use real fingerprints from it for training and testing. Data from the different sensors (i.e., CrossMatch, Biometrika, and GreenBit) are used here. We also use an optical fingerprint sensor named ZKTeco LIVE 20R [78] with a resolution of 500 dpi to enrich datasets by collecting additional fingerprints. All 70 participants, including 45 males and 25 females, aged from 18 to 47 years, pressed on the sensor with normal pressure five times per

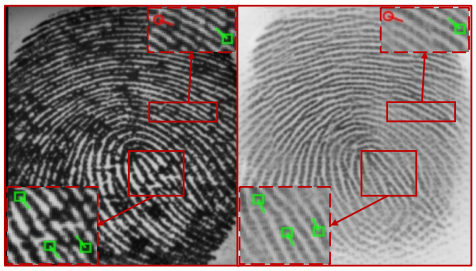
Table 1: Details of the datasets

Sensor	Images Num(Fingers Num)	
	Train	Test
CrossMatch	1790(829)	150(150)
Biometrika	1546(374)	100(100)
GreenBit	681(146)	50(50)
ZKTeco LIVE 20R	1875(406)	150(150)
Total	5892(1755)	450(450)

finger (except for the little finger). Thus we obtained 560 fingers and 2800 fingerprint images. We performed screening on both the public and collected datasets, resulting in a total of 6,342 fingerprint images from 2,205 fingers that met our criteria. Subsequently, we divided these fingers into a training set and a testing set, using a ratio of 4:1. The training set comprises 5,892 images from 1,755 fingers, while the testing set consists of 450 images, each from a distinct finger. The details of the datasets used for training and testing are shown in Table 1. There are no overlapping subjects in the training and testing sets. The main motivation behind gathering additional data is that, after applying screening, the fingerprints within public datasets do not offer an adequate amount of training data. This limitation significantly hampers the performance of the attack. Consequently, we incorporate additional samples to augment the dataset, enhance sample diversity, and ensure the robustness of our attack model's training outcomes. In fact, if more public datasets become accessible in the future, there is a possibility that we could consider substituting the data we have collected. It is ensured the experiments follow the internal review board (IRB) protocol of the host institution.

8.3 Metrics

Before verifying *FingerFaker*, we build the AFRS at different security levels, which are defined by false match rate (FMR) values, i.e., 1%, 0.1%, and 0.01%, corresponding to level1, level2, and level3, respectively. A higher security level means better system security, and they are achieved by different thresholds that can be calculated



(a) Real fingerprint (b)Fake fingerprint

Figure 9: Comparison of real and fake fingerprints, where left (a) is a real one and right (b) is the fake one generated by *FingerFaker*. Enlarged plots show minutiae in these areas.

by a cross-matching approach based on the FVC2002 protocol [27]. We evaluate the results using attack success rate (ASR) and fingerprint quality. We attack all fingerprints in the testing set, and ASR is defined as the ratio of successfully attacked fingerprints to all fingerprints. It can be defined as:

$$ASR = \frac{S}{T}, \tag{19}$$

where S is the number of fingerprints successfully attacked, and T is the number of all target fingerprints. After a successful attack, we conduct a quality on the generated fingerprint using NFIQ2. For defining the security levels of the AFRS system, we use the false match rate (FMR) with the following:

$$FMR = \frac{N_{FM}}{N_{IA}}, \tag{20}$$

where N_{FM} is the number of error matches and N_{IA} is the number of impostor attempts, i.e., each fingerprint is matched to a different fingerprint except itself.

8.4 Evaluation protocol

In each attempt, we first select a fingerprint from the testing set and translate it into features (i.e., minutiae) by a specific method (i.e., MINDTCT for open-source AFRS based on minutiae matching and VeriFinger for COTS AFRS based on deep learning). Next, we enroll features of such a fingerprint into the AFRS. Then, we repeat the iteration of the targeted attack phase. To compare the fake fingerprint with the enrolled one, we translate it by the same method as the enrolled one, and a corresponding fingerprint matcher (i.e., Bozorth3 for open-source AFRS and VeriFinger for COTS AFRS) will be used to generate the match score between them. The attack will be counted to succeed if the match score exceeds the threshold (depending on different security levels) in limited iterations (here is 40). Otherwise, it will be counted as a failure. We launch an attempt on all the fingers in the testing set and count the results to show the performance of *FingerFaker*.

8.5 Overall Performance

We evaluate the performance of *FingerFaker* in breaking the AFRS of an open-source AFRS. We pre-train the fingerprint generator with 5892 fingerprints in the training set and attempt to generate fake fingerprints to match with fingerprints in the testing set. First, we show a random pair consisting of a real (a) and a fake (b) fingerprint in Figure 9, where we also show minutiae within two areas. The plot shows *FingerFaker* can generate fake fingerprints owning similar minutiae as the real one. Then, we evaluated *FingerFaker* statistically. Figure 10 shows the ASR of *FingerFaker* under different security levels. The result indicates that *FingerFaker* can achieve 97.78% ASR even at the highest security level (i.e., FMR = 0.01%), showing superior performance.

AFRSs will remove low-quality fingerprints to avoid false matching, which means the quality of a fake fingerprint is a significant metric for evaluating *FingerFaker*. Hence, we calculate the qualities of fake fingerprints that can successfully spoof the AFRS and compare them with the qualities of real ones. Figure 11 shows the distribution of qualities of fake fingerprints and real ones, respectively. The x-value represents the quality of real/fake fingerprints, and the y-value represents the rate of fingerprints evaluated as this quality. The average quality values for fake and real fingerprints are

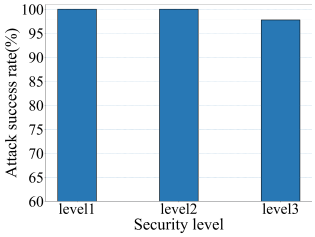


Figure 10: Overall performance.

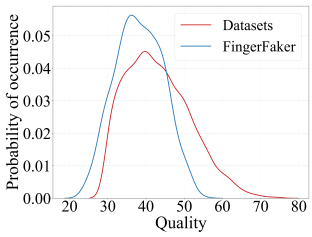


Figure 11: Quality distribution.

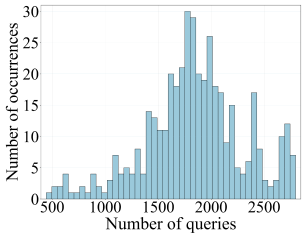


Figure 12: Number of queries.

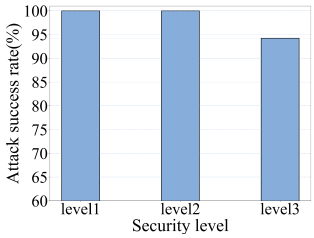


Figure 13: VeriFinger performance.

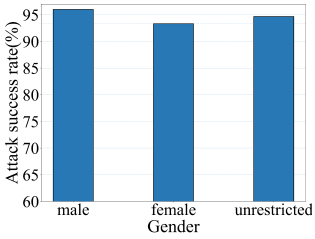


Figure 14: Gender performance.

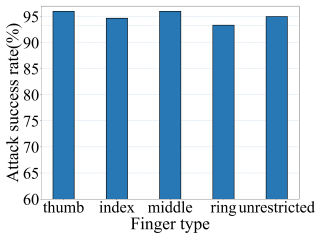


Figure 15: Different fingers.

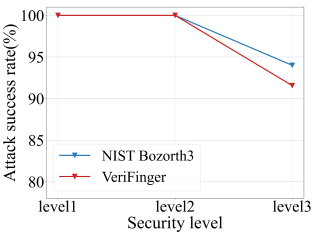


Figure 16: Cross-AFRSs: same algorithm.

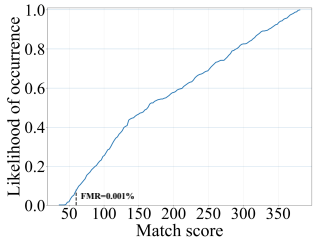


Figure 17: Cross-AFRSs: different algorithms.

38.18 and 43.14. The result indicates that the distribution and average of our proposed algorithm match those of the original datasets. Based on the result, *FingerFaker* can generate high-resolution fake fingerprints, whose quality is close to real ones, to make sure they get into the AFRS.

8.6 Number of Queries

The number of queries is a critical parameter in adversarial attacks. Hence, we count the number of queries of all successful spoofs in Section 8.5. Figure 12 shows the distribution of the number of queries, where the x-value represents the number of queries. The y-value shows the victim count that *FingerFaker* can generate his/her fake fingerprints with this number of quires. As shown in Figure 12, *FingerFaker* can spoof the AFRS with an average of **1839** queries. In the worst case, *FingerFaker* can also break the AFRS with 2797 queries. This result is significantly lower than state-of-the-art adversarial attacks in other authentication systems, i.e., **5000 for speaker verification** [14] and **10000 for face verification** [23]. Indeed, *FingerFaker* takes less than 1000 queries to attack 23 of the victims, whereas *FingerFaker* only queries 444 times in the fastest case to break the AFRS. Hence, *FingerFaker* can implement an effective spoofing attack within limited queries. Considering that too many queries may cause alarm, we can put them into different days, e.g., we can query 30 times a day and complete all within two weeks. Meanwhile, we further discuss the latent optimization of the number of queries in Section 10.

Table 2: Ablation Study

Method	Level3 (FMR=0.01%) ASR(%)	Quality
FingerFaker w/o T	95.11	29
FingerFaker w/o L	87.56	33
FingerFaker w/o P	8.33	18
FingeFaker(Ours)	**97.78**	**38**

T is a two-factor evolutionary strategy, P means pseudo-minutiae-set, and L means minutiae loss.

Table 3: The performance of different Ages.

Age	Level1 (FMR=1%)	Level2 (FMR=0.1%)	Level3 (FMR=0.01%)
	ASR(%)		
21-30	100.00	100.00	93.33
31-40	100.00	100.00	95.00
41-50	100.00	100.00	94.28

8.7 Ablation

We conduct the ablation experiment to quantify the gain of components of *FingerFaker*, including pseudo-minutiae-set (hereafter P), fingerprint generator(hereafter G), two-factor evolutionary strategy (hereafter T), and the minutiae loss (hereafter L). Specifically, we attempt to generate fake fingerprints by continuous feature vector when we experiment without P. When we test without our generator, we employed the state-of-the-art fingerprint generator [11] to replace our fingerprint generator, with both generators trained on the same dataset. Meanwhile, we generate fake fingerprints by only match-score when we test without T. Also, we generate fake fingerprints by the loss function of the previous work [56] when

Table 4: Comparison of FingerFaker with state-of-the-art works of spoofing AFRSs

Algorithm	Optimizer	Generator	FMR=0.01% ASR(%)	Attack without prior knowledge	Targeted Attack
Cappeli *et al.* [13]	/	Gabor filtering	23.00	✗	✓
Li and Kot [44]	/	AM-FM model	86.48	✗	✓
Feng and Jain [26]	/	AM-FM model	93.62	✗	✓
Cao and Jain [12]	/	AM-FM model	99.38	✗	✓
Bouzaglo *et al.* [11]	/	Conditional GANs	99.89	✗	✓
Roy *et al.* [59]	Hill climbing	/	1.88	✓	✗
FingeFaker(Ours)	Genetic algorithm	GAN-base generator	**94.22**	✓	✓

we experiment without L. Table 2 shows that the performance is reduced by 89.45% and 10.22% without pseudo-minutiae-set and the minutiae loss, respectively. And using other fingerprint generators will reduce the attack success rate by 17.11%. In addition, the two-factor evolutionary strategy brings about a 36% improvement in the quality of the fingerprint image while maintaining performance. Hence, our designs are efficient and contribute to boosting the performance of spoofing AFRSs. Table 2 illustrates that the absence of the pseudo-minutiae-set and minutiae loss results in performance degradation of 89.45% and 10.22%, respectively while utilizing other fingerprint generators leads to a reduction in the attack success rate of 17.11%. Furthermore, the implementation of the two-factor evolutionary strategy enhances the quality of the fingerprint image by 36%, while maintaining system performance. Consequently, our designs are efficient and effectively improve the performance of spoofing AFRSs.

9 REALISTIC STUDY ON COTS AFRS

9.1 Performance and Comparison

In this section, we employ the same evaluation protocol as outlined in Section 8.4. To be more specific, we have shifted our attack target to a COTS-AFRS, namely VeriFinger. We optimize fake fingerprints to match real ones in the testing set by querying the COTS AFRS. The result, as shown in Figure 13, indicates that *FingerFaker* can achieve 100% ASR in the lower two security levels (i.e., FMR = 1% and 0.1%) and still achieve 94.22% ASR in the highest security level, which indicates the high performance of *FingerFaker* in spoofing the COTS AFRS. In addition, we compare *FingerFaker* with **state-of-the-art** works of spoofing AFRS [10–13, 26, 44, 58, 59]. Several works [11–13, 26, 44, 58] attempt to generate fake fingerprints from the prior knowledge of the targeted fingerprint (i.e., minutiae) by mathematical models and deep learning models. Other works [59] attempt to conduct an untargeted attack, i.e., the attack is successful as long as the fake fingerprint can match with anyone in the enroll list. *FingerFaker* is the first work to generate fake fingerprints of the targeted victim to spoof the AFRS without the victim's prior knowledge. Specifically, we propose the concept of the pseudo-minutiae-set and design the GAN-based generator/genetic algorithm to generate fake fingerprints/optimize the pseudo-minutiae-set. As shown in Table 4, *FingerFaker* gets rid of the strong assumption on the minutiae of the target fingerprint and can get a competitive performance.

9.2 Impact of Gender

We are curious about whether the gender of the victim would affect the performance of *FingerFaker*. To this end, we select 75 men's fingerprints and 75 women's ones from the testing set to test the performance of *FingerFaker* and attempt to generate fake fingerprints to match with them. As shown in Figure 14, there is a 2.67% difference in ASR between attacking men's fingerprints and women's ones, which has no significant difference. Thus, the gender of the victim has little effect on *FingerFaker*.

9.3 Impact of Age

As individuals age, their fingerprints may wear, potentially affecting the performance of *FingerFaker*. To assess the impact of age on *FingerFaker*, we conducted experiments on the fingerprints of 120 individuals aged between 21 and 49. These fingerprints were divided into three groups, each spanning a 10-year age range. Subsequently, we registered these fingerprints in COTS-AFRSs at different security levels and used *FingerFaker* to generate counterfeit fingerprints. Table 3 demonstrates that *FingerFaker* successfully deceived AFRSs that had already registered fingerprints from different age groups. Furthermore, our analysis revealed that the ASR did not show significant differences among different age groups, indicating that *FingerFaker*'s performance remains consistent regardless of the age of the individuals being attacked.

9.4 Impact of Sensor

Victims can enroll fingerprints with different fingerprint sensors. Thus, we explore the performance of *FingerFaker* on spoofing AFRSs enrolled through different fingerprint sensors in the testing set. As shown in Table 5, *FingerFaker* can achieve high performance in spoofing AFRS enrolled by different fingerprint sensors, which indicates that fingerprint sensors do not affect the performance of *FingerFaker*.

9.5 Impact of Different Fingers

Different fingers have different shapes and sizes. Thereby, it is interesting to verify whether different fingers of victims affect the performance of *FingerFaker*. To explore the impact of different fingers, we select four types of fingers (i.e., thumb, index finger, middle, and ring finger) and randomly select 75 fingerprints for each type from the testing set and test the performance of *FingerFaker* by attacking the selected fingerprints. Figure 15 shows the performance of attacking different fingers, where the ASR varies from 93.33% to 96%. The result indicates that the adversary can effectively attack different fingers leveraging *FingerFaker*.

9.6 Cross-AFRS Attack

Although most AFRSs return a match score, some do not. To enhance *FingerFaker*, we have designed a cross-AFRS attack. In this attack, we select another AFRS as the target and generate fake fingerprints to spoof the system. The generated fingerprints are then used to break the targeted AFRS. We consider two attack scenarios.
Scenario 1: the targeted and the attacked AFRSs use the same algorithm to achieve fingerprint authentication, but they are enrolled by the same fingerprint at different times. Fingerprint recognition is a mature technology with limited algorithms [2]. Thus, it is reasonable that two AFRSs use the same algorithm.
Scenario 2: the targeted and the attacked AFRSs use different algorithms while the victim enrolls them by the same fingerprint at different times. Specifically, we choose two common AFRSs (i.e., NIST and VeriFinger) to test the performance because they are widely used in daily life [10, 41] and research [11, 12, 26, 44, 53, 59].
Setup: We use the testing part (in Table 1) to enroll the targeted AFRS. Meanwhile, we enroll the attacked one with other fingerprints from the same fingers. Next, we deploy *FingerFaker* to spoof the attacked AFRS, which can get fake fingerprints and work on the targeted AFRS.
Result: Figure 16 shows the performance under scenario 1. When the two AFRSs both use NIST, the attack success rate achieves 94%. When they both use VeriFinger, the attack success rate reaches 91.56%. The result shows *FingerFaker* can effectively spoof the targeted AFRS by the cross-AFRS attack when the two AFRSs use the same algorithm.

Figure 17 illustrates the cumulative distribution function (CDF) of the match scores for fake fingerprints generated by *FingerFaker* when the targeted AFRS uses NIST and the attacked AFRS uses VeriFinger. The y-axis represents the rate of the match score of fake fingerprints below the corresponding x-value. Our results demonstrate that 86.9% of fake fingerprints can successfully spoof the targeted AFRS when the AFRS is at security level 1, whereas 75.33% of fake fingerprints can break the highest security level. These findings suggest that different architectures of AFRSs may lead to performance degradation, as previously reported in state-of-the-art adversary attacks on speaker identification systems [14]. Specifically, NIST employs a traditional minutiae matching approach (detailed in Section 3), while VeriFinger claims to use a deep learning-based method for minutiae extraction and matching [52]. They extract completely different features. VeriFinger can extract richer and more extensive minutiae, which is the reason for the performance degradation. Despite the performance degradation, *FingerFaker* is still capable of conducting cross-AFRS attacks on more than three-fourths of fingerprints, highlighting the practical impact of such an attack. However, it is challenging to spoof the VeriFinger-based AFRS by attacking the NIST-based AFRS (less than 30% success rate under the highest security level) since the VeriFinger has a superior performance than NIST [60]. Hence, the adversary can select another VeriFinger-based AFRS to launch the attack.

Table 5: The performance of different sensors.

Sensor	Level1 (FMR=1%)	Level2 (FMR=0.1%)	Level3 (FMR=0.01%)
	ASR(%)		
CrossMatch	100.00	100.00	95.33
Biometrika	100.00	100.00	95.00
GreenBit	100.00	100.00	94.00
ZKTeco 20R	100.00	100.00	93.33

10 DISCUSSION AND FUTURE WORKS

The Number of Queries Reduction: The issue of high query costs is common among adversarial attacks. However, *FingerFaker* requires fewer queries than state-of-the-art adversarial attacks in other authentication systems, such as 5000 for speaker verification [14] and 10000 for face verification [23]. The fact shows the superiority of *FingerFaker*. Furthermore, many studies are currently attempting to reduce the query cost of adversarial attacks. For example, building a substitution model [46] can reduce queries in adversarial attacks. Additionally, a previous work [34] combines the substitution model and the return of the target system to achieve a balance between performance and queries. *FingerFaker* aims to reveal the latent risks of AFRS, such as adversaries breaking the safeguard without the victims' knowledge, while future works can focus on reducing query costs by above works.
Hard-Label Attacks: In previous literature, there have been many hard-label black-box attacks targeting adversarial machine learning [17, 24, 66]. Nevertheless, in our current attack setting (i.e., verification), these attacks prove to be ineffective, as hard-label attacks only yield binary results of 0 and 1. It's worth noting that if we were to transition to an identification scenario, the use of hard-label attacks on *FingerFaker* would become a viable option.
Liveness Detection: Liveness detection is a mainstream fingerprint anti-counterfeiting technology. However, it is known to have a negative impact on AFRSs' usability due to the burden of calculation and the high false rejection rate [69]. As a result, it is not suitable for high-efficiency scenarios where previous research [57] has shown that COTS AFRSs can be broken by fake fingerprints. Consequently, *FingerFaker* can effectively target such AFRSs. Moreover, several studies [3, 47, 75] have demonstrated that spoofing attacks aimed at AFRSs with liveness detection can also generate high-precision fake fingerprints. As a result, *FingerFaker* can also be used to break AFRSs with liveness detection.
The size of the "pseudo-minutiae-set": The size of FingerFaker

plays a crucial role in determining its performance, as it must be large enough to accurately predict the information present in the target fingerprint. However, a larger size also results in an increased number of attempts, necessitating more queries to achieve results. As a result, we have made a deliberate trade-off by setting the size of FingerFaker to 30.

Printing FingerFaker in the physical world: The exploration of applying *FingerFaker* in the physical world presents an intriguing avenue of research. One potential approach involves using a laser engraving machine to etch counterfeit fingerprints onto a conductive silicon film, or alternatively, employing a 3D printer to create a physical fingerprint membrane. [3, 25] These fabricated fingerprint films could then be employed to press onto the target fingerprint scanner(e.g., an iPhone or an Android device), enabling deception attacks within the physical domain. However, it is imperative to address the mitigation of physical noise during the printing process, including potential manufacturing errors, as these inaccuracies can significantly diminish the effectiveness of attacks in a physical context. This represents a challenging area, and we intend to continue our investigation into this topic in the future.

11 COUNTERMEASURES

Fake detector: The first countermeasure is to train a detector to distinguish between fake and real fingerprints. Previous works [31, 45, 70, 77] use real and fake fingerprints (generated by the targeted attack) to train the detector. However, these approaches require a large number of fake fingerprints to train the detector and increase the false rejection rate.

Device authentication: The second countermeasure is to require recognition requests to be made by trusted devices, i.e., add device authentication [1, 67] to provide higher security. Users must provide and update a list of trusted devices for the supplier to prevent the adversary from querying the AFRS with their devices.

Stricter error limits: The third countermeasure is to reduce the number of errors allowed by the AFRS. Because *FingerFaker* needs to query the AFRS, such queries are errors for the system. Therefore, the AFRS can set an upper bound of errors to defend against such attacks. However, extremely low error bounds will greatly reduce the system's availability. One possible compromise is to reset the error count when the user successfully authenticates. However, the attacker can assume that the user normally uses the system a certain number of times per day, which allows the attacker to get enough queries in multiple intervals.

12 RELATED WORK

Reconstruction of fingerprints: Breaking security of fingerprint systems has been explored by previous works, where they spoof AFRSs by reconstructing fingerprints from prior knowledge (e.g., minutiae) of the target fingerprint [11, 12, 26, 44, 58]. For instance, mathematical modeling between minutiae and the fingerprint was proposed to reconstruct fake fingerprints [12, 26, 44, 58]. Recently, a deep-learning-based approach also achieved superior performance in spoofing AFRSs [11]. However, these attacks are based on the strong assumption of prior knowledge of the target fingerprint, where the adversary needs to obtain minutiae protected by multiple security [30, 43, 65] from the database. Therefore, these attacks are hard to implement in real scenarios and lack practical value. Our work explores the vulnerability and designs an attack, namely *FingerFaker*, to spoof AFRSs without the target fingerprint/minutiae, which leads to a practical attack to break fingerprint-based security to threaten the victim in a real scenario.

Adversarial attacks: In past decades, efforts have been devoted to investigating spoofing biometrics systems leveraging adversarial attacks, which threaten biometrics-based security [14, 54, 63, 74]. In the beginning, researchers find the vulnerability of the decision boundary of deep learning is also common in facial recognition systems and voice recognition systems, which inspires them to propose adversarial attacks in such a task and achieve effective spoofing attacks [31, 51]. Recently, an attack aims at generating a general fake fingerprint leveraging conventional adversarial attacks [10]. However, it only achieves a limited performance due to the lack of adaptation and adjustment to the attack of AFRSs. Specifically, conventional adversarial attacks based on GANs cannot preserve the "pseudo-minutiae-set", which leads to poor performance. Our work proposes an advanced generator based on the concept of "pseudo-minutiae-set" to tackle areas of weakness, which realizes the high-performance adversarial attack on AFRSs.

13 CONCLUSION

In this paper, we design a novel attack, *FingerFaker*, to spoof COTS AFRSs. Specifically, we design a two-stage scheme to generate high-resolution fake fingerprints, where we propose the concept of "pseudo-minutiae-set" to remove the dependency on prior knowledge of the fingerprint and design a two-factor evolutionary strategy to optimize such features for generating a high-resolution fake fingerprint, meanwhile, a well-designed loss function is used in a GAN-based training strategy to preserve "pseudo-minutiae-set" in fake fingerprints. Our evaluation shows the high performance of *FingerFaker* on both open-source and COTS AFRSs. In addition, we also propose several countermeasures to mitigate the harm of *FingerFaker*.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China under grant 2020AAA0107700, the National Natural Science Foundation of China under grants 62032021, 62372406, 61972348, 62172359, and 62102354.

REFERENCES

[1] Inayat Ali, Sonia Sabir, and Zahid Ullah. 2019. Internet of things security, device authentication and access control: a review. *arXiv preprint arXiv:1901.07309* (2019).
[2] Mouad MH Ali, Vivek H Mahale, Pravin Yannawar, and AT Gaikwad. 2016. Overview of fingerprint recognition system. In *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*. IEEE, 1334–1338.
[3] Sunpreet S Arora, Kai Cao, Anil K Jain, and Nicholas G Paulter. 2016. Design and fabrication of 3D fingerprint targets. *IEEE Transactions on Information Forensics and Security* 11, 10 (2016), 2284–2297.
[4] M Arul Selvan and S Selvakumar. 2019. Malicious node identification using quantitative intrusion detection techniques in MANET. *Cluster computing* 22, 3 (2019), 7069–7077.
[5] Aware. 2023. Aware. https://www.aware.com/identification-verification/.
[6] Roli Bansal, Priti Sehgal, and Punam Bedi. 2011. Minutiae extraction from fingerprint images-a review. *arXiv preprint arXiv:1201.1422* (2011).
[7] Daniel Bichler, Guido Stromberg, Mario Huemer, and Manuel Löw. 2007. Key generation based on acceleration data of shaking processes. In *UbiComp 2007:*

Ubiquitous Computing: 9th International Conference, UbiComp 2007, Innsbruck, Austria, September 16-19, 2007. Proceedings 9. Springer, 304–317.
[8] BioKey. 2023. BioKey. https://www.bio-key.com/portalguard/.
[9] BioKey. 2023. BioKey Case Study. https://www.bio-key.com/resources/?type=case-studies.
[10] Philip Bontrager, Aditi Roy, Julian Togelius, Nasir Memon, and Arun Ross. 2018. Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. IEEE, 1–9.
[11] Rafael Bouzaglo and Yosi Keller. 2022. Synthesis and reconstruction of fingerprints using generative adversarial networks. *arXiv preprint arXiv:2201.06164* (2022).
[12] Kai Cao and Anil K Jain. 2014. Learning fingerprint reconstruction: From minutiae to image. *IEEE Transactions on information forensics and security* 10, 1 (2014), 104–117.
[13] Raffaele Cappelli, Dario Maio, Alessandra Lumini, and Davide Maltoni. 2007. Fingerprint image reconstruction from standard templates. *IEEE transactions on pattern analysis and machine intelligence* 29, 9 (2007), 1489–1503.
[14] Guangke Chen, Sen Chenb, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. 2021. Who is real bob? adversarial attacks on speaker recognition systems. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 694–711.
[15] Jiansheng Chen and Yiu Sang Moon. 2008. The statistical modelling of fingerprint minutiae distribution with implications for fingerprint individuality studies. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society.
[16] Yu Chen, Yang Yu, and Lidong Zhai. 2023. InfinityGauntlet: Expose Smartphone Fingerprint Authentication to Brute-force Attack. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 2027–2041. https://www.usenix.org/conference/usenixsecurity23/presentation/chen-yu
[17] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. 2018. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457* (2018).
[18] Ana Cholakoska, Bjarne Pfitzner, Hristijan Gjoreski, Valentin Rakovic, Bert Arnrich, and Marija Kalendar. 2021. Differentially Private Federated Learningfor Anomaly Detection in eHealth Networks. In *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*. 514–518.
[19] CloudABIS. 2023. CloudABIS. https://www.m2sys.com/.
[20] CloudABIS. 2023. CloudABIS Case Study. https://www.m2sys.com/biometric-fingerprint-software-case-studies/.
[21] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. 1995. Simulated binary crossover for continuous search space. *Complex systems* 9, 2 (1995), 115–148.
[22] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.
[23] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. 2019. Efficient decision-based black-box adversarial attacks on face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7714–7722.
[24] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. 2019. Efficient decision-based black-box adversarial attacks on face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7714–7722.
[25] Joshua J Engelsma, Sunpreet S Arora, Anil K Jain, and Nicholas G Paulter. 2018. Universal 3D wearable fingerprint targets: Advancing fingerprint reader evaluations. *IEEE Transactions on Information Forensics and Security* 13, 6 (2018), 1564–1578.
[26] Jianjiang Feng and Anil K Jain. 2010. Fingerprint reconstruction: from minutiae to phase. *IEEE transactions on pattern analysis and machine intelligence* 33, 2 (2010), 209–223.
[27] FVC2002. 2002. Fingerprint Verification Competition. http://bias.csr.unibo.it/fvc2002/.
[28] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. 2016. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.* 45, 3 (2016), 882–929.
[29] Luca Ghiani, David A. Yambay, Valerio Mura, Gian Luca Marcialis, Fabio Roli, and Stephanie Schuckers. 2017. Review of the Fingerprint Liveness Detection (LivDet) competition series: 2009 to 2015. *Image Vis. Comput.* 58 (2017), 110–128.
[30] Marta Gomez-Barrero, Emanuele Maiorana, Javier Galbally, Patrizio Campisi, and Julian Fiérrez. 2017. Multi-biometric template protection based on Homomorphic Encryption. *Pattern Recognit.* 67 (2017), 149–163.
[31] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[32] Google. 2023. Android-12-cdd biometric sensors. https://source.android.com/docs/security/features/biometric.
[33] Lin Hong, Yifei Wan, and Anil Jain. 1998. Fingerprint image enhancement: Algorithm and performance evaluation. *IEEE transactions on pattern analysis and machine intelligence* 20, 8 (1998), 777–789.
[34] Zhichao Huang and Tong Zhang. 2019. Black-box adversarial attack with transferable model-based embedding. *arXiv preprint arXiv:1911.07140* (2019).
[35] Innovatrics. 2023. Innovatrics. https://www.innovatrics.com/innovatrics-abis/.
[36] Xudong Jiang and Wei-Yun Yau. 2000. Fingerprint minutiae matching based on the local and global structures. In *Proceedings 15th international conference on pattern recognition. ICPR-2000*, Vol. 2. IEEE, 1038–1041.
[37] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. 1988. Design of an image edge detection filter using the Sobel operator. *IEEE Journal of solid-state circuits* 23, 2 (1988), 358–367.
[38] Sieeka Khan. 2019. Samsung Galaxy S10 Fingerprint Scanner Hacked. https://www.sciencetimes.com/articles/19758/20190406/samsung.htm.
[39] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[40] Zoe Kleinman. 2014. Politician's fingerprint 'cloned from photos' by hacker. https://www.bbc.com/news/technology-30623611.
[41] Kenneth Ko et al. 2007. User's guide to nist biometric image software (nbis). (2007).
[42] John R Koza and Riccardo Poli. 2005. Genetic programming. In *Search methodologies*. Springer, 127–164.
[43] Cai Li and Jiankun Hu. 2016. A Security-Enhanced Alignment-Free Fuzzy Vault-Based Fingerprint Cryptosystem Using Pair-Polar Minutiae Structures. *IEEE Trans. Inf. Forensics Secur.* 11, 3 (2016), 543–555.
[44] Sheng Li and Alex C Kot. 2012. An improved scheme for full fingerprint reconstruction. *IEEE Transactions on Information Forensics and Security* 7, 6 (2012), 1906–1912.
[45] Zhenguang Liu, Peng Qian, Xiaoyang Wang, Yuan Zhuang, Lin Qiu, and Xun Wang. 2021. Combining Graph Neural Networks with Expert Knowledge for Smart Contract Vulnerability Detection. *TKDE* (2021). https://doi.org/10.1109/TKDE.2021.3095196
[46] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
[47] Davide Maltoni, Dario Maio, Anil K Jain, and Salil Prabhakar. 2009. Synthetic fingerprint generation. *Handbook of fingerprint recognition* (2009), 271–302.
[48] Anna Mikaelyan and Josef Bigun. 2012. Ground truth and evaluation for latent fingerprint matching. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 83–88.
[49] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).
[50] Farid Ghareh Mohammadi, Farzan Shenavarmasouleh, M Hadi Amini, and Hamid R Arabnia. 2020. Malware detection using artificial bee colony algorithm. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*. 568–572.
[51] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.
[52] Neurotechnology. 2021. Verifinger SDK. https://www.neurotechnology.com/verifinger.html.
[53] Neurotechnology. 2022. Verifinger Case Studies. https://www.neurotechnology.com/cgi-bin/customers.cgi.
[54] Dinh-Luan Nguyen, Sunpreet S Arora, Yuhang Wu, and Hao Yang. 2020. Adversarial light projection attacks on face recognition systems: A feasibility study. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 814–815.
[55] NIST. 2010. NIST Biometric Image Software (NBIS). https://www.nist.gov/services-resources/software/nist-biometric-image-software-nbis.
[56] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2337–2346.
[57] Aditya Singh Rathore, Yijie Shen, Chenhan Xu, Jacob Snyderman, Jinsong Han, Fan Zhang, Zhengxiong Li, Feng Lin, Wenyao Xu, and Kui Ren. 2022. FakeGuard: Exploring Haptic Response to Mitigate the Vulnerability in Commercial Fingerprint Anti-Spoofing. *Proceedings 2022 Network and Distributed System Security Symposium* (2022).
[58] Arun Ross, Jidnya Shah, and Anil K Jain. 2007. From template to image: Reconstructing fingerprints from minutiae points. *IEEE transactions on pattern analysis and machine intelligence* 29, 4 (2007), 544–560.
[59] Aditi Roy, Nasir Memon, and Arun Ross. 2017. Masterprint: Exploring the vulnerability of partial fingerprint-based authentication systems. *IEEE Transactions on Information Forensics and Security* 12, 9 (2017), 2013–2025.
[60] Anush Sankaran, Tejas I Dhamecha, Mayank Vatsa, and Richa Singh. 2011. On matching latent to latent fingerprints. In *2011 international joint conference on biometrics (IJCB)*. IEEE, 1–6.
[61] Mathew J. Schwartz. 2014. Apple iPhone 6 Touch ID Hacked. https://www.bankinfosecurity.com/apple-iphone-6-touchid-hacked-a-7348.
[62] SecuGen. 2023. SecuGen. https://secugen.com/products/webapi/.

[63] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*. 1528–1540.

[64] Meng Shen, Zelin Liao, Liehuang Zhu, Ke Xu, and Xiaojiang Du. 2019. Vla: A practical visible light-based attack on face recognition systems in physical world. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), 1–19.

[65] Erez Shmueli, Ronen Vaisenberg, Ehud Gudes, and Yuval Elovici. 2014. Implementing a database encryption solution, design and implementation issues. *Comput. Secur.* 44 (2014), 33–50.

[66] Satya Narayan Shukla, Anit Kumar Sahu, Devin Willmott, and Zico Kolter. 2021. Simple and efficient hard label black-box adversarial attacks in low query budget regimes. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1461–1469.

[67] G Edward Suh and Srinivas Devadas. 2007. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*. IEEE, 9–14.

[68] S Supatmi and ID Sumitra. 2020. Fingerprint Matching Using Bozorth3 Algorithm and Parallel Computation on NVIDIA Compute Unified Device Architecture. In *IOP Conference Series: Materials Science and Engineering*, Vol. 879. IOP Publishing, 012109.

[69] Amirhosein Toosi, Andrea Bottino, Sandro Cumani, Pablo Negri, and Pietro Luca Sottile. 2017. Feature fusion for fingerprint liveness detection: a comparative study. *IEEE Access* 5 (2017), 23695–23709.

[70] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204* (2017).

[71] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022* (2016).

[72] Ruxin Wang, Congying Han, and Tiande Guo. 2016. A novel fingerprint classification method based on deep learning. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 931–936.

[73] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8798–8807.

[74] Lei Zhang, Yan Meng, Jiahao Yu, Chong Xiang, Brandon Falk, and Haojin Zhu. 2020. Voiceprint mimicry attack towards speaker verification system in smart home. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 377–386.

[75] Qijun Zhao, Anil K Jain, Nicholas G Paulter, and Melissa Taylor. 2012. Fingerprint image synthesis based on statistical feature models. In *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*. IEEE, 23–30.

[76] Yongfang Zhu, Sarat C. Dass, and Anil K. Jain. 2007. Statistical Models for Assessing the Individuality of Fingerprints. *IEEE Trans. Inf. Forensics Secur.* 2, 3-1 (2007), 391–401.

[77] Yuan Zhuang, Zhenguang Liu, Peng Qian, Qi Liu, Xiang Wang, and Qinming He. 2020. Smart Contract Vulnerability Detection using Graph Neural Network. In *IJCAI*. 3283–3290. https://doi.org/10.24963/ijcai.2020/454

[78] ZKTeco. 2021. ZKTeco LIVE 20R. https://www.ebay.com/itm/134070768618.