Characterizing Active User-Device Physical Unclonable Functions (PUFs) based on Mobile Device Touchscreen

Ryan A. Scheel
Electrical & Computer Engineering
Iowa State University
Ames, IA, USA
scheel.ryan@gmail.com

Akhilesh Tyagi
Electrical & Computer Engineering
Iowa State University
Ames, IA, USA
tyagi@iastate.edu

ABSTRACT

Mobile systems have unique security requirements. An atomic unified signature of the user and the device is more useful than just a user password or only a device biometric. We propose a physical unclonable function (PUF) derived from the touch screen of a mobile device. We characterize such a unified PUF for both its variability and reproducibility. For the (same device, same user, different challenge), (same device, different user, same challenge), (different device, same user, same challenge), we benefit from as large a variability in the response as possible. On the other hand, for (same device, same user, same challenge), we want perfect reproducibility in the response. We show 60+ bits Hamming distance in the unified PUF responses of length 128 bits when variability is expected. We illustrate 0 bits of error in reproducibility scenarios through the use of an innovative statistical concentrator serving the role of ECC (error correcting codes) in traditional PUFs. We also demonstrate the promise of these PUFs to serve as biometric hardware pseudorandom number generators (PRGs) by putting them through Montreal TESTU01 suite of tests. Our best PUFs pass all the tests except occasionally failing 3. This PUF was implemented on Nexus 7 devices running Android.

Keywords

physical unclonable function (PUF), mobile device authentication

1. INTRODUCTION

Mobile devices are becoming the primary user interface terminals of the modern world with the computing servers infrastructure being pushed to the cloud. With wearable devices on the horizon, the number of mobile devices per capita is likely to explode.

Securing a mobile device is significantly more challenging. Physical possession allows for side-channel attacks. Authentication is more difficult since device connectivity is not always available due to mobility. Not being able to rely upon constant connectivity also reduces the options available to implement a mobile root-of-trust.

A physical unclonable function (PUF) [11] provides a device chip-set biometric which can be wrapped into various cryptographic protocols. A typical PUF is based on the inherent variability in the silicon fabrication process. The dopant distribution and lithographic etching are examples of physical phenomena that lead to individualized energy-delay profile for each transistor. The same transistor in two different chips, even when fabricated on the same wafer, exhibits its individuality. Such biometric individuality can be captured as a challenge-response function block. Typical PUF implementations use among others a cascaded ring of inverters where these delay variations are amplified. Challenges result in a sampling of a specific bit in this ring oscillator. Another schema sends a 0 down one path and 1 down another. At each stage, the two bits carry on in straight paths or switch over their paths based on a challenge bit. At the last stage, an arbiter determines whether 0 or 1 bit won the race, which also becomes one bit of the response.

These schemes do well in capturing unique physical attributes of the physical device. A mobile device however unlike a cloud server or a desktop client is too closely tied to a user. From secure services perspective, it is difficult to separate the user from the device. A device in the hands of Alice may be eligible for a different set of services than when in the hands of Bob. What we need as a PUF is something which generates a response from a challenge based on the entangled biometric of both the user and the device. Such entangled, user-device (UD) PUFs are the focus of this paper. Specifically, we establish an inviolable, atomic, combined user-device (UD-)identity. The inviolability comes from the physical layer of the silicon in a manner similar to biometrics. Atomicity comes from the atomic granularity of user and PUF composition.

Contributions:. **(1)** Establishing that an entangled PUF can be realized from the combined biometric of silicon and of a human user.
(2) Exploring the design space for a touch screen entangled PUF to capture design parameters that provide the requisite variability and reproducibility.
(3) Establishing the pseudo-random number generator properties of the proposed UD PUF. This provides the security

assurance that the UD PUF is not easily modeled.

Touch Screen as the User Device Entangler:. In today's mobile devices, many sensors exist, explicitly designed to interact with the user. If these sensors exhibit their own semiconductor unique identities, they constitute a good initial candidate set for UD PUF.

Touch screen is one such plausible PUF sensor. Capacitive or resistive touch screens use an array of electronics capable of measuring capacitive or resistive change induced by the touch. These CMOS transistors exhibit the same variability that has been exploited in the traditional silicon PUFs. In addition, there is sensing circuitry that detects the row and column number where the touch-induced capacitance or resistance change occurs. The sensing logic is replicated into a grid of touch screen regions so that the events in multiple regions can be detected in parallel. The size of this grid keeps increasing to support fine grained multiple gestures. This sensing logic is another source of variability in the touch screen based PUFs. It is plausible that the same silicon fabrication processes that lead to variability in the traditional silicon PUFs will induce similar variability in the measured current and voltages detecting touch in touch screens.

The challenge can be a simple line segment drawn on the screen. The user is asked to trace it. Android framework generates a sequence of Motion Event objects in response to such tracing. The Android framework includes a class `MotionEvent` (http://developer.android.com/reference/android/view/MotionEvent.html). This class, among many other methods, includes a method `final float getPressure()` which returns a value between 0 and 1. On a capacitive touch screen, it stands to reason that the device level variability must exist for the same reasons as for the classical silicon PUFs. The doping levels of capacitive permittivity layer will have the same statistical variation in the fabrication process. Similarly, the capacitance measurement circuits in these screens have the same statistical delay (or current level) variance as in silicon PUFs. The `getPressure()` method should reflect such per device variability in the pressure measurement.

The other desirable characteristic of pressure returned by `getPressure()` method is that it entangles the user behavior and the device biometrics in an atomic, inseparable manner. Part of the pressure reading is determined by the transistors and capacitors embedded within the touch screen. Part of the pressure reading is influenced by the user behavior, how s/he traces a path: how much physical pressure, how closely is the challenge line segment followed in the response. There is no non-trivial deterministic quantitative model to separate the user pressure contribution from the device pressure contribution.

Variability and Reproducibility of Touch Screen UD-PUF:. Any PUF is designed to give a unique, variable response to a unique challenge. The variability aspect quantifies how different the responses are over two different challenges - we could even specify a metric capturing some kind of distance within the challenge space. For a UD-PUF f, one aspect of its variability for instance is captured by the following equation $\sum_{i,j,k} HD(f(C_i, U_j, D_k),\ f(C'_i, U'_j, D'_k))$. $f(C_i, U_j, D_k)$ captures the UD-PUF f response on a challenge line segment C_i, traced by a user U_j, on device D_k. The function $HD()$ measures usual Hamming distance of two binary response strings. We expect *(different challenge, same user, same device)* scenarios $HD(f(C_i, U_j, D_k),$ $f(C'_i, U_j, D_k))$ to generate variability with nonzero response Hamming distances. We will similarly expect variability in *(same challenge, different user, same device)* $HD(f(C_i, U_j, D_k),$ $f(C_i, U'_j, D_k))$ and *(same challenge, same user, different device)* $HD(f(C_i, U_j, D_k),\ f(C_i, U_j, D'_k))$ scenarios. Question is how much variability is desirable? Should the average Hamming distance in $HD(f(C_i, U_j, D_k),\ f(C'_i, U_j, D_k))$ equation be half the response length N? We characterize this variability in this paper for touch screen based UD-PUF. An interesting, somewhat unexpected, lesson learned is the effect of quantization hysteresis on the variability.

For the *(same challenge, same user, same device)* $HD(f(C_i, U_j, D_k),\ f(C_i, U_j, D_k))$ scenarios, we expect reproducibility. In other words, the response Hamming distance should be 0. This is more difficult than it seems. In the classical silicon PUF implementations, the reproducibility violations occur when chip's operating conditions such as temperature are different than its characterization operating conditions. With UD-PUF this is further complicated by the fact that the human user behavior- path tracing, is rarely identically reproduced. We characterize the reproducibility error rate for the proposed UD-PUF. We then introduce a statistical concentrator ECC (error correcting concentrator like an error correcting code in coding theory) to generate close to perfect reproducibility.

Paper Organization:. Section 2 presents related work in the area. An outline of a conceptual framework to use the UD-PUF for device authentication and other secure services is given in Section 3. Section 4 presents our results on UD-PUF variability characterization. The reproducibility results are given in Section 5. The pseudo random number generator properties of UD-PUF are assessed in Section 6. Finally, Section 7 concludes the paper and Section 8 offers thoughts on the future work.

2. RELATED WORK

Physical Unclonable Functions (PUFs) are like biometric of silicon - unique to each individual chip so as to serve as an identity, with a random distribution. The delay of a fabricated silicon transistor is a random function despite an identical mask level geometry due to random distribution of dopants and lithographic diffraction. These statistical variations are large enough to differentiate chips from the same wafer. Many different kind of silicon PUFs have been proposed [15], [5], [6], [17], [7], [3], [11], [16]- ring oscillator, SRAM memory cell, arbiter, latch and flip-flop.

Such physical layer functions need both (1) per chip variability and (2) same chip reproducibility. The variability ensures that distinct devices produce different outputs on the same input. Reproducibility on the other hand is needed for predictability and determinism in the device authentication behavior.

These PUFs have been deployed in many applications [11], [3]. Their security has also been analyzed in many scenarios [8], [13].

A modern mobile device has an increasingly larger number of sensors - mostly to introduce new UI paradigms. It is not uncommon to have accelerometer, gyroscope, LCD touch screen, temperature and humidity sensor, barometer, RGB intensity sensor, proximity sensor, and gesture sensor. All these sensors are rooted in some kind of physical layer based on some analog activity. If these activities arise from some electrical phenomena based on capacitances, inductances and resistances, it is likely that the fabrication process has inherent randomized statistical distribution. Any of these activities or a combination thereof can constitute a viable atomic user-device identity.

Some work has started within the last year along this line of thought. All of them seem to still be aiming to establish a unique PUF like fingerprint for the device. A unified atomic user-device identity is not the goal. Dey et al. [4] use an accelerometer as the device fingerprint. The challenge is certain level signal applied to the vibrator, which activates the accelerometer to generate a response signature. Aysu et al. [1] use Micro Electro Mechanical Systems (MEMS) as a fingerprint PUF for low cost embedded solution in place of relatively more expensive ring oscillators or arbiters. More recently, Boneh's crypto group was in news [2] for developing fingerprinting techniques for mobile devices. There are no technical articles on this work yet. The San Francisco Chronicle talks about the challenge being flipping over a mobile device, and the measured response or fingerprint is the accelerometer activity. This could have elements of a combined user-device fingerprint - however, it is presented as a device fingerprint with the goal of masking out the user variance. Liu at al. [10] have a purported goal of creating a new UI by acknowledging and computing accelerometer activity driven gestures. Their hypothesis is that such gestures give many more degrees of freedom than the traditional gestures giving rise to richer user interfaces.

SenSec [19] builds a Markov process based predictive model from multiple motion sensor data such as accelerometer and gyroscope to classify it as a specific user. Napa et al. [14] use the unique finger placement biometric of a user to characterize a user on the basis of the placement of multiple fingers for multiple gestures on a touch screen. Zahid et al. [18] use the keystroke events time separation as a user biometric for authentication.

None of these research efforts have goals identical to ours - that of establishing an atomic user-device identity, and its applications.

3. MODEL, SCHEMA, AND MOBILE SECURE SERVICES

Model:. The UD-PUF consists of a characterization/profiling phase followed by the deployment phase.

During characterization, for a given user-device pair, a broad set of challenge-response pairs is collected - $CRSET(U_i, D_j) = \{(C_{i,j,k_0}, R_{i,j,k_0}), (C_{i,j,k_1}, R_{i,j,k_1}), \ldots, (C_{i,j,k_l}, R_{i,j,k_l})\}$.

Each challenge C_{i,j,k_m} can be a line segment, a series of points specifying several connected line segments, a seed point (x_i, y_i) and a curve generation schema such as Poincare function or fractal curves. The main expectation is that a challenge be a path long enough to generate a response of desired length (128-256 bits for AES keys, 1024-2048 bits for RSA keys). It may also be feasible to allow the user to trace their own repeatable, personalized path such as a signature.

The proposed UD-PUF layer converts a given challenge C_{i,j,k_l} into the corresponding response through a reproducible profiling phase - $R_{i,j,k_l} = UD - PUF_f(C_{i,j,k_l}, U_i, D_j)$.

Who should be holding the pre-characterized challenge-response set $CRSET(U_i, D_j)$? This entity necessarily serves as a root-of-trust for all the transactions based on this set. Based on the applications that need to be supported, a third party certification authority could be such an entity. Google Wallet like application can use such a scenario to verify the atomic user and device identity before providing services. The local root of trust within the device could be this entity to support UD-PUF based user-device unique signatures/passwords. Note that how to wrap the challenge-response set $CRSET(U_i, D_j)$ securely within such a root of trust, and the corresponding secure protocols for the communication between the cloud services layer, the mobile device, and a third party certification authority (CA) are not the focus of this paper.

The scenarios plausible for UD-PUF based authentication include the following:

1. Cloud based secure services maintain a user-device domain oriented access control layer. These domains may or may not have partial orders defining access control. A third party CA may be needed as a root of trust to hold the challenge-response set $CRSET(U_i, D_j)$. Note that it may be feasible to wrap this third party CA into the secure services provider.

2. Replace the classical password based authentication with a UD-PUF challenge response mechanism to access the device or device based apps.

3. UD-PUF could also be composed with the software based cryptographic primitives such as pseudo-random number generator, encryption and hash to make these primitives more robust.

4. UD-PUF VARIABILITY

We have conducted an evaluation of UD-PUF variability on 10 Nexus 7 devices with 3 distinct users. We wrote an Android app to collect the following data. The challenges are the paths drawn on the screen which are generated by a simple program that just uses Java libraries to randomly generate 4 (x, y) pairs which are connected into a convex path. The algorithm makes sure that all the (x, y) pairs are at least a minimum distance from each other. The paths are seeded with an incrementing count before each use to ensure easy reproducibility for each individual path. The LHS figure in Figures 1 & 2 shows the generated path in

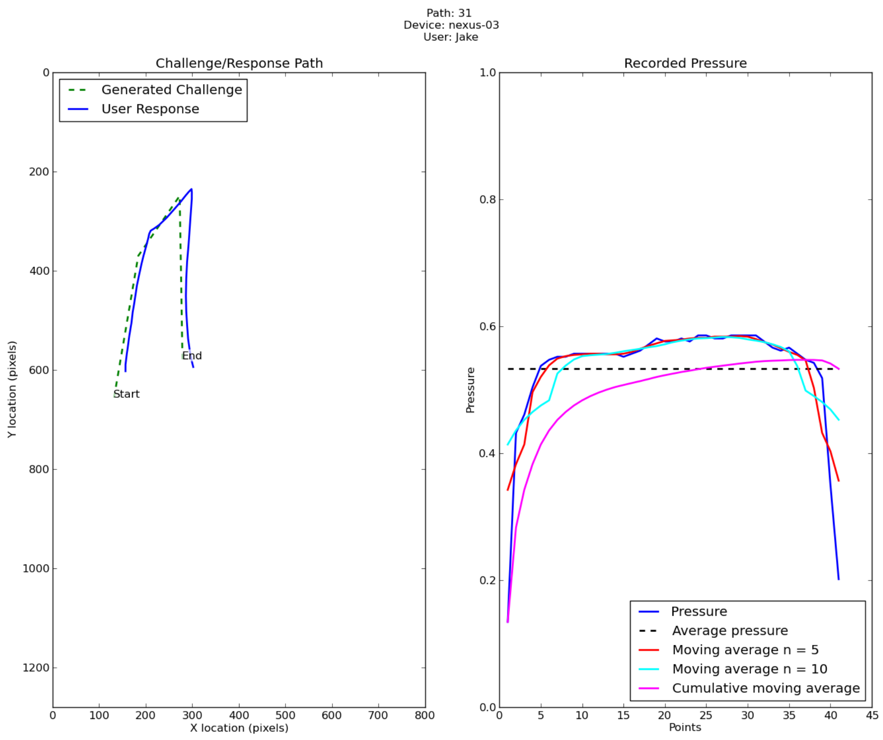

Figure 1: Path Challenges and Pressure Responses

dashed lines and the user traced path in solid lines for two different users and devices. Overall, for each of the 10 Nexus 7 devices, for each of the 3 users, 100 paths were drawn and traced for a reasonably sized data set.

In Android, a user traced path is returned as a sequence of class `MotionEvent` objects. Part of this class declaration looks as follows:

```
public final class MotionEvent
 extends InputEvent
 implements Parcelable

 final int getAction()
 final float getAxisValue(int axis, int pointerIndex)
 final float getPressure()
```

Without having to modify the Android framework, *getPressure()* seems like a good candidate for the source of a PUF, since it returns an analog float value. Each `MotionEvent` object carries a path sequence number, and its (x, y) coordinates among other information.

How to generate a challenge?:. The raw analog pressure values are between 0 and 1 as a float. We need to convert them into a binary sequence. Each path can result in anywhere from 150-400 `MotionEvent` objects being sampled. The Android events layer chooses a sampling frequency on the basis of its own loading. In Figures 1 & 2 RHS, solid blue line shows the raw pressure data which is a jagged graph.

Arbiter:. We need to convert this pressure value sequence into a binary response sequence. One obvious approach is to generate a reference pressure line. Any pressure above it is interpreted as a 1, and every thing below is interpreted as a 0. The choice of an arbiter reference line is what we call a quantizing mechanism. There are many possible approaches to quantizing this data.

average based arbiter: A simple approach is to take the average of all the pressure points. If a given path point's pressure exceeds this average, it is quantized as a binary 1, otherwise it is quantized as a binary 0. This has the advantage of resulting in responses that have roughly equal 0's and 1's - one of the required properties of a random string.

This quantization however has many runs of 0's and 1's since as seen in Figures 1 & 2 RHS, entire path segments are above or below the flat dashed blue pressure average line. Such 0 and 1 runs are bad if these responses are to be pseudorandom. As we discuss later, these strings fail 11 out of 26 PRG (pseudo-random generator) tests.

moving average $n = 5$ based arbiter: An alternate is to create a low-pass filter by keeping an n-size moving window. A running average in this n-window is main-

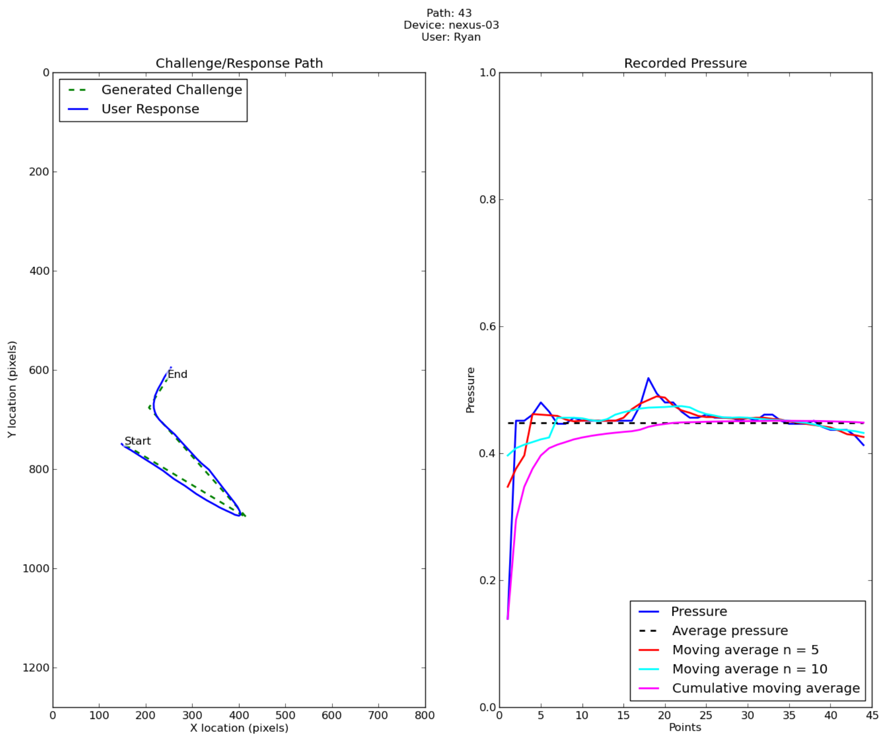

Figure 2: Another Path Challenge and Pressure Responses

tained. The quantization of the Point p_m is done based on the $n = 5$ moving average $avg5_m = (p_{m-1}+p_{m-2}+p_{m-3}+p_{m-4}+p_{m-5})/5$ where p_i is the pressure at Point i. If the pressure $p_m \geq avg5_m$ then it is quantized to 1, else it is quantized to 0. This arbiter passes almost all tests for PRG. In other words, a PRG based on this UD-PUF behaves like a cryptographic pseudo-random number generator. Note that this is a significantly stronger property than any of the silicon passive PUFs [15], [5], [6], [17], [7], [3], [11], [16] have been able to show. Most of the physical PUFs can best show a non-zero Hamming distance between two supposedly different responses. They have to use a cryptographic hash function such as SHA [12] on the PUF response to get any kind of pseudo-randomness. The red solid line in Figures 1 & 2 RHS shows the $n = 5$ quantization. Note that this line closely hugs the original raw data curve.

moving average $n = 10$ based arbiter: This quantization is similar to the preceding $n = 5$ quantization except that it uses the moving average of the preceding 10 points. This $n = 10$ moving average is shown as green/blue solid line in Figures 1 & 2 RHS. It does worse than $n = 5$ quantization on PRG tests.

cumulative moving average based arbiter: This is the extreme of the moving average with maximum dampening - the average of $p_0, p_1, \ldots, p_{m-1}$ is used to quantize p_m. This is shown as purple solid line in Figures 1 & 2 RHS. It has longer 0- and 1-runs than the $n = 5$ and $n = 10$ moving averages and hence does worst among the moving average quantization approaches on PRG tests. However, it still performs better than the flat average based quantization.

Variability in UD-PUF Responses:. Recall that the (same user, same device, different path), (different user, same device, same path), and (same user, different device, same path) should result in different responses. How much variability exists in these scenarios?

There are two ways to evaluate this variability. In one case, we can just look at Hamming distance between the responses. In another, we can try to assess if the response outputs behave like a pseudorandom sequence. The second case endows more robustness and more desirable cryptographic properties on these PUFs.

Hamming Distance between UD-PUF Responses:. We measured average Hamming distance between responses within the scenarios where variability is expected such as $\sum_{i,j,k} HD(f(C_i, U_j, D_k),\ f(C'_i, U'_j, D'_k))$. Table 1 shows our preliminary results when same path is used with different device/user scenarios. Due to symmetry of data, we have only shown an upper-triangular matrix. These are Hamming distance results when 128-bits responses are generated. Note that the average Hamming distance is close to 60 bits, almost half the string length. As a comparison, AEGIS arbiter PUFs [17] reported a Hamming distance of about 40 over

	D/U 0	D/U 1	D/U 2	D/U 3	D/U 4	D/U 5	D/U 6	D/U 7	D/U 8
D/U 0	0	61	61	67	64	67	63	66	67
D/U 1		0	65	56	63	60	63	64	62
D/U 2			0	58	65	58	60	59	61
D/U 3				0	62	59	64	61	60
D/U 4					0	63	63	64	65
D/U 5						0	65	60	56
D/U 6							0	64	68
D/U 7								0	62
D/U 8									0

Table 1: Hamming Distance, Same Path, Different Device/User, Flat Average Quantization. D/U denotes a Device/User Combination

D/U 0	D/U 1	D/U 2	D/U 3	D/U 4
63	59	56	41	61
D/U 5	D/U 6	D/U 7	D/U 8	
62	40	63	25	

Table 3: Average Hamming Distance over Different Paths for Several Device/User Combinations, Flat Average Quantization. D/U denotes a Device/User Combination

D/U 0	D/U 1	D/U 2	D/U 3	D/U 4
63	64	64	64	62
D/U 5	D/U 6	D/U 7	D/U 8	
63	64	63	63	

Table 4: Average Hamming Distance over Different Paths for Several Device/User Combinations, $n = 5$ Moving Average Quantization. D/U denotes a Device/User Combination

128-bit strings.

Table 2 reports the same data when quantization is performed with $n = 5$ moving average. Note that there are no appreciable differences between the flat average quantization of Table 1 and $n = 5$ moving average quantization of Table 2. In fact, $n = 10$ moving average and cumulative moving average yield similar data and hence we do not report them.

Similar data capturing average Hamming distance over multiple challenge paths for many user/device combinations ((user, device, *) space) is presented in Tables 3 and 4 for flat average quantization and $n = 5$ moving average quantization respectively. Note that the flat average quantization yields a lower variability for the different path, same device/user scenario than the $n = 5$ moving average quantization.

We show two of these different paths for the same user on different devices along with the corresponding pressure data in Figure 3. Note that the pressure differences exist along a significant fraction of the paths.

5. UD-PUF REPRODUCIBILITY

When (same challenge, same user, same device) combination is used, we expect perfectly reproducible response. Otherwise, a close enough response will be rejected in authentication for not being the expected fingerprint or password. In silicon PUFs, this issue arises since the profiling and PUF challenge generation may occur under different ambient conditions. Variable temperature affects transistor delays which in turn affects PUF's responses. In UD-PUF, user's interaction- tracing a path with some pressure, is inherently more variable than reproducible. How do we generate consistent responses despite the presence of a human user?

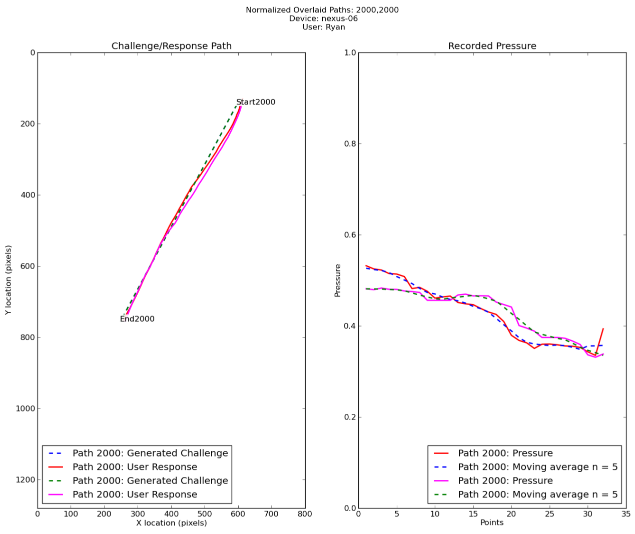

Figure 4: Challenge Path and Two Traced Paths result in Hamming Distance 14 on 40-bit Responses

We rely on statistical error correction for this. Just like in error correcting codes (ECC) in coding theory, a large sphere of codes around a valid codeword is not assigned to code words, we leave unused space - collection of paths around each valid response. If a void code word is transmitted, ECC force maps it to the valid code word closest to it. If a corrupted response occurs, we map it statistically into its ”center-of-gravity” response. The downside of this approach is that the valid challenge/response space is reduced.

MotionEvent Point Alignment:. The goal is to ensure that two separate traced paths for the same challenge by the same

	D/U 0	Dev/User 1	D/U 2	D/U 3	D/U 4	D/U 5	D/U 6	D/U 7	D/U 8
D/U 0	0	63	63	62	63	67	65	63	64
D/U 1		0	63	63	63	64	63	63	63
D/U 2			0	63	62	64	63	62	64
D/U 3				0	63	65	64	61	64
D/U 4					0	63	65	64	63
D/U 5						0	65	63	62
D/U 6							0	65	63
D/U 7								0	64
D/U 8									0

Table 2: Hamming Distance, Same Path, Different Device/User, $n = 5$ Moving Average Quantization. D/U denotes a Device/User Combination

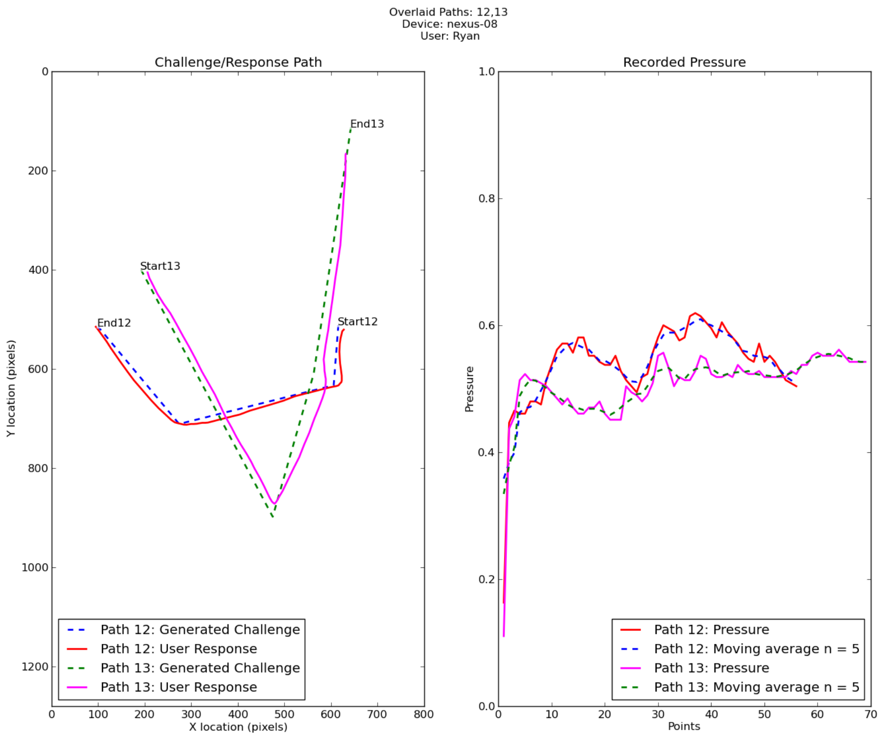

Figure 3: Two Distinct Paths/Same User/Same Device and Pressure Responses

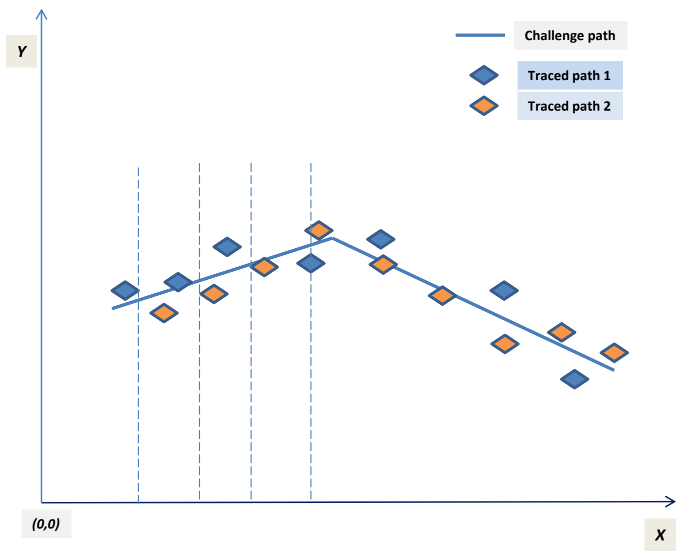

Figure 5: Point Alignment for Two Separate Traced Paths

user result in the same response. However, in Android events layer, two distinct sets of `MotionEvent` objects may be delivered for two separate traced paths. The OS sampling is asynchronous from our needs for synchrony between two traced paths.

Figure 5 shows a challenge path and two traced paths by the same user. The blue points path sampled MotionEvent objects are misaligned with the red/orange points path. Even the number of points sampled is different between the two paths. Comparing the two traced paths at the sampled points only amplifies the pressure value differences since the sampled points are not aligned at the same (x, y) coordinates.

First thing we perform is point alignment. We fix the evaluation points regardless of the locations of the sampled points. In Figure 5, the vertical dashed lines form these evaluation points. We have chosen to define the evaluation points to be vertical lines when the line segment slope is within the range $(-45°, 45°)$. Otherwise the sampling points are defined to be horizontal lines to maximize the projection resolution. All the traced paths are intrapolated to the evaluation points. In Figure 5, both the blue and red/orange paths will be evaluated at the 4 dashed vertical lines through interpolation. This will be a canonical representation for each traced path which is more likely to be reproducible.

Initial Reproducibility Schema:. We evaluated reproducibility after incorporating point alignment canonical representation. As shown in Figure 4, for a challenge path, two traced responses are drawn. We would like the Hamming distance of these traced paths' pressure values after quantization to be low. Hamming distance varies with the quantization scheme. In this case, Hamming distance between the two path responses was 14 bits out of a 40-bit response with moving average $n = 5$ quantization. This Hamming distance reduces to 3 with cumulative $n = 40$ quantization.

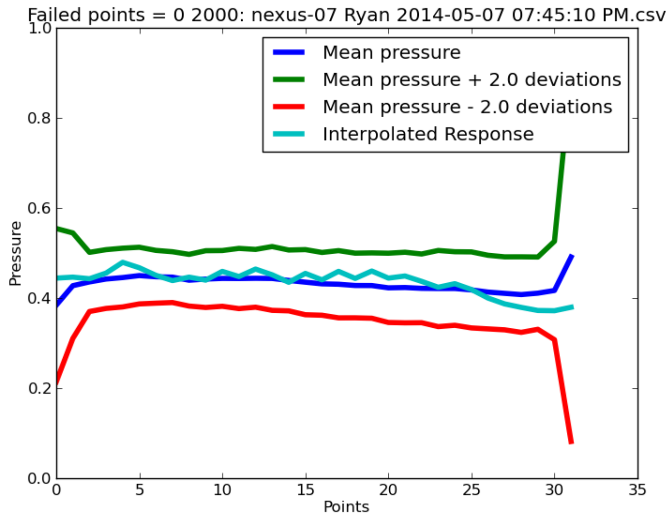

Figure 6: Statistical Concentration/Correction for Profiled User

The point alignment by itself does not seem to be sufficient for reproducibility. Ideally, we would have liked to bring down the Hamming distance to be within 2-6 range, and then apply an error correcting code (ECC).

Statistical Concentration or Error Correction:. The profiling phase entails developing a challenge response model for a given user and device. As shown in Figure 6, there is a challenge path, whose proxy is the dark blue line capturing the mean of the sampled data. We define an acceptable statistical band around it - green and red lines in Figure 6.

For these experiments, we limited ourselves to single line segments. The number of evaluation points defined on these line segments is 32. This results in 32-bit responses. Of course, this can be easily extended to multi-line segment challenge paths with higher number of bits in responses.

For each challenge path, each user traces it 50 times. Hence, at each evaluation point i, we get 50 samples $p_{i,0}, p_{i,1}, \ldots, p_{i,49}$. Each evaluation point i is modeled as a statistical acceptor or hypothesis H_i. We assume that these points are distributed with a normal distribution $\mathcal{N}(\mu, \sigma)$. We estimate this distribution with classical estimation theory as $H_i = \mathcal{N}(\mu_i, \sigma_i)$.

Once this model has been built through profiling, let us consider a user tracing a response to a challenge. At the ith evaluation point, let the intrapolated pressure be p_i. By 3-sigma rule, the probability that the point $\mu - 3\sigma \le p_i \le \mu + 3\sigma \approx 0.997$ and $\mu - 2\sigma \le p_i \le \mu + 2\sigma \approx 0.95$.

Our statistical acceptor can set a range $[\mu - k\sigma, \mu + k\sigma]$. If k is high, the band of influence in Figure 6 denoted by green and red lines is bigger. If it is too high, another user's (U_j) response might be acceptable within the profile of user U_i. We have found $k = 2$ or the ECC band $[\mu - 2\sigma, \mu + 2\sigma]$ to be a good choice.

Figure 6 shows user U_i's response in cyan (light) blue within the profile generated for user U_i. Note that the response is fully contained within the acceptable band for each evaluation point. If p_i at ith evaluation point is outside $[\mu - 2\sigma, \mu + 2\sigma]$ band, that point is rejected. If a point is accepted, the mean value μ_i at that evaluation point becomes the response.

In summary, an intrapolated, traced sequence of pressure values $p_0, p_1, \ldots, p_k$ is transformed into a canonical $\mu_0, \mu_1, \ldots, \mu_k$ pressure sequence if no points are rejected. This pressure sequence $\mu_0, \mu_1, \ldots, \mu_k$ will then be quantized using moving average $n = 5$ arbiter to generate the binary response.

Security Analysis. Note that for normal distribution, the probability that a given point is within 2σ band is bounded by $Pr[\mu - 2\sigma \le p \le \mu + 2\sigma] = 0.95$ by 3-sigma rule. Similarly, $Pr[\mu - 3\sigma \le p \le \mu + 3\sigma] = 0.997$.

With 2σ method, the *false negative* probability that a valid response is rejected depends on the threshold k selected in the authentication algorithm. When $0 \le k \le N$ points out of N points from the response place outside the 2σ band of μ, the response is rejected.

Let us consider an N bit response. Each bit corresponds to a normal distribution. Assume that each bit is independent of the others. Let us start with $k = 1$, that is if even one point fails to place within the 2σ band, the user is not authenticated. The probability that at least one pressure point out of N points falls outside the acceptable 2σ band is *1 - Pr(all N points place within* 2σ *band).* This is given by $1 - .95^N$. For $N = 32$, this false negative probability is 0.80629, which is quite high. If we set $k = 2$, the false negative probability that at least two points will fall outside 2σ band for a genuine user is *1- Pr(all N points place within* 2σ *band) - Pr(Exactly one point places outside* 2σ *band).* This is given by $1 - .95^N - N * .05 * .95^{N-1}$. For $N = 32$ with $k = 2$, this probability is .48004. Continuing this further, for $k = 3$, this probability is *1- Pr(all N points place within* 2σ *band) - Pr(Exactly one point places outside* 2σ *band) - Pr(Exactly two points place outside* 2σ *band).* This is $1 - .95^N - N * .05 * .95^{N-1} - \binom{N}{2} * .05^2 * .95^{N-2}$. For $N = 32$, this probability reduces to .21389.

Continuing in the same vein, if we go to $k = 4$, this probability rapidly reduces to $1 - .95^N - N * .05 * .95^{N-1} - \binom{N}{2} * .05^2 * .95^{N-2} - \binom{N}{3} * .05^3 * .95^{N-3} - \binom{N}{4} * .05^4 * .95^{N-4}$. For $N = 32$, this value is 0.020354, which is an acceptable value. However, if we go upto $k = N/4$, this expression reduces significantly to $1 - (\sum_{i=0}^{k} \binom{N}{i} * .05^i * .95^{N-i})$. For $N = 32$, $k = N/4 = 8$, this evaluates to $1.9112e^{-005}$. Overall, the choice of $k = N/4$ will give negligible probabilities for false-negative.

With $k = N/4$, in the false negative probability expression among the k terms subtracted from 1 in $\sum_{i=0}^{k} \binom{N}{i} * .05^i * .95^{N-i}$, the $i = k = N/4$ term dominates. This term has the form $\binom{N}{N/4} * .05^{N/4} * .95^{3N/4}$. By Stirling's approximation for $N!$ given by $\sqrt{2\pi N}\left(\frac{N}{e}\right)^N$, $\binom{N}{N/4}$ can be simplified to approximately $\frac{(4/3)^{3N/4} * 4^{N/4}}{\sqrt{(3/8) * \pi * N}}$. When multiplied by the rest of the multiplicands, this leads to $\frac{.2^{N/4} * 1.2667^{3N/4}}{\sqrt{(3/8) * \pi * N}}$. This term asymptotically grows rapidly to subtract significant amount from 1 to lead to asymptotically 0 false negative probability.

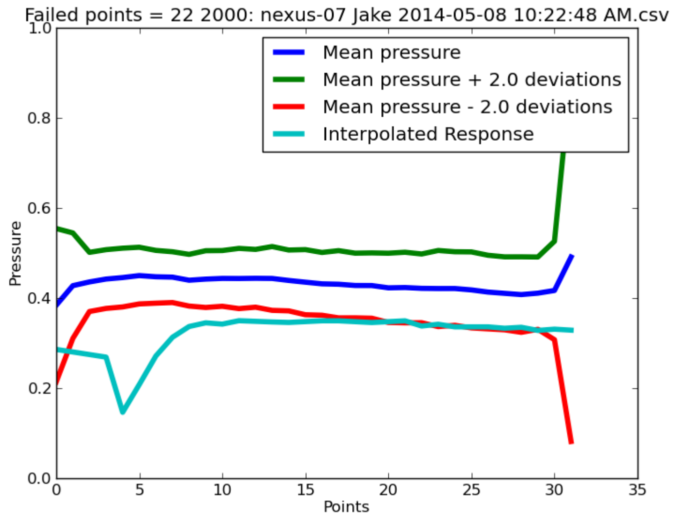

Figure 7: Statistical Concentration/Correction for a User other than Profiled User

An alternate mechanism to reduce the false negative probability would be to perform authentication a few times, say 3 times. If majority of authentication runs succeed, the response is authenticated. This is similar to triple modular redundancy (TMR) system. With .02 probability of false negative for $N = 32$ with $k = 4$, the TMR false negative probability is further reduced to $1 - .98^3 - 3 * .02 * .98^2$ which equals .00118, one order reduction.

Authentication of User U_j *in User* U_i *Profile.* Figure 7 shows the same data when user U_i profile is used to evaluate a path generated by user U_j. Note that on average, in this scenario, 22 out of 32 points are rejected with acceptable band $[\mu - 2\sigma, \mu + 2\sigma]$. If we were to increase the acceptable band to $[\mu - 3\sigma, \mu + 3\sigma]$, the number of rejected points will decrease reducing the robustness of the reproducibility phase.

6. UD-PUF AS A PSEUDO RANDOM NUMBER GENERATOR (PRG)

An Ideal PUF will appear to generate its responses to successive challenges as if they came from a pseudo-random generator. If the PUF behavior is close to that of a PRG, an adversary is not likely to succeed in modeling it. This prevents a large class of attacks against a PUF. For classical silicon PUFs, such characterizations do not exist. Most silicon PUFs target sufficient Hamming distance separation between responses. If pseudo-random-ness in responses is needed, the PUF responses are put through a hash function such as SHA-1.

Such a characterization is also ideal for small data sets. Note that out of 2^{128} possible responses, our experiments generated a very small subset. More precisely, if our challenge is 80 bits (10-bits for each x or y coordinate, and 4 such x, y pairs). The overall challenge, response space is 208 bits with a 128-bit response. We only generated a few thousand challenge-response pairs. This is a very sparse data set. Estimating average Hamming distance of the underlying domain is a very difficult, and poorly understood problem. Predicting whether such a thousand long sequence came from a PRG has been studied more thoroughly. Simard et al. [9] have developed a TESTU01 suite for pseudo-randomness that puts the response sequences through a variety of linear-congruential PRG equivalence. It also checks for weaknesses such as long 0- and 1- runs.

We put our responses from all four quantization techniques (1) flat average quantization, (2) $n = 5$ moving average quantization, (3) $n = 10$ moving average quantization, and (4) cumulative moving average quantization through TESTU01 suite of tests. The following results indicate PRG effectiveness of these responses on a battery of 26 statistical tests enumerated in the following.
1. smultin_MultinomialBitsOver.
2. snpair_ClosePairsBitMatch in $t = 2$ dimensions.
3. snpair_ClosePairsBitMatch in $t = 4$ dimensions.
4. svaria_AppearanceSpacings.
5. scomp_LinearComp.
6. scomp_LempelZiv.
7. sspectral_Fourier1.
8. sspectral_Fourier3.
9. sstring_LongestHeadRun.
10. sstring_PeriodsInStrings.
11. sstring_HammingWeight with blocks of $L = 32$ bits.
12. sstring_HammingCorr with blocks of $L = 32$ bits.
13. sstring_HammingCorr with blocks of $L = 64$ bits.
14. sstring_HammingCorr with blocks of $L = 128$ bits.
15. sstring_HammingIndep with blocks of $L = 16$ bits.
16. sstring_HammingIndep with blocks of $L = 32$ bits.
17. sstring_HammingIndep with blocks of $L = 64$ bits.
18. sstring_AutoCor with a lag $d = 1$.
19. sstring_AutoCor with a lag $d = 2$.
20. sstring_Run. 21. smarsa_MatrixRank with 32x32 matrices.
22. smarsa_MatrixRank with 320x320 matrices.
23. smarsa_MatrixRank with 1024x1024 matrices.
24. swalk_RandomWalk1 with walks of length $L = 128$.
25. swalk_RandomWalk1 with walks of length $L = 1024$.
26. swalk_RandomWalk1 with walks of length $L = 10016$.

(1) Flat average quantization:

```
Version:  TestU01 1.2.3
File:/home/git/PUFProject/OutputGenerated/Strat1
Number of bits:   480
Number of statistics:  16
Total CPU time:   00:00:00.01
The following tests gave p-values outside
    [0.001, 0.9990]:
(eps  means a value < 1.0e-300):
(eps1 means a value < 1.0e-15):

      Test                          p-value
----------------------------------------------
 1  MultinomialBitsOver           4.2e-104
 2  ClosePairsBitMatch, t = 2       1.3e-6
 4  AppearanceSpacings             1 - 4.1e-10
 6  LempelZiv                      1 - eps1
12  HammingCorr, L = 32              eps
13  HammingCorr, L = 64              eps
14  HammingCorr, L = 128             eps
18  AutoCor                        1 - eps1
19  AutoCor                        1 - eps1
20  Run of bits                      eps
20  Run of bits                      eps
----------------------------------------------
All other tests were passed
```

(2) $n = 5$ moving average quantization:

```
Version: TestU01 1.2.3
File: /home/git/PUFProject/OutputGenerated/Strat2
Number of statistics:  16
Total CPU time:   00:00:00.01
The following tests gave p-values outside
       [0.001, 0.9990]:
(eps  means a value < 1.0e-300):
(eps1 means a value < 1.0e-15):

      Test                          p-value
----------------------------------------------
 1  MultinomialBitsOver           2.8e-12
 6  LempelZiv                       0.9996
19  AutoCor                         3.2e-6
20  Run of bits                     3.9e-9
----------------------------------------------
All other tests were passed
```

(3) $n = 10$ moving average quantization:

```
Version: TestU01 1.2.3
File: /home/git/PUFProject/OutputGenerated/Strat3
Number of bits:   480
Number of statistics:  16
Total CPU time:   00:00:00.01
The following tests gave p-values outside
     [0.001, 0.9990]:
(eps  means a value < 1.0e-300):
(eps1 means a value < 1.0e-15):

      Test                          p-value
----------------------------------------------
 1  MultinomialBitsOver            3.4e-38
 6  LempelZiv                      1 -  2.9e-6
18  AutoCor                        1 - eps1
20  Run of bits                      eps
20  Run of bits                      eps
----------------------------------------------
All other tests were passed
```

(4) cumulative moving average quantization:

```
Version: TestU01 1.2.3
```

```
File: /home/git/PUFProject/OutputGenerated/Strat4
Number of bits:   480
Number of statistics:  16
Total CPU time:   00:00:00.01
The following tests gave p-values outside
    [0.001, 0.9990]:
(eps  means a value < 1.0e-300):
(eps1 means a value < 1.0e-15):

      Test                          p-value
----------------------------------------------
  1  MultinomialBitsOver          3.2e-121
  4  AppearanceSpacings            1 -  3.1e-9
  6  LempelZiv                     1 - eps1
 12  HammingCorr, L = 32             eps
 13  HammingCorr, L = 64             eps
 14  HammingCorr, L = 128            eps
 18  AutoCor                       1 - eps1
 19  AutoCor                       1 - eps1
 20  Run of bits                     eps
 20  Run of bits                     eps
-----------------------------------------------
All other tests were passed
```

Note that out of twenty-six PRG tests applied in the TestU01 battery, $n = 5$ moving average quantization fails only 4 categories, $n = 10$ moving average quantization fails 5, cumulative moving average quantization fails 10, and flat average quantization fails 11. We plan to use $n = 5$ moving average as our quantization technique on this basis. The $n = 5$ moving average quantization consistently fails *AutoCorr*, *MultinomialBitsOver* , and *Run Of Bits*. It fails *LempelZiv* occasionally. This is fairly good performance as PRG.

7. CONCLUSIONS

A Mobile device is more tightly coupled to its user than the traditional computing model. Delivering cloud based services in a user-device differentiated domain is a challenging problem. We offer a user-device entangled physical unclonable function (PUF) as a mechanism to provide differentiated user-device services, authentication and fingerprinting.

UD-PUF captures biometric of both the silicon on the device and the native user behavior. For UD-PUF to be effective, it should exhibit high variability in its response over any of user, device, or challenge axes. Challenge is a path drawn on the touch screen. We have established high degree of variability in responses. For a change in challenge, device, or user, 60+ bits change in a 128-bit response. n-moving average based arbiter is needed to provide dampening to generate high variability.

We also have to ensure that for the same challenge, same user, and same device, response can be reproduced. We achieve this by deploying a statistical concentrator/acceptor/ECC scheme. We show that perfect differentiation between users is achieved when a traced path (response) is evaluated within a given user's profile.

In summary, we have established the viability of UD-PUF as an authentication mechanism for mobile devices.

8. FUTURE WORK

This is only the first step in studying such UD PUFs. Additional security vulnerabilities of these PUFs need to be explored. For instance, if the adversary can replay the MotionEvent objects through a compromised authentication framework in the OS, all bets are off. What kind of root-of-trust should such authentication frameworks be wrapped in? How repeatable are the UD PUF challenge-response pairs - with gloves, in a moving vehicle, in a dusty environment? Do such UD PUFs enable a new class of hardware entangled cryptography?

9. REFERENCES

[1] A. Aysu, N. F. Ghalaty, Z. Franklin, M. P. Yali, and P. Schaumont. Digital fingerprints for low-cost platforms using mems sensors. In *Proceedings of the Workshop on Embedded Systems Security*, WESS '13, pages 2:1–2:6, New York, NY, USA, 2013. ACM.

[2] H. Bojinov and D. Boneh. Stanford researchers discover alarming method for phone tracking, fingerprinting through sensor flaws, October 2013. San Francisco Chronicle, http://blog.sfgate.com/techchron/2013/10/10/stanford-researchers-discover-alarming-method-for-phone-tracking-fingerprinting-through-sensor-flaws/.

[3] S. Devadas. Physical unclonable functions and secure processors. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 65–65, Berlin, Heidelberg, 2009. Springer-Verlag.

[4] S. Dey, N. Roy, W. Xu, and S. Nelakuditi. Acm hotmobile 2013 poster: Leveraging imperfections of sensors for fingerprinting smartphones. *SIGMOBILE Mob. Comput. Commun. Rev.*, 17(3):21–22, Nov. 2013.

[5] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. In *Proceedings of the 18th Annual Computer Security Applications Conference*, ACSAC '02, pages 149–, Washington, DC, USA, 2002. IEEE Computer Society.

[6] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160, New York, NY, USA, 2002. ACM.

[7] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. Fpga intrinsic pufs and their use for ip protection. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '07, pages 63–80, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] S. Katzenbeisser, U. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. Pufs: Myth, fact or busted? a security evaluation of physically unclonable functions (pufs) cast in silicon. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'12, pages 283–301, Berlin, Heidelberg, 2012. Springer-Verlag.

[9] P. L'Ecuyur and R. Simard. Testu01: A c library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33(4):22–40, 2007.

[10] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive Mob. Comput.*, 5(6):657–675, Dec. 2009.

[11] D. Merli, G. Sigl, and C. Eckert. Identities for embedded systems enabled by physical unclonable functions. In M. Fischlin and S. Katzenbeisser, editors, *Number Theory and Cryptography*, volume 8260 of *Lecture Notes in Computer Science*, pages 125–138. Springer Berlin Heidelberg, 2013.

[12] National Institute of Standards and Technology (NIST). *FIPS-180-2: Secure Hash Standard*, Aug 2002. available online at http://www.itl.nist.gov/fipspubs/.

[13] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[14] N. Sae-Bae, N. Memon, K. Isbister, and K. Ahmed. Multitouch gesture-based authentication. *Information Forensics and Security, IEEE Transactions on*, 9(4):568–582, April 2014.

[15] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 9–14, New York, NY, USA, 2007. ACM.

[16] G. E. Suh, C. W. O'Donnell, and S. Devadas. Aegis: A single-chip secure processor. *IEEE Design & Test of Computers*, 24(6):570–580, 2007.

[17] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas. Design and implementation of the aegis single-chip secure processor using physical random functions. In *Proceedings of the 32Nd Annual International Symposium on Computer Architecture*, ISCA '05, pages 25–36, Washington, DC, USA, 2005. IEEE Computer Society.

[18] Z. Syed, S. Banerjee, and B. Cukic. Leveraging variations in event sequences in keystroke-dynamics authentication systems. *Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05)*, 0:9–16, 2014.

[19] J. Zhang, X. Wang, P. Wu, and J. Zhu. Sensec: Mobile security through passive sensing. *2013 International Conference on Computing, Networking and Communications (ICNC)*, 0:1128–1133, 2013.