Beware of Your Screen: Anonymous Fingerprinting of Device Screens for Off-line Payment Protection

Zhe Zhou
Fudan University
zhouzhe@fudan.edu.cn

Di Tang
CUHK
td016@ie.cuhk.edu.hk

Wenhao Wang
SKLOIS, Institute of Information Engineering, CAS
wangwenhao@iie.ac.cn

Xiaofeng Wang
IU Bloomington
xw7@indiana.edu

Zhou Li
University of California, Irvine
zhou.li@uci.edu

Kehuan Zhang
CUHK
khzhang@cuhk.edu.hk

ABSTRACT

QR-code mobile payment becomes increasingly popular, being offered by major banks (e.g., ICBC) and payment service providers (e.g., PayPal). Unlike mobile payment solutions provided by hardware vendors (e.g., Apple Pay and Samsung Pay), QR code payment schemes do not rely on any hardware support and can therefore be easily deployed. However, the security guarantee of the new scheme is less clear: in the absence of hardware protection, users' digital wallet can be vulnerable to an OS-level adversary, who could steal her secret for generating payment tokens.

We find that the physical features of a phone's screen can enhance the security protection of this QR-code payment, serving as a second-factor authentication. Due to manufacturing imperfections, the luminance levels of the pixels on the screen vary across the screen's display area, which can be used to uniquely characterize the screen. This physical fingerprint cannot be stolen even when the OS is fully compromised, since the adversary cannot observe the physical features of the screen.

However, screen fingerprinting could also undermine the mobile payer's privacy, as less trusted merchants could use it to track customers and infer their purchase history and preferences. In this paper, we propose a new authentication solution that *anonymously fingerprints mobile screens*. The approach, called *AnonPrint* obfuscates a screen, which hides its fingerprint from the merchants. In the meantime, the payment provider, who shares a secret with the payer, is able to reconstruct the mask and authenticate the payer through her obfuscated fingerprint.

ACM Reference Format:
Zhe Zhou, Di Tang, Wenhao Wang, Xiaofeng Wang, Zhou Li, and Kehuan Zhang. 2018. Beware of Your Screen: Anonymous Fingerprinting of Device Screens for Off-line Payment Protection. In *2018 Annual Computer Security Applications Conference (ACSAC '18), December 3–7, 2018, San Juan, PR, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3274694.3274721

ACSAC '18, December 3–7, 2018, San Juan, PR, USA

ACM ISBN 978-1-4503-6569-7/18/12…$15.00
https://doi.org/10.1145/3274694.3274721

1 INTRODUCTION

Major payment service providers, including PayPal, Alipay, WeChat Pay (Chinese providers with hundreds of millions users) and ICBC (the largest bank in the world), have started to allow their customers to generate QR code on their phones using the secret of their digital wallet to do payment (Section 2), even in the absence of communication with the services. This kind of payment is extremely popular now and is carrying transactions worth trillions of US dollar last year [4]. Such an *off-line mobile payment* scheme features a design oriented towards ease to use. A problem here, however, comes from the limited understanding of its security guarantee: unlike other payment solutions, like Apple Pay and Samsung Pay, the QR-code payment does not have hardware protection and therefore is vulnerable to an attack on the mobile OS. What's worse, a recent work discovered that an adversary can sniff a QR code from one mobile device using a malicious app, without breaking the OS, and use it at another place, causing monetary loss to buyer or merchant [8].

Mobile display identification. Our study shows that this emerging payment solution can actually be better protected by leveraging the physical features that characterize the screen of the payer's smartphone. More specifically, previous works figured out that each screen has its unique physical features that can be detected by specialized devices. Specifically, its back light that produces luminance, which cannot be made uniform across all pixels on the screen during the manufacturing process [23]. Such features can also be picked up by a close-in camera in lower accuracy, as the one used in a commercial POS scanner, during code scanning[1]. Identification of such a fingerprint, however, is nontrivial, due largely to the presence of the projective distortions resulted from random scanning angles, which may cause the same screen look differently in different scans. We developed a new approach that corrects such distortions by projecting the image of a screen to a fixed observing plane and then reliably extracting the identifying features from the projection in a highly efficient way (Section 3).

Convenient and reliable screen fingerprinting has significant security and privacy implications, particularly for the off-line payment. The unique physical features of the screen can serve as a natural means for a second-factor authentication: even when the user's secret (e.g., the secret key for generating QR codes) is stolen, even when her phone has been *fully compromised* by the adversary,

[1] Some high-end scanners use laser to light up the code, but they also include a camera to shoot the code.

still no one can perform an off-line payment transaction successfully without accessing her physical device or synthesizing the features of her screen, which we found is hard in the absence of a high-resolution picture of the screen. On the other hand, the screen fingerprint of the payer's phone makes her harder to stay anonymous during her purchases, because the vendor could take advantage of the opportunity when scanning the QR code displayed on her phone to capture the fingerprint from the screen. This information, once exposed to a third party, such as an advertising (ad) company, can be used to link the same device's transactions across different stores, enabling the party to track the customer's purchase activities.

Anonymous screen fingerprinting. An intriguing challenge here is how to enable service provider to utilize the screen to enhance security protection while preserve the privacy at the same time. This is by no means trivial: again, in the payment scenario, we need to hide the physical features of one's display to the POS scanner, which is under the control of the vendor, but somehow communicate them to the payment service provider in a reliable way; also the operating system is considered to be untrusted and any protection working here needs to be built on top of the screen's physical properties. In our research, we found that all these aims can actually be achieved, through a carefully designed technique called *AnonPrint*.

AnonPrint is designed to use randomly generated visual *one-time masks* (a pixel pattern with dots set to various brightness levels) to obfuscate the distinguishable features of a user's screen. Such a mask is designed to hide the physical properties of a screen, and meant to be indistinguishable from a real fingerprint. In the meantime, for the party that knows the mask, such as the payment service provider, it can still verify whether the features collected from the protected screen are indeed related to the authorized device. More specifically, our approach randomly creates a smooth textured pattern for each transaction (which is also known to the provider), and displays such a pattern as the background of QR code to disarrange the brightness of a screen, in line with its real-world physical properties: neighboring dots are correlated and the levels of brightness change smoothly. Leveraging our feature extraction technique, such an obfuscated feature can still be reliably captured by the scanner and delivered to the provider. The provider can then re-obfuscate the fingerprint of the authorized device with the same mask, and run a correlation test on both the synthesized fingerprint and the one reported (by the vendor) to determine whether it comes from the authorized party.

We implemented this technique and evaluated it over up to 100 smartphones. Our experiment results show that these devices can be reliably differentiated from each other with or without random masks. Also, once the masks were applied, we found that the correlations between the fingerprints extracted from the same device (under different masks) dropped to the level of those between two different devices, indicating that the attempt to establish a link between different payment transactions, as mentioned earlier, will fail.

Contributions. The contributions of the paper are outlined as follows:

- *New discovery.* We found that screen luminance unevenness can be used to uniquely fingerprint a device and developed a new technique that reliably and efficiently extracts such features from each screen.
- *New technique.* We analyzed the security and privacy implications of the screen fingerprinting, under the settings of off-line mobile payment, and highlight the privacy risk of correlating different payment transactions and the security opportunity of using the features as a reliable means for a second factor authentication. Further, we developed an innovative technique to *anonymously* recover the fingerprint from each device, protecting the user from the curious vendor, as well as the OS-level adversary.
- *Implementation and evaluation.* We implemented our design and evaluated it over a large number of real-world smartphones. Our experiment results show that the new technique can effectively protect the user from both untrusted vendors and the untrusted OS during an off-line payment.

2 BACKGROUND

In this section, we introduce the background about the uneven brightness property of screen and off-line QR payment, and then present the adversary model of our research.

2.1 Screen

Brightness unevenness. Key to our screen fingerprinting technique is the observation that there are subtle differences in the maximum luminance of the pixels at different locations of the screen. Taking LCD screens as an example (see Figure 1), the light emitted by the back light unit has to come across many layers to get adjusted and finally perceived by human. Due to the imperfection in the screen manufacturing, which could bring defects to its components, the light from the back light unit can be spread unevenly over the screen, making the border darker than the center and causing the Mura effect. The transparency of the polarizer varies at different parts of the screen, and therefore the default angle for bending the polarized lights can be varied as well when no voltage is applied. Although on a qualified product, all such subtle defects are barely observable to human eyes, they can however be reliably identified by a camera, as observed in our study.

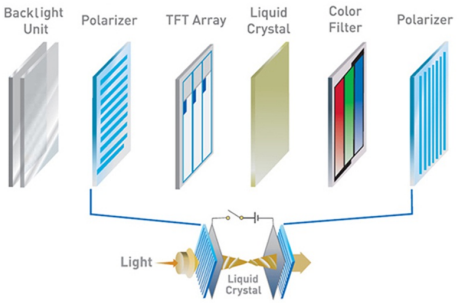

Figure 1: Typical structure of a LCD screen.

Non-LCD screens. Not all smartphones have LCD screens. As a prominent example, Samsung products typically use AMOLED

panels that have no back-end light source, and instead, they are built upon organic light emitting diode, with every pixel emitting lights of different strengthes. Those panels also have luminance unevenness, which can be picked up by the same technique we developed for analyzing screens, as shown by our evaluations.

2.2 Off-line QR Payment

Figure 2: A POS supporting QR code payment.

QR-code payment is gaining popularity in recent years mainly because of its ease of use. When a buyer checks out her purchases, she only needs to run a *wallet* app and clicks on its "pay" button to show a one-time QR code on the screen , and presents the screen close to the camera of the vendor's POS machine for scanning. After the vendor enters the right amount the buyer needs to pay, the scan starts and the POS machine will notify both parties if the transaction is approved by the payment service provider (see Figure 2 for POS and [2] for more details).

Payment transaction. To use the payment scheme, the payer needs to set up an account with the payment provider (e.g., Alipay), install its wallet app and log in her account through the app when she is ready to pay. The app contains a piece of secret downloaded from the provider for generating one-time tokens.

When the payer initiates a payment transaction (by pressing the payment button of the wallet app), the app proceeds as illustrated in Figure 3 to complete the transaction. Specifically, it uses the current time, the user ID and the stored secret as inputs to generate a random token (usually using HMAC), which is then encoded into a QR code to be shown on the payer's screen. After being scanned into the POS system, the token is extracted from the scanned image, together with other transaction information like the total amount and currency type, and sent to the payment provider.

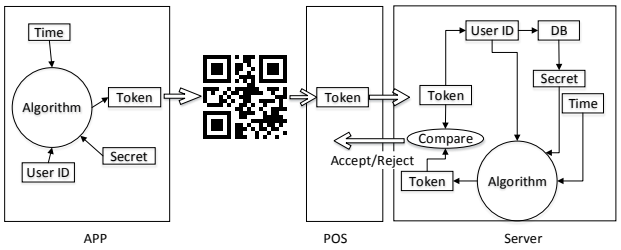

Figure 3: Off-line QR code payment flow chart.

The provider, upon receiving the token and transaction information, recovers from the token the user ID and uses the information to retrieve the stored secret of the payer, in order to verify the token. When the token is valid and still fresh (within a given time frame), the provider checks whether payer's fund is sufficient, and notifies the vendor of the acceptance of the payment.

Threat of OS-level and app-level attack. Compared with other mobile payment schemes, however, the QR code payment is limited in its capability to defend against OS-level adversary. All off-line payment schemes rely on the secret stored on the mobile device, which is often protected through hardware means. For example, Apple Pay uses the secure element to store payment secret and runs the payment process within its secure enclave. Samsung Pay leverages its KNOX (built upon top of ARM trust-zone) to protect the payment information in its secure element. They can all mitigate the threat from the OS-level attackers, since the payment secret stays within the hardware component the OS cannot directly access. For the payment schemes not provided by the device manufacturers, like PayPal, Alipay, etc., the hardware level protection is not in place and can therefore only trust the OS to protect the payment process. Once the OS is compromised (particularly on rooted or jail-broken devices), the adversary can then get access to the payment secret to generate legitimate payment token to steal from the victim. Actually, during our research, we analyzed AliPay on a jail-broken device and successfully extracted the secret.

Even if the mobile OS is not tampered, a recent study showed that a malicious app is able to extract payment token during the off-line payment scenario [8]. In particular, the reflection of the QR code will appear on the glass of POS scanner when user's mobile device comes close, and the malicious app can sniff it by taking a picture of the scanner glass.

2.3 Adversary Model

An off-line payment transaction involves four actors: the payer, the OS, the vendor and the payment provider. In our research, we assume the provider to be trusted and collaborative, willing to protect the payer's account and her transaction privacy, while the software and the vendor are considered to be less trusted, each with different capabilities. However, we do *not* assume they *collude*.

Curious vendor. More specifically, we consider that the vendor is curious but honest. He can save a photo of the screen of payer's phone, during the QR code scan, by reprogramming[2] or customizing his scanner. However, we do not assume that he can acquire the secret within the payer's phone, including the payment secret and other information for generating the protection for the screen (Section 4). We neither assume he can get or set the internal status of the POS (scanner is outside the POS and is controlled by the vendor). In the meantime, we assume that he does not attempt to directly steal money from the payer, an attempt that can be detected by the provider (e.g., overcharging the payer for the purchased item).

Instead, the malicious intent of the vendor is to benefit from tracking or helping a third party track the payer's purchase activities. It is undoubted that purchasing activity logs are valuable to

[2]Some scanner manufacturers provide programming guide to help vendor modify the logic of the scanner[3].

merchants who are eager to know what their customers bought, returned and what they might be interested. Starbucks, a well known Apple friendly brand, hesitated to support Apple pay before 2016 because Apple pay is too anonymous [1]. On the other hand, we assume the input to the tracking technique here is limited to the screen image containing QR code (e.g., we assume the vendor does not take a picture of buyer and use it for tracking).

Note that there are situations where the payment is done overtly, without hiding the payer's identity. Prominent examples include purchasing through membership card. In these cases, the identity information proactively provided by the payer enables the vendor to not only link her transactions together across different stores within the same organization, but also connect her behaviors **across organizations**, should the vendor decide to sell the identity with purchase information to a third party like the advertiser, which results in unexpected privacy leakage (imagine a restaurant learns what you'd like to order according to your Walmart's history) . However, the off-line scheme like the QR pay is designed to provide the payer *option* to remain anonymous.

Untrusted OS. The OS in our model is untrusted, which can be completely under the control of the adversary who not only can get access to the secret of the victim's wallet but can also learn other information, for example, the secrets for creating screen masks (Section 4). Further, the adversary also tries to acquire the fingerprint of the device, though she cannot directly observe it from the software of the device.

As mentioned before, the QR code payments like the one used by Paypal do not have a secure container for the payment secret in the phone, and only ask the OS to protect the secret, so when the OS is compromised, everything is exposed to the adversary, except the hardware features of the screen that cannot be directly "seen" from the software stack of the system. With the wallet secret, the adversary can generate a valid token and QR code (Section 2.2), by using the same algorithm (which we assume is public) and time information, for off-line payment.

Our assumption of untrusted OS is practical for the mobile payment scenario. Not all Android phone manufactures are prompt in pushing updates to their users, resulting in a fragmented Android eco-system with a lot of vulnerable devices open to attackers. Recent attacks also showed that attackers can easily get the token of the payment [6] with the help of vulnerable OS components.

3 SCREEN FINGERPRINTING

Our research shows that unique physical features of smartphone screens can be used to identify them and defend against adversary who has even obtained the payment secret. Extraction of these features, however, is nontrivial, due to the impact of projective distortions introduced when the screen is scanned by the camera. In this section, we present a technique that addresses the issue and fingerprints a screen using the image that can be conveniently taken during a payment scan.

3.1 Overview

Firstly, we briefly show how the screen fingerprint protects users from secret key leakage if only the device is not physically acquired by the attacker. At the registration phase, a photo of the screen should be securely uploaded to the server, with which the server can extract a piece of fingerprint. Every time a transaction happens, the server not only verifies if the QR code token is correct, but also compares the fingerprint from the scanned photo against the one collected at the registration phase. The transaction can only be approved if the QR code and the fingerprint are both correct.

Our scheme does not protect users whose OSes are compromised before registration. But if the registration phase is safe, even when the OS is compromised *thereafter*, the attacker will fail to launch the attack. When the attacker controls OS, she can acquire the secret key for QR code generation with the acquired privilege. However, there is no way for her to acquire the authentic screen fingerprint, so the check of fingerprint comparison will fail and the server will be notified.

3.2 Photo Extraction and Correction

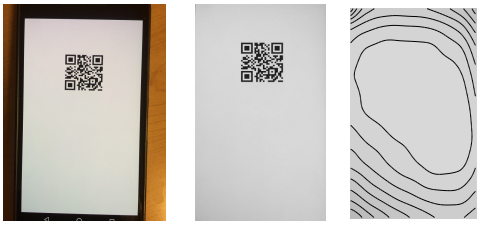

Figure 4: Fingerprint visualization.

Figure 4 visualizes the luminance fingerprint of a screen, which is illustrated through distortion correction, blurring and contour drawing. This fingerprint can be picked up by the POS camera when it is positioned at a right distance and angle toward the screen. What is expected is that the screen is in parallel with the camera lens, with its center right under the camera. In practice, hardly can this be done perfectly by a payer, as illustrated in Figure 2, even when a POS system is calibrated to identify the QR code shown on the screen (with the code boxed by a square visualized to the party who scans). As a result, the images snapped by the POS camera are often distorted, which brings in the trouble to effective extraction and comparison of fingerprints (distortion may make one phone similar to another). Following we elaborate how this challenge is addressed in our research through proper corrections.

Distortion correction. The picture of a screen taken by POS can be viewed as a projection from the screen plane to the camera perspective plane (as they are not parallel), with an unknown angle (see Figure 5). Most important to the distortion correction is to identify the angle. For this purpose, we come up with a new approach that leverages the *QR code on the screen as a reference* to realign the whole image through a projective transformation, which turns out to be very effective in recovering the distorted screen fingerprint.

Specifically, we consider that the phone screen sits on a display plane (D). When its picture is taken, the screen image is projected from D to a shooting plane (S), which is distorted. Our goal is to project the image from S to a correction plane C parallel to D, through a projective transformation (T).

Solving a projective transformation that perfectly reverts the distortion requires the positions of at least 4 different points on S

Figure 5: Projective distortion illustration. The three images are considered to be on three planes (D, S and C) respectively.

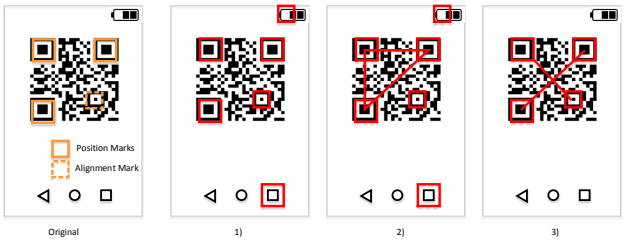

Figure 6: Bullseyes locating algorithm.

and their counterparts on C. It turns out the four "bullseyes" of QR code can be treated as the 4 markings. The "bullseyes" (see Figure 6) are used by the decoding program to locate the QR code, including three positioning markings to identify the direction of the code, and an alignment marking to help with orientation. Specifically, we first extract the coordinations of these "bullseyes" on S using standard QR decoding algorithm and then map them to the middle of a rectangle (0, 0) to (1800, 1080) of C, to obtain the transformation parameters.

The above approach works when the coordinates of all four "bullseyes" are correctly detected. However, these coordinations cannot be obtained when QR decoding algorithm fails. To extract fingerprint in this case (for multiple scan cases described later), the coordinations are inferred from the screen picture using the algorithm described below:

(1) Our approach extracts rectangles by identifying all quadrilaterals from the picture and removing those quadrilaterals unlikely to be foursquares. Tools like `regionprops` of Matlab Image Processing Toolbox can achieve this goal.
(2) Within the rectangles, our approach further selects the triples with their centroids forming an isosceles right triangle (approximately, due to the distortion). Oftentimes, this process will discover a triple that consists of the top-left, the top-right and the bottom-left bulleyes, as illustrated in Figure 6 middle.
(3) Finally, we look for a quadrilateral (the alignment mark) whose centroid is approximately on the mid-perpendicular of the right angle side of the isosceles we just found. The quadrilateral should be moderately far from the right angle vertex, which is shown in Figure 6 right.

Screen image recovery. Using the coordinates of the four bullseyes, we can perform projective transformation to project the screen image from the shooting plane S to the corrected plane C with distortion eliminated. In essence, all the pixels within the image on S are aligned to C while their relative location to the four markings are preserved. This process can be done using the standard projective transformation solving method [5].

3.3 Fingerprint Extraction & Comparison

Fingerprint extraction. To get a clean fingerprint from the transformed picture, we firstly applied noise cancellation techniques. We applied a radius 2 gaussian filter to the photo to remove the high frequency noise. Then we extract the luminance level after grayscaling the photo.

After that, a fingerprint of a screen is generated as a matrix of the *luminance* for each pixel (0 to 255). To visualize the screen features, we use contours to connect the pixels with similar luminance levels and then run Gaussian Blurs to smooth out the images. The right image in Figure 4 is the fingerprint for the screen image on left.

Fingerprint comparison. Once the a fingerprint it extracted, we need to compare it with other fingerprints to determine if the screen has been seen before. For this purpose, we utilize the Pearson product-moment correlation coefficient to measure the similarity of two fingerprints. The correlation coefficient is chosen since it measures the distance between two vectors based upon the similarity of the *relations* among the elements within each individual vector. This allows us to overcome the differences in the absolute brightness values for the same pixel observed under various light conditions. Specifically, given two fingerprints, which are two matrices with each element being either the brightness level of a pixel (100 to 255) or of 0 if the value of the element is removed during the distortion correction (border or virtual button) phase, we first convert these matrices into two vectors v and v', through concatenating their rows together, sequentially. Then we remove the elements from individual positions within v and v' if at least one of them in that position is 0. After that, our approach compares v and v' by calculating the Pearson product-moment correlation coefficient. The result needs to be adjusted by the difference in the numbers of "effective elements" in these two vectors, that is, those not being removed (non-zero in value). This is important because the difference in vector lengths should also be taken into consideration in the similarity measurement, but our previous steps dropped this information. Let d be such a difference between the two matrices and e be the total number of non-zero elements copied to vectors. The *similarity* between two fingerprints, Δ, is calculated as follows:

$$\Delta = (1 - \frac{d}{e})CorrCoef(v, v')$$

The similarity value is then compared to a threshold to determine whether the two fingerprints belong to the same phone.

4 ANONYMOUS SCREEN AUTHENTICATION

While our fingerprinting technique could protect the user against adversary who attempts to steal payment token and spares it with another device, it also enables unwanted user tracking. In this section, we describe our design of AnonPrint, which obfuscates a screen with a digital brightness mask to prevent the linking across payment transactions, while still enabling an authorized party to authenticate the owner of the screen.

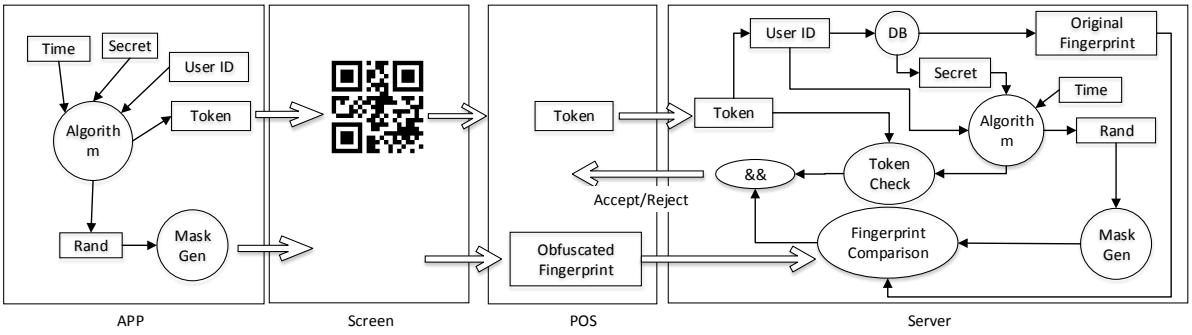

Figure 7: Overview of AnonPrint.

4.1 Overview

Figure 7 illustrates the framework of AnonPrint and the payment process that supports this anonymous screen authentication. The framework is built upon the existing off-line payment system, with only moderate changes made to the wallet app, POS scanner and the service on the payment provider side. More specifically, the payer first needs to submit the original screen fingerprint of her device to the payment provider when she opens an account. The wallet app is modified to synchronize a secret random seed with the provider, which could be achieved through hashing the time for the payment (as encoded in the QR code) together with a shared secret using a cryptographic hash function (e.g., SHA-256). This seed further bootstraps a pseudo random number generator (PRNG) each time when the wallet app needs to provide each party a sequence of random numbers for mask generation. This mask is displayed as the background for displaying the QR payment token, from which the POS scanner extracts the obfuscated screen fingerprint in addition to decoding the QR code and passes the information to the payment provider. Upon receiving the fingerprint and the payer's token, the provider retrieves the shared secret and the original screen fingerprint using her claimed ID. Then, the same mask used by the payer is re-constructed and together with the original fingerprint, serving as inputs for synthesizing a new obfuscated fingerprint. This fingerprint is compared with the one from the payer's screen and the transaction can be approved when their similarity is above a threshold and other security checks are made. Following we elaborate how the screen is masked, how the fingerprints are compared and our analysis of the security properties.

4.2 Screen Obfuscation

Our idea to obfuscate a screen is to create a digital luminance pattern, called *mask*, to hide the screen's hardware fingerprint for each payment transaction. Such a mask is automatically generated by a digital wallet app, based upon a PRNG seeded with a random number synchronized with the payment service provider, so the provider can also generate the same mask to authenticate the payer. The mask needs to be realistic, similar to a real fingerprint in terms of the distribution of brightness levels. Further it should work on not only a physical screen but also its image, since the latter is all the provider has about the payer.

Mask generation. A screen fingerprint is characterized by a smooth luminance change observed across neighboring pixels: the luminous intensities of these pixels, when their inputs have all been set to the maximum (255) [3], can be slightly different, due to their physical features; such a difference is minor between the pixels close to each other, since any large, abrupt change is likely to disqualify the whole product. To obfuscate this hardware fingerprint but maintain a screen's realistic looking, it is important that the mask generated will fully preserve this property. Here is how we do that in our research:

(1) *Random zone selection:* Our approach first produces a 180*108 pure white (with all pixels set to 255) image as the background and randomly selects from the image 20 mutually disjoint zones, each with a size of 16*16.
(2) *Dot darkening:* From each zone, we randomly choose 3 pixels and set their pixel value to a random number between 0 to 100.
(3) *Smoothing:* For each zone, AnonPrint *blurs* it using *Gaussian Smoothing*, an image processing technique that, intuitively, "smoothes out" the dark color of the selected pixels to its neighboring pixels. In particular, their values are elevated by proportionally reducing the values of their neighboring pixels, based upon their distances to these pixels, according to the Gaussian distribution. The radius of the Gaussian smooth filter is set to 10 in our implementation.
(4) *Resizing:* The mask image is resized and scaled to a 1800*1080 matrix whose values range from 220 to 255. The size is identical to the original fingerprint, so that it can mask the whole area to create fingerprint. Besides, the resizing also implies smoothing.

Mask registration. To apply the technique to the QR-code payment, each user first need to register to the payment provider with an image of her unprotected screen when all pixels are set to the maximum gray-scale. This can be done by taking a close-in picture of the screen using a second device (another phone or a laptop), or pay without mask at a trusted vendor. The secret used by device's PRNG is also transmitted to they payment service in this stage.

[3] All payment apps we surveyed change the luminance level to maximum when the QR payment UI is displayed, to increase the success rate of QR decoding.

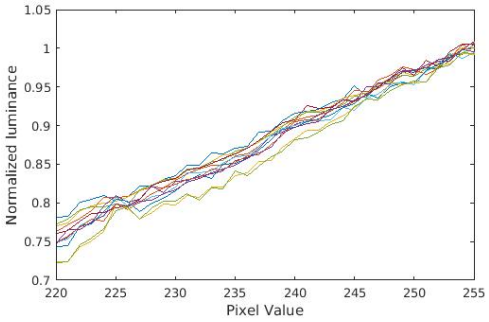

Figure 8: The brightness attenuation model.

4.3 AnonPrint Verification

During the payment, an image of a masked screen is taken to authenticate the payer. This is done on the payment service provider's side by reconstructing the mask using the shared secret, and then obfuscate the fingerprint for comparing with the image from the vendor. Most challenging here is how to build the obfuscated screen fingerprint, which, due to the vendor's lack of access to the physical device, can only be *synthesized* from the original fingerprint and the mask.

Screen attenuation model. To synthesize an obfuscated fingerprint, we need to understand how a pixel's brightness is attenuated when displaying the content with various pixel gray-scale value. For this purpose, we selected 10 areas from a screen, each of 10*10 in size, and continuously reduced the value of each pixel in the area in a controlled environment, by one at a time, from 255 to 220. Each time we took a picture of the screen and measured each pixel's observed luminance value. Figure 8 illustrates the normalized luminance levels (in terms of pixel value in photo) observed from the images vs. the pixel values we digitally set for the screen.

As we can see here, the observed brightness of a pixel attenuates approximately linearly with its decreased pixel value (0 for totally dark and 255 for pure white), which can be modeled by the attenuation function: $y = 1 - k(255 - x)/255$, where k is the contrast factor, x is the input pixel value and y is the normalized luminance level observed from the photo. A challenge here is how to determine k, which actually varies for different phones and even different shots of the same screen, due to the contrast level set for individual phones and most importantly the scanning camera's automatic adjustment of its ISO sensitivity according to the ambient light. Following we describe how to synthesize an obfuscated fingerprint using the model, particularly how to find k.

Fingerprint synthesis. Our attenuation experiment shows that there is a linear relation between observed luminance intensity and pixel gray-scale value. This allows the payment service provider to synthesize the obfuscated fingerprint using the original screen's fingerprint and the reconstructed mask. Let the mask matrix be X and the original fingerprint matrix be f. The synthesized fingerprint f' can be calculated by $f' = f \circ (1 - k(255 - X)/255)$. The question here is how to determine k, which varies across different scans.

We calculate k in our research based on the observation that the fingerprint synthesized using a right k should have a higher similarity with the fingerprint f_o extracted from an authorized screen, compared with that based upon an incorrect k. So, finding k can be modeled as the following optimization problem:

$$k = \arg\max_{\kappa \in D} \ \Delta(f \circ (1 - \kappa(255 - X)/255), f_o)$$
$$s = \Delta(f \circ (1 - k(255 - X)/255), f_o)$$

Here Δ is the correlation coefficient mentioned in Section.3 and D is the range of k. To find out D, we recorded the optimized k using a large range $[0, 10]$ over 100 images taken from 50 phones in common environment. The experiment shows that under normal ambient light situations (as under the POS machine), k falls between 0.25 and 6. Further given an image of an authorized screen, a mask the screen wears and its original fingerprint, the similarity s between a synthesized obfuscated fingerprint and that extracted one from the image is found to be approximately unimodal across possible k values: i.e., increase first and then decrease. Therefore, the provider can run a ternary search to find the k within $[0.25, 6]$ that maximizes the similarity s. If s goes above the threshold (Section 5), the device is authenticated, otherwise, it fails.

In a rare case when a legitimate screen fails to be recognized by the provider, service provider can ask the payer to enter a strong password (directly into the POS system) for the authentication.

Multi-scan. A single scan on today's POS system typically yields a low recognition rate, typically below 50%. What happens in real world is multi-round, consecutive scans until a frame can be picked up by the decoder, and the message decoded from QR code passes the error detecting code checks. This property is used in our design of AnonPrint, for the purpose of achieving a high security guarantee for the second-factor authentication without undermining the utility of the technique. To explicate the approach, we first need to define False Rejection Rate (FRR) and False Accept Rate (FAR), as follows:

$$FAR = \frac{number\ of\ accepted\ unauthorized\ fingerprints}{number\ of\ attack\ attempts}$$
$$FRR = \frac{number\ of\ failed\ authorized\ fingerprints}{number\ of\ authentication\ attempts}$$

Given an FRR of α and an FAR of $(1 - \beta)$ for a single scan (i.e., showing a single mask), after N independent rounds, with each round a different mask shown on the screen, we are expecting a final FRR of α^N (that is, rejecting a fingerprint when all N rounds fail), and FAR of $(1 - \beta^N)$ (that is, accepting a fingerprint when at least one round succeeds). AnonPrint can be designed to utilize such a multi-round scan to strike a balance between the overall FAR and FRR. Specifically, during a QR scan, the wallet application continuously generates N different masks and display them as a background for the same QR code one after another, each lasting a fixed time interval. To help the scanner differentiate these masks, each mask is labeled by a mark — a small dark square with 10*10 pixels — displayed at a given position on the screen. The position of the square signals the scanner that a different mask is in use. In this way, totally N images will be sent to the payment provider, which accepts the transaction if at least one of them is considered to match the payer's screen fingerprint.

5 EVALUATION

In this section, we report our evaluation against AnonPrint, on the unlinkability it can achieve and its accuracy in distinguishing different devices.

5.1 Experiment Settings

Data collection. Due to the absence of the programmable POS scanner and source code of the off-the-shelf wallet app, we cannot implement and evaluate the whole payment process involving AnonPrint. Instead, we focus on understanding the security properties of obfuscated screen fingerprints. For this purpose, we collected a set of smartphones from a group of volunteers for our experiments, which come from major phone manufacturers, as shown in Table. 1 (To protect their privacy, detailed information about their devices were not recorded.). All 100 the phones are used to understand the effectiveness of the screen fingerprint in identifying devices (see Section 5.2). 50 of them are used to evaluate the anonymity protection (4 are removed later because of stains or cracks on screen) and the effectiveness of AnonPrint (4 are removed accordingly) separately. During the experiments, we use an iPhone 6s to capture images for screen fingerprinting. Also, we implemented an Android application that displays QR code and obfuscates a screen using masks derived from given random numbers for anonymous payment (for iPhones, images are generated by a computer and sent to phone for display). To collect the fingerprints from each device, we first display on each screen a QR code without obfuscation, and then continue to show 5 different masks on the screen with the same code. Each time, we take a picture from the screen and use the image to extract fingerprints, masked or not.

Table 1: Statistics of the Brand of Participants' phones.

Brand	iPhone	Samsung	Others (Huawei, Oppo, etc)	Total
Quantity	44	17	39	100

Computing platform. We use a desktop computer to simulate the POS-side fingerprint extractor and provider-side fingerprint authentication server. The desktop runs on a machine with Intel Core i5 3.2GHz CPU, 12 GB memory and uses Ubuntu 14.04 operating system. The analysis routine (mainly image processing and comparison) is developed using Matlab 2015b. We also implement the obfuscation mechanism on a Nexus 6 phone, with 3 GB memory.

5.2 Fingerprint Accuracy

We firstly evaluated if screen fingerprint can be used to differentiate devices. Figure 9 illustrates the distributions of the fingerprint similarity, Δ, over 40 images taken from 20 phone screens, two images per screen. The blue solid line represents the similarity between the images of the same phone, and the red dots are for those between two different phones. As we can see from the figure, rarely does the cross-phone similarity go above 0.9, while self-similarity stays above 0.95 most of the time. So we choose 0.90 to 0.95 as the threshold range for determining whether two fingerprints indeed belong to the same phone.

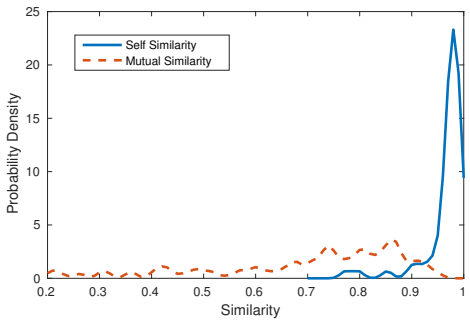

Figure 9: PDF of fingerprint similarity.

To validate the threshold, we collected 160 images from 80 additional phones (still two images per phone) for testing. For the images, again we calculate the similarity and the self-similarity for each of them, and further link two images together if their similarity goes above the threshold th, in an attempt to understand the feasibility of using the fingerprints to authenticate or track devices. This result is then compared with the ground truth (i.e., whether two images indeed come from the same phone), to identify the percentage of the images that have been correctly linked to their counterparts, which we call *coverage*. In the meantime, we also inspect each image to find out how many other images it is similar to (with a similarity above the threshold).

The study shows that, when th is set to be 0.93, the coverage reaches 96.3%, but on average an image is linked to 2.195 images, indicating that at least one another device is confused with each phone, on average. We can further raise th to 0.945. In this case, the coverage descends to 88.75% while incorrect links per image drop to 0.070.

From the service provider's perspective, with a 0.93 th, 96.3% of the time, authentic users can pass the authentication correctly while one 1 out of 160 adversarial trails may succeed. The rate can be further reduced to negligible 0.07% with 7.5% more failed normal cases. From the vendors' perspective, they can correlate 96.3% of their customers' transactions together, though may also incorrectly bring another customer's transaction to one's profile. This could still be acceptable if the vendors use the linked transactions for purchasing preference mining and targeted advertisements. This indicates that for 88.75% of transactions, the vendors can accurately identify other transactions from the same customer, by simply looking at the features of her screens.

5.3 Anonymity Protection

We then evaluate the protection provided under the masked scenario. To understand the anonymity achieved by AnonPrint, we conducted an experiment to evaluate how likely a curious adversary (i.e., the vendor) can correctly link two obfuscated fingerprints together. Specifically, we collected 2 masked screens from each of the 46 phones and analyzed the similarities of their obfuscated fingerprints under the same phone and across different phones.

Effectiveness of obfuscation. Like the experiments in Figure 9, we compute the pairwise similarity of the two obfuscated fingerprints for each phone, as well as that for different phones. Compared

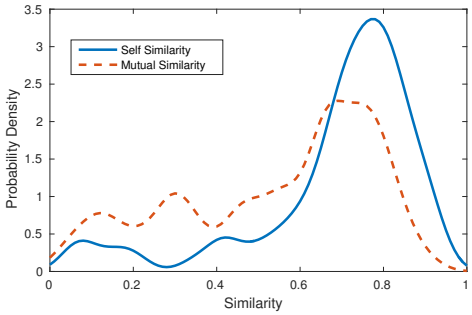

Figure 10: PDF of fingerprint similarity (after masking).

with that of unprotected screens (Figuree 9), the distribution of the self-similarity here has changed significantly, moving towards that of the mutual similarity (for two unrelated phones). It is clear from the figure that the chances of finding two similar fingerprints from the same phone become much lower. Meanwhile, compared with the distribution in Figure 9, the distribution of the mutual similarity is also slightly affected by the obfuscation, moving towards that of the self-similarity. As a result, these distributions become less distinguishable, which makes the customer tracking difficult: the vendor is left with the option of either choosing a large th to trace a very small number of users with reasonable accuracy or a smaller th to trace more users with a much higher error rate. Regardless of the choice, the anonymity for a single user is significantly elevated.

The anonymity protection of Anonprint can be quantitatively measured by the ratio of the common part of the two similarity curves. The common part occupies 69.4% of the two curves with the protection of Anonprint (Figure. 10). In contrast, the common part is only 15.4% without the protection of Anonprint (Figure. 9). We believe such large overlap (over 50%) makes the fingerprint useless for tracking. Therefore, we conclude that Anonprint indeed breaks vendors' capability of linking screen fingerprint.

5.4 Fingerprint Verification

The enhancement of anonymity comes with the cost of a decrease in verification accuracy, causing a legitimate payment request more likely to be rejected by the provider or an unauthorized payment request a little more likely to be accepted. To understand such impacts, we analyzed the verification accuracy when masks are applied.

Experiment setup. In our experiment, we first use the 46 phones to act as the payers to show their QR codes to our scanner (simulating fingerprint registration). The registered fingerprints in this step are denoted by F_{Reg}. Then, all volunteers present the QR code with the background obfuscated by a random mask (simulating the authentication step during the payment). All 46 fingerprints collected at this step are denoted by F_{Auth}. Next, each volunteer loads the QR code with the masks from other volunteers for 4 times, and we collected all 164 (invalid 20 excluded) such fingerprints (denoted by F_{Imp}). This simulates the scenario when the adversary has controlled the victim's phone and acquired all payment secrets to generate payment screens (correct QR code and correct masks). After that, all registered fingerprints F_{Reg} are used together with the reconstructed masks to synthesize obfuscated fingerprints (F_{Syn}), which are compared to their corresponding fingerprints F_{Auth}, in order to compute the self similarity. The fingerprints F_{Imp} are compared to the corresponding F_{Syn} (from the genuine mask owner instead of the attacker) to compute the mutual similarity.

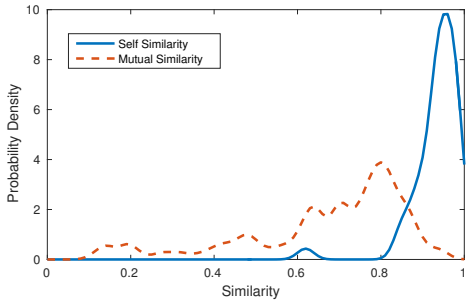

Figure 11: PDFs of similarity in fingerprint verification.

Experiment result. Figure 11 shows the PDFs for both the self similarity and the mutual similarity. Ideally, with the original fingerprints F_{Reg} in the database and the same masks shown by the wallet app, the server could totally remove the effects from obfuscation to achieve the distributions in Figure 9. Actually, the distributions shown in Figure 11 indeed come close to these in Figure 9, except the distribution for mutual similarity moving up and left towards that of the self similarity, which moves down a bit. The problem is caused by the obfuscation and de-obfuscation processes, steps that inevitably bring in noise. As a result, the verification on obfuscated fingerprints becomes less accurate than that on the unprotected ones.

Nevertheless, Table 2 shows that a good balance between the FRR and FAR can still be found. Specifically, under different thresholds as shown in the first column of the table, we present both FRR and FAR. As we can see from the table, even though the FRR and FAR observed from a single scan can be less ideal (e.g., 8.70% and 4.88% at the threshold 0.88), with an automated 2-round scan (with *the same mask*), AnonPrint can achieve a FRR of 0.76% or a FAR of 1.22%, which provides the user choices between the convenience (low FRR) and high protection (low FAR). In practice, a recommendation could be a sufficiently small FRR and a reasonable FAR to enable daily operations (mostly by legitimate payers) to go smoothly and in the meantime raise the bar significantly high for an attack should it happen. For example, under a threshold of 0.91, a 3-round scan achieves an FRR of 1.03% and an FAR of 5.39%: that is, a legitimate user may get falsely rejected (and therefore has to enter a strong password) once in almost a hundred times, while for the adversary, even when he manages to steal the victim's *all* secrets (except the screen fingerprint), he could only succeed once in about 19 times (assuming that the payment service provider immediately notifies the legitimate user each time a transaction fails for a given mask and the strong password is not received). Note that such an FRR/FAR balance is in line with that achieved by biometric-based second authentication factor. For example, keystroke pressure has an over 14% FRR and FAR [24]. Note that our approach is designed to *hide* the authentication secret (the screen fingerprint) through masking,

Table 2: FRR/FAR in different threshold and bootstrap count.

th \ N	1	2	3
0.88	8.7%/4.88%	0.76%/9.52%	0.07%/13.94%
0.89	15.22%/3.66%	2.32%/7.19%	0.35%/10.58%
0.90	19.57%/3.05%	3.83%/6.01%	0.75%/8.87%
0.91	21.74%/1.83%	4.73%/3.63%	1.03%/5.39%
0.94	45.64%/0.61%	20.84%/1.22%	9.51%/1.82%

unlike biometric second factors (facial features, fingerprints, typing patterns, etc.), which are often openly disseminated.

5.5 Performance

We further evaluate the performance impacts of our technique, through measuring the time consumption (averaged from 10 tests) introduced at different steps, both on the client (the phone) and the server.

Registration time consumption. A user needs to upload a image containing his original fingerprint to the server, when he wants to add a new device to the service. After having logged in, the user needs around 30 seconds to launch the application, take a photo for the new device and upload it to server, which is totally acceptable and negligible. Also, we believe the registration step happens rarely, because a user won't frequently change his devices used for payment, which is concluded and supported by our survey showing that people usually stick to one device for more than a year. Specifically, users change their phones with two main reasons: the phone is outdated; the phone is broken. For the first one, the evolution of hardware has been slowed in recent years, so our surveyed users, in average, change their phones with this reason for more than 1.6 years. For the second one, accidents breaking their phones happen in average every one year, which is not frequent.

Mask generation. Our study shows that mask generation on Nexus 6 takes 0.05 seconds on average. We compare it to the delay of QR code displaying using WeChat, a leading wallet app, to understand its impact. Specifically, to measure the delay of Wechat, we run a phone with a 29Hz camera to record the whole QR generation process in the app, from the moment when the button is clicked until the code is displayed. The recording shows that the app takes around 276ms (8/29s) to generate and show the QR code. So, we conclude that the overhead introduced by AnonPrint (only 50ms) is small for the off-line payment.

Fingerprint extraction. After a screen image is taken, the fingerprint is extracted and sent to the payment service provider through the POS network. Such an image is of the size of 80-100 KB in the JPEG format. Our desktop takes 0.64 seconds to extract the fingerprint from it.

Fingerprint verification. Our study finds that this step takes 2.4 seconds on average and the overhead is incurred by the ternary search for the correct contrast factor k when matching the fingerprints. The overhead is noticeable but not significant, given the fact that just acquisition of QR code from a screen typically takes 1 second, not to mention the further delays in communication and processing on the payment provider side, in an off-line payment transaction. The performance of this search step could be improved by calculating the left and right forks during the ternary search in parallel, which could reduce the processing time by half.

6 DISCUSSION

Environment Variance. One limitation of our work is that the fingerprint extraction process may be interfered by strong ambient light. The precision of the fingerprint extraction scheme might be impacted under such circumstances. However, the QR code scanning usually happens in controlled and dim environment (e.g., inside a store), to facilitate rapid QR code recognition. Therefore, the extracted fingerprint is usually free from glare. To guarantee dim environment, some scanner manufactures designed their scanners with lens facing up, as shown in Figure. 12. In this way, when being scanned, the phone covers the whole scanner and makes the scanning environment nearly completely dark.

Figure 12: A scanner facing up.

Aging Problem. The fingerprint of a screen may differ along time because of hardware aging, which may fail users' legal transactions. This could be solved by updating the fingerprint periodically. Because nowadays users frequently make mobile transactions, the server may notice the shift and update the fingerprint thereby. Specifically, when the interval between consecutive transactions is small, the server could update the fingerprint if the fingerprint has a relatively large deviation but the transaction still passes the authentication.

7 RELATED WORKS

7.1 Hardware Fingerprint.

Besides screens, a lot of hardware modules on smartphones have their unique features enabling users tracking.

Radio frequency fingerprint. Arackaparambil *et al.* proposed a technique using the clock skews of the radio frequency modules of 802.11 networks to uniquely identify wireless network devices [7]. PARADIS was proposed to detect the source network interface card of an 802.11 frame using passive radio-frequency analysis. The work mainly exploits the imperfection of the transmitter [11]. Again, in 802.11 networks, {Machine, NIC Driver, OS} together are used to fingerprint a network device [15]. Besides Wi-Fi network device fingerprinting, there is also a generic RF device fingerprinting

scheme [27]. However, those methods cannot be directly applied to track the payer, because the off-line payment does not require a Wi-Fi network.

Sensor fingerprint. Accelerometers that are widely equipped on smartphones can be used to fingerprint the phones [16], because of the manufacture imperfection. Besides, smartphone speakers are different from one phone to another, which enables applications running on the phone to measure the characteristics of the sound played by the phone for fingerprinting [14]. The process can even be silent to people around [28]. However those hardware based fingerprinting schemes cannot be used to protect the payer from attackers who controlled the operating system, as those fingerprints can be easily measured by the OS and then sent to attackers. With those fingerprints, an attacker can manipulate the outputs of the sensor, making them appear to have a correct feature, as a result, bypassing the authentication.

7.2 Unconventional Tracking Method.

Besides hardware based tracking methods, there are a lot of unconventional tracking methods that trace device users silently without using hardware feature.

Web browser tracking. To track a web browser user, the attacker can check the availability of a specific font set, time zone, screen resolution etc, without the help of traditional cookie [10]. Besides, versions and configurations that appear in the HTTP request head can be used to fingerprint the user [17]. Nikiforakis *et al.* demonstrated that how over 800,000 users are fingerprintable even when they are using user-agent-spoofing extensions [22].

Miscellaneous tracking. Smartphones can be tracked simply through their personalized device configurations [19]. The applications installed on a phone can also be used to predict the trait of the phone user [25].

7.3 Biometric 2nd factor authentication.

A lot of biometrics can be used to authenticate users as assistance to password. Besides regular ones that have been widely deployed, like face, fingerprint and iris, there are also some implicit biometric authentication methods.

Keystroke dynamics. Different people typing even the same password have different patterns, which also can be used to authenticate users. Banerjee *et al.* and Teh *et al.* surveyed some works related to keystroke dynamics as authentication and identification [9, 26]. Keystroke dynamics cannot be used the in QR code payment scenario because payers do not need typing. Besides, according to the survey, the best scheme supporting both text and digits input (nowadays, most passwords require both types) on the mobile platform still have a 12.8% EER [13], even with a modified device: that is, an FRR and FAR both reaching 12.8%. On mobile platform, the error rates are even worse, as they can exceed 13% [12, 20]. This level of effectiveness is well below what we can do even on anonymized screen fingerprints.

Gait. Gafurov *et al.* proposed a framework to use the motion sensor on wearable devices to do gait authentication and identification[18]. Ngo *et al.* presented a large database containing 744 subjects, which enables them to optimize gait authentication algorithms considering different genders, ages ground conditions etc [21]. The gait biometrics however can hardly be collected in our scenario because the QR code payment is offline, in which case the service provider has no access to the inertial sensor output information.

8 CONCLUSION

In this paper, we present a new technique that enhances the security protection to popular QR-based payment, without undermining the payer's privacy. Our technique leverages the features of the payer's screen, which is characterized by its unique luminance unevenness introduced by the imperfect manufacture process, ensuring that even when the payer's digital wallet has been fully compromised, an unauthorized payment still cannot succeed. The screen features, however, would raise a privacy concern, were it naively deployed to authenticate the payer, since it could be abused by the vendors to link one's different purchases together. To address this concern, we present AnonPrint that obfuscates the phone screen during each payment transaction. This new approach defeats the attempt to correlate the payer's purchase activities and also enables the payment provider, who can reconstruct the obfuscation mask, to authenticate the payer. We evaluated its efficacy through experiments, which all demonstrate the promise of this new solution.

9 ACKNOWLEDGEMENT

We thank anonymous reviewers and our shepherd Amir Houmansadr for the suggestions. The work was sponsored by Shanghai Sailing Program. The IU author is supported in part by the NSF 1408874, 1527141, 1618493, 1801432 and ARO W911NF1610127. The CUHK authors are supported by National Natural Science Foundation of China (Grant No. 61572415), and the General Research Funds (Project No. 14208818) established under the University Grant Committee of the Hong Kong Special Administrative Region.

REFERENCES

[1] 2016. Apple Pay Is Too Anonymous for Some Retailers. http://www.nectarpartners.com.au/single-post/2014/10/20/Apple-Pay-Is-Too-Anonymous-for-Some-Retailers. [Online; accessed 19-Oct].
[2] 2016. A pos supporting QR code. http://jepower.en.made-in-china.com/product/fNnQmIEPmxkd/China-IC-Card-Reader-POS-Device-Support-Four-Means-of-E-Payment-Functions.html. [Online; accessed 19-Oct].
[3] 2016. Spec of a QR code Scanner. http://www.waspbarcode.com/~/media/pdfs/waspbarcode/products/scanner-pdfs/wdi4600-2d-barcode-scanner.ashx. [Online; accessed 19-Oct].
[4] 2017. 5.5 Trillion US Dollar Transactions over QR code in China. http://www.pymnts.com/news/payment-methods/2017/mobile-payments-hit-5-5-trillion-in-china/. [Online; accessed 7-Aug].
[5] 2018. Finding the Transform matrix from 4 projected points. https://math.stackexchange.com/questions/296794/finding-the-transform-matrix-from-4-projected-points-with-javascript. [Online; accessed 13-Jun].
[6] 2018. Your Mobile Money Could be Stolen By This Attack. https://freewechat.com/a/MzIzNzcyNzkzNQ==/2247491423/1. [Online; accessed 3-Sep].
[7] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. 2010. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the third ACM conference on Wireless network security*. ACM, 169–174.
[8] Xiaolong Bai, Zhe Zhou, XiaoFeng Wang, Zhou Li, Xianghang Mi, Nan Zhang, Tongxin Li, Shi-Min Hu, and Kehuan Zhang. 2017. Picking Up My Tab: Understanding and Mitigating Synchronized Token Lifting and Spending in Mobile Payment. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 593–608. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/bai
[9] Salil P Banerjee and Damon L Woodard. 2012. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research* 7, 1 (2012), 116–139.

[10] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. 2011. User tracking on the web via cross-browser fingerprinting. In *Nordic Conference on Secure IT Systems*. Springer, 31–46.
[11] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. 2008. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*. ACM, 116–127.
[12] P Campisi, E Maiorana, M Lo Bosco, and A Neri. 2009. User authentication using keystroke dynamics for cellular phones. *IET Signal Processing* 3, 4 (2009), 333–341.
[13] Nathan L Clarke and Steven M Furnell. 2007. Authenticating mobile phone users using keystroke analysis. *International journal of information security* 6, 1 (2007), 1–14.
[14] Anupam Das, Nikita Borisov, and Matthew Caesar. 2014. Do you hear what i hear?: fingerprinting smart devices through embedded acoustic components. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 441–452.
[15] Loh Chin Choong Desmond, Cho Chia Yuan, Tan Chung Pheng, and Ri Seng Lee. 2008. Identifying unique devices through wireless fingerprinting. In *Proceedings of the first ACM conference on Wireless network security*. ACM, 46–55.
[16] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable.. In *NDSS*.
[17] Peter Eckersley. 2010. How unique is your web browser?. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 1–18.
[18] Davrondzhon Gafurov, Einar Snekkenes, and Patrick Bours. 2007. Gait authentication and identification using wearable accelerometer sensor. In *Automatic Identification Advanced Technologies, 2007 IEEE Workshop on*. IEEE, 220–225.
[19] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix Freiling. 2016. Fingerprinting Mobile Devices Using Personalized Configurations. *Proceedings on Privacy Enhancing Technologies* 2016, 1 (2016), 4–19.
[20] Emanuele Maiorana, Patrizio Campisi, Noelia González-Carballo, and Alessandro Neri. 2011. Keystroke dynamics authentication for mobile phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 21–26.
[21] Thanh Trung Ngo, Yasushi Makihara, Hajime Nagahara, Yasuhiro Mukaigawa, and Yasushi Yagi. 2014. The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication. *Pattern Recognition* 47, 1 (2014), 228–237.
[22] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2013. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and privacy (SP), 2013 IEEE symposium on*. IEEE, 541–555.
[23] Fumihiko Saitoh. 1999. Boundary extraction of brightness unevenness on LCD display using genetic algorithm based on perceptive grouping factors. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, Vol. 2. IEEE, 308–312.
[24] Sougata Sen and Kartik Muralidharan. 2014. Putting âĂŸpressureâĂŹon mobile authentication. In *Mobile Computing and Ubiquitous Networking (ICMU), 2014 Seventh International Conference on*. IEEE, 56–61.
[25] Suranga Seneviratne, Aruna Seneviratne, Prasant Mohapatra, and Anirban Mahanti. 2014. Predicting user traits from a snapshot of apps installed on a smartphone. *ACM SIGMOBILE Mobile Computing and Communications Review* 18, 2 (2014), 1–8.
[26] Pin Shen Teh, Andrew Beng Jin Teoh, and Shigang Yue. 2013. A survey of keystroke dynamics biometrics. *The Scientific World Journal* 2013 (2013).
[27] Oktay Ureten and Nur Serinken. 2007. Wireless security through RF fingerprinting. *Canadian Journal of Electrical and Computer Engineering* 32, 1 (2007), 27–33.
[28] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. 2014. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 429–440.