Sensor-based Wearable PUF

Kazuhide Fukushima, Seira Hidano and Shinsaku Kiyomoto
KDDI R&D Laboratories, Inc., 2-1-15 Ohara, Fujimino-shi, Saitama, 356-8502, Japan

Keywords: Wearable, IoT, Smartphone, Sensor, Accelerometer, Gyroscope, PUF.

Abstract: The Physically Unclonable Function (PUF) is a technique that generates unique device identifiers based on variations in the manufacturing process. The Internet of Things (IoT) has become widespread, and various kinds of devices are now available. Device authentication and key management are essential to provide a secure service to these devices. We can use the unforgeable identifier generated by the PUF as a key for encryption and authentication. However, the existing PUFs require a dedicated hardware or low-level software, i.e., driver. Thus, they are impractical to use on smartphones or IoT devices due to the severe limitations of production cost and power consumption. In this paper, we propose a sensor-based PUF that utilizes the accelerometer and gyroscope, which are widely available on smartphones and IoT devices. We implement the proposed PUF on a smartwatch and show that accelerometer-based PUF achieves good usability, extreme robustness, and a high entropy of 91.66 bits.

1 INTRODUCTION

Android is an open mobile platform developed by the Open Handset Alliance (OHA) led by Google, Inc (Open Handest Alliance, 2010). It had grown to be the top-share smartphone platform in the world (82.8% market share) in the second quarter of 2015 (IDC, 2015), and more than 1,800,000 applications are available from the Android Market as of November 2015 (Statista, 2015). Google have announced the release of the Brillo, which is an Android-based platform for the Internet of Things (IoT) (Google, 2015). Users can add new features to their smartphones and IoT devices by installing applications. However, an attacker may analyze applications to find a secret key to decrypt protected content or get authentication information illegally.

One approach is to generate the unique key dynamically based on the device information of smartphone or IoT device. For example, the MAC address of a Wi-Fi adapter and Bluetooth adapter has been used as the input to a key generation function. However, the MAC address can be easily modified in a device where the administrator privilege is compromised. Furthermore, the current version of Android and iOS prohibit general applications from getting the MAC address. Android 6.0 (API level 23) returns a fixed value `02:00:00:00:00:00` for *WifiInfo.getMacAddress()* and *BluetoothAdapter.getAddress()* that is a method to get the MAC address of the Wi-Fi and Bluetooth adapter respectively (Google, 2015). iOS7 returns the same value for similar APIs (Apple, 2013). The protection of the key generation algorithm is another critical issue. The Android SDK contains an obfuscation tool: ProGuard (Lafortune, 2002) to protect against unauthorized analysis and modification. Nonetheless, this mechanism offers only limited protection since it relies solely on a software mechanism.

Another approach is hardware-based protection, and one idea is to use tamper-proof hardware. Mobile phones have a tamper-proof device, such as a user identity module (UIM) (3GPP2, 2000) or subscriber identity module (SIM) (The 3rd Generation Partnership Project (3GPP), 1990) that provides secure storage for service-subscriber keys and secure computational capability. The serial number of the SIM card can be used as a valid identifier. Android provides the *getSimSerialNumber()* method to get the identifier. The Trusted Computing Group (TCG) has established technology specifications for the Trusted Platform Module (TPM) that is available in smartphones and PCs (Trust Computing Group, 2016). The TPM provides the cryptographic functions to enhance the security of the platform, and it is used as a root of trust. However, most IoT devices, including wearables, do not have dedicated hardware.

The Physically Unclonable Function (PUF), which generates unique device identifiers based on variation in the manufacturing process, is a promis-

Fukushima, K., Hidano, S. and Kiyomoto, S.
Sensor-based Wearable PUF.
DOI: 10.5220/0005946702070214
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT*, pages 207-214
ISBN: 978-989-758-196-0

ing alternative. We can use the device identifier as a key. The device identifier generated by the PUF is hard to analyze since it does not appear in digital format on the device. However, the existing PUFs depend on additional and dedicated hardware; thus, they are impractical to use in IoT devices due to the severe limitations of production cost and power consumption. The Static Random-Access Memory (SRAM) PUF and Dynamic Random-Access Memory (DRAM) PUF utilize existing hardware, but they are still infeasible since they require low-level software, i.e., a driver.

Smartphones and IoT devices have various sensors and hardware such as an accelerometer, gyroscope, proximity sensor, microphone, speaker, and camera. Most wearables have an accelerometer and gyroscope to change the behavior based on user actions. In this paper, we propose a sensor-based PUF that utilizes the characteristic values of accelerometer and gyroscope. The proposed PUF is widely available on smartphones and IoT devices.

2 RELATED WORK

The Physically Unclonable Function (PUF) (Pappu et al., 2002) is a technique that generates unique device identifiers based on variations in the manufacturing process. The Arbiter PUF utilizes the difference in the signal transmission delay (Gassend et al., 2004; Lee et al., 2004), and the Glitch PUF is based on the signal transition (Suzuki and Shimizu, 2010). Kumar et al. proposed the Butterfly PUF, which uses the initial state of flip-flops (Kumar et al., 2008). Gassend et al. proposed the Ring Oscillator PUF based on the difference in the oscillating frequency of the ring oscillator (Gassend et al., 2002). Finally, Tuyls et al. proposed the Coating PUF based on the capacitance of the coating materials containing dielectric particles (Tuyls et al., 2006). These PUFs depend on additional and dedicated hardware; thus, they are impractical to use in smartphones or IoT devices.

Some PUFs based on the existing hardware have been proposed. An SRAM PUF utilizes the initial data in memory when the power is turned on (Chopra and Colopy, 2009; Maes et al., 2009a; Maes et al., 2009b). Krishna et al. proposed a memory-cell-based PUF that uses intrinsic process variations in the read and write reliability of cells in static memory (Krishna et al., 2011). Liu et al. proposed a DRAM PUF that uses the decay time and output stability (Liu et al., 2014). Keller et al. proposed a PUF based on the influence of temperature and time on the charge decay (Keller et al., 2014). A DRAM PUF proposed by Tehranipoor (Tehranipoor et al., 2015) uses initial data similar to an SRAM PUF. However, the SRAM and DRAM PUFs are still impractical in smartphones or IoT devices. They require dedicated drivers to extract the characteristic features of devices, which imposes an additional cost.

Thus, we propose a sensor-based PUF for smartphones and IoT devices. The proposed PUF can extract the characteristic values of sensors through the standard API of the OS. Thus, it does not require the additional hardware and dedicated drivers.

3 PROPOSED METHOD

We propose a sensor-based PUF for smartphones and IoT devices including wearables. The proposed PUF acquires the maximum and minimum values of the accelerometer and gyroscope as the characteristic values of the sensors and generates device identifiers based on these characteristic values. We describe the features of the accelerometer and gyroscope in Section 3.1 and 3.2, respectively. Section 3.3 describes the method to generate a device identifier based on characteristic values of the sensors.

3.1 Accerelometer

A 3-axis accelerometer measures the accelerations along the x, y, and z-axis. Figure 1 shows the direction of these axes. Most smartphones and wearables have an accelerometer that can measure up to ± 2 G (19.6 m/s^2) or ± 4 G (39.2 m/s^2).

The maximum and minimum values of the accelerometer along each axis differ from one device to another. Thus, these values can be used to generate device identification data. We construct a 6-dimensional data set that consists of the maximum and minimum values along the x, y, and z-axis and extract digits that have enough variety. The characteristic value of the accelerometer can be achieved by concatenating these digits.

3.2 Gyroscope

A 3-axis gyroscope measures the angular velocities around the x, y, and z-axis. Figure 2 shows the rotation direction around these axes. Many wearables have a gyroscope that can measure up to 2000 deg/s (34.9 rad/s).

The maximum and minimum values of the gyroscope around each axis differ from one device to another. Thus, these values can be used to generate device identification data. We construct a 6-dimensional

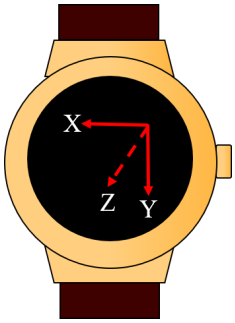

Figure 1: Axis direction of accelerometer.

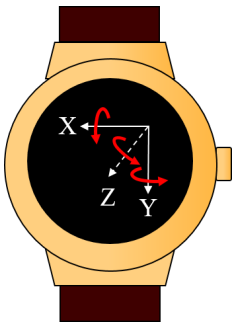

Figure 2: Rotation direction of gyroscope.

data set that consists of the maximum and minimum values around the x, y, and z-axis and extract digits that have enough variety. The characteristic value of the gyroscope can be achieved by concatenating these digits.

3.3 Device Identifier Generation

We now describe the detailed behavior of the sensor-based PUF to generate the device identifier. The identifier generation process consists of the acquisition of the characteristic values of the sensors (step 1) and identifier generation using a one-way function (step 2).

Step 1 Acquisition of the Characteristic Values of the Sensors. The sensor-based PUF requires the maximum values and minimum values of the accelerometer and gyroscope. A user needs to shake and twist the hand holding or wearing a device. This process stores the tentative maximum and minimum values of sensors. These tentative values are updated when the current sensor value is larger or smaller than the tentative maximum or minimum value, respectively. We consider the tentative values as the actual maximum and minimum values after a user shakes the device for a specified time (a few seconds), and the tentative values are stable. We show a sample implementation of this step in the appendix.

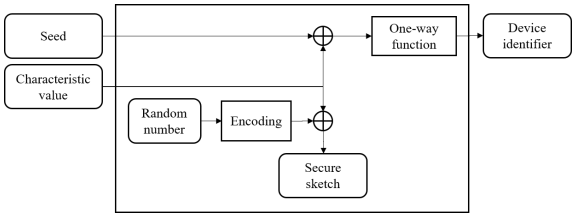

Figure 3: Fuzzy extractor in registration phase.

Step 2 Generation of Device Identifier. The sensor-based PUF uses a one-way function to generate a device identifier from the concatenated characteristic values of the accelerometer and gyroscope. A device identifier has to vary with each application or service when the identifier is used as a key. In this case, applications or services can concatenate a seed to the input to the one-way function to fulfill the requirement.

We can use a fuzzy extractor (Dodis et al., 2008) to deal with minor deviations of the characteristic values of the sensors. The fuzzy extractor generates and registers a secure sketch SS for future identifier extractions in the registration phase. It generates the random number r, encoded with an error-correcting code C. The secure sketch SS is the exclusive-or of the characteristic value of sensor w, and $C(r)$ or $SS = w \oplus C(r)$. The random number r is disposed of when the registration process completes. In the device identifier generation phase, the fuzzy extractor recovers the original characteristic value w from the secure sketch SS calculated from the current characteristic value w' in the device. It calculates $C(r)$ as

$$C(r) = EC(SS \oplus w') = EC[C(r) \oplus (w \oplus w')],$$

where EC is the error-correcting component and $w \oplus w'$ denotes the minor deviation of the characteristic values of the sensor. Finally, the fuzzy extractor generates the device identifier as

$$DeviceID(w', s) = h(w \oplus s) = h(SS \oplus C(r) \oplus s)$$

using the one-way function h. Figure 3 shows the fuzzy extractor in the registration phase and Figure 4 shows the device identifier generation phase.

The sensor values for each axis can be considered as independent. Thus, the sensor-based PUF separately applies the fuzzy extractor to each characteristic value (maximum and minimum values for x, y, and z-axis).

The proposed PUF encodes the characteristic values of the sensor to binary data with Gray code. The Hamming distance between adjacent values is one encoding; thus, minor deviations up to $\pm t$ can be recovered with the error correcting code of t correction

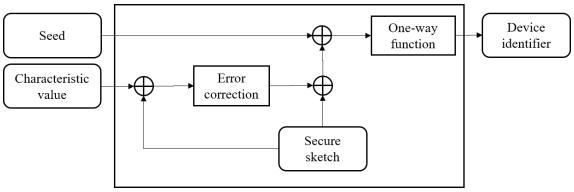

Figure 4: Fuzzy extractor in device identifier generation phase.

Table 1: Maximum and minimum values of acceleration.

	Maximum	Minimum
x-axis	19.460205 m/s^2	-19.539200 m/s^2
y-axis	19.228836 m/s^2	-19.770569 m/s^2
z-axis	19.755707 m/s^2	-19.243698 m/s^2

capability. However, the robustness achieved by the fuzzy extractor has a trade-off with entropy. We discuss the trade-off in section 5.2.

4 IMPLEMENTATION

We have implemented the sensor-based PUF as an Android Wear application. The PUF application is executed on an LG Watch Urbane smartwatch to evaluate the entropy and robustness.

The LG Watch Urbane has an accelerometer that can measure from -2 G (-19.6 m/s^2) to 2 G (-19.6 m/s^2). The maximum and minimum values of the sensor vary with the devices and axis. Table 1 shows an example. The number in the second decimal place and the following digits have enough variety. Thus, the PUF application extracts the numbers in the second to sixth decimal place as a character string with five figures from each of the maximum and minimum values of the accelerations along the x, y, and z-axis. The application can acquire the characteristic value of the accelerometer with 30 characters by concatenating these six character strings.

The LG Watch Urbane has a gyroscope that can measure from -2,000 degree/s (-34.9 rad/s) to 2,000 degree/s (34.9 rad/s). Table 2 shows an example. The number in the fourth decimal place and the following digits have enough variety. Thus, the PUF application extracts the numbers in the fourth to sixth decimal place as a character string with three figures from each of the maximum and minimum values of angular velocities around the x, y, and z-axis. The application can acquire a characteristic value of the accelerometer with 18 characters by concatenating these six character strings.

The PUF application generates a device identifier with 128-bit length from the characteristic value of an accelerator or gyroscope by using SHA-256 and truncating the last 128 bits. We do not use the fuzzy extractor in the application since it extracts the identical characteristic values from the accelerometer and gyroscope consistently.

Table 2: Maximum and minimum values of angular velocity.

	Maximum	Minimum
x-axis	34.997101 rad/s	-35.000763 rad/s
y-axis	34.997055 rad/s	-35.000809 rad/s
z-axis	34.997803 rad/s	-35.000061 rad/s

Figure 5: Screenshot of sensor-based PUF application.

Figure 5 shows a photo of the PUF application on the LG Watch Urbane. The application regenerates and displays the generated device identifier whenever the tentative maximum and minimum values of sensors are updated. The tentative maximum and minimum values are reset to zero when the user presses the reset button (RST). The application stores the device identifier when the register button (REG) is pressed, and deletes the identifier when the unregister button (URG) is pressed. We can use this function to evaluate the robustness of the device identifier. The application compares the generated identifier and stored identifier and displays ”OK” if they are identical or ”NG” otherwise.

We have compared the usability of the accelerometer-based PUF and gyroscope-based PUF in an LG Watch Urbane. The accelerometer-based PUF can generate the same identifier within a few seconds. The measurement range of the accelerometer is from -2 G to 2 G, and the sensor value quickly reaches the maximum or minimum when a user shakes the device. However, the identifier generated by the gyroscope-based PUF is not stable. The measurement range of the gyroscope is from -2,000 degrees/s to 2,000 degrees/s. A user has to rotate the device around three axes at a rate of six revolutions per second, which is difficult if the user continues to wear the device. As a result, the accelerometer-based PUF is more useful than the gyroscope-based PUF in

an LG Watch Urbane. Thus, we evaluate the entropy and robustness of the accelerometer-based PUF in the following section.

5 EVALUATION

We evaluate the compatibility of the sensor-based PUF with the requirements for PUFs. Then, we evaluate the robustness and entropy of the device identifier generated by the accelerometer-based PUF.

5.1 Compatibility with Requirements

Maes and Verbauwhede showed the requirements for PUFs: evaluable, unique, reproducible, unpredictable, one-way, unclonable and tamper evident (Maes and Verbauwhede, 2010). We show the compatibility of the sensor-based PUF with these requirements.

Evaluable. The sensor-based PUF uses the characteristic values of the existing sensors, and these values can be acquired through the existing APIs. The PUF can be implemented as a general application at minimal cost. Furthermore, the sensor-based PUF uses only the extraction and concatenation of character strings, one-way function, and optional exclusive-or and operations for the error-correcting code. Thus, the PUF is feasible in resource-restricted devices.

Unique. The device identifier generated by the sensor-based PUF is unique since it has high entropy. We evaluate the entropy of the proposed PUF in section 5.2.

Reproducible. The device identifier generated by the sensor-based PUF is reproducible since it has high robustness. We evaluate the robustness of the proposed PUF in section 5.3.

Unpredictable. The sensor-based PUF is not applicable to the challenge-response model. Thus, we cannot evaluate the unpredictability of the proposed PUF.

One-way. The sensor-based PUF uses a one-way function to generate the device identifier from the characteristic values. The one-way function guarantees this requirement for the sensor-based PUF.

Unclonable and Tamper Evident. The sensor-based PUF acquires the characteristic values of sensors using software (application) and generates the device identifier. Thus, we have to protect against acquisition and modification of the characteristic values and generated device identification. For example, attackers can make a clone of the sensor-based PUF by acquiring the characteristic values of the sensor. They can also modify the values of the sensors or the generated device identifier in the memory. Thus, we have to protect against these attacks by applying memory protection techniques(Ostrovsky, 1990; Goldreich and Ostrovsky, 1996; Nakano et al., 2012). Attackers may modify the identifier generation algorithm so that it outputs an arbitrary identifier. We have to protect these attacks by applying software tamper-proof techniques to prevent modification of the application. We will study the security against physical attacks in our future research.

5.2 Entropy

We have evaluated the Quadratic Renyi entropy of the device identifier generated by an accelerometer-based PUF. The Quadratic Renyi entropy is defined as:

$$H_2(B) = -\log_2 \sum_{b \in \mathcal{B}} p_B(b)^2.$$

$H_2(B)$ is a particular case ($\alpha = 2$) of the Renyi entropy $H_\alpha(B)$, which is defined as:

$$H_\alpha(B) = \frac{1}{1-\alpha} \log_2 \sum_{b \in \mathcal{B}} p_B(b)^\alpha.$$

The Shannon entropy is defined as

$$H(B) = -\sum_{b \in \mathcal{B}} p_B(b) \log_2 p_B(b).$$

The Renyi entropy and Shannon entropy have the relationship: $\lim_{\alpha \to 1} H_\alpha(B) = H(B)$. The Reyni entropy is decreasing with respect to α. Thus, the Quadratic Renyi entropy is smaller than or equal to the Shannon entropy, or $H_2(B) \leq H(B)$. The same holds when the B is according to the uniform distribution.

The $\sum_{b \in \mathcal{B}} p_B(b)^2$ is the probability that the two sampled values are identical, and the Quadratic Renyi entropy is referred to as the collision entropy. The Quadratic Renyi entropy can be written as

$$H_2(B) = -\log_2 p_D(0)$$

with the probabilistic mass function of the distance D between two sampled values $p_D(d)$.

The Quadratic Renyi entropy has been used to evaluate the entropy of biometric information (Hidano et al., 2010; Hidano et al., 2012). However, it is

difficult to predict the probability $p_D(d)$ directly with a limited number of samples. Thus, we used a non-parametric approach with a specific kernel function that can estimate $p_D(d)$ with a small error. Kokonendji (Kokonendji et al., 2007) proposed a non-parametric estimation of probability mass functions using discrete triangular distribution. The probability mass function can be estimated as

$$\hat{p}_D(x) = \frac{1}{n}\sum_{i=1}^{n} T_{a,h,x}(X_i),$$

where X_1, X_2, ..., X_n are random samples from a count distribution with an unknown probability mass function. $T_{a,h,x}(X)$ is the probability mass function of the discrete triangular distribution of the order h and with the arm a. $T_{a,h,x}(X)$ is given as:

$$T_{a,h,x}(X) = \begin{cases} \frac{(a+1)^h - |X-x|^h}{P(a,h)} & (X-a \le x \le X+a) \\ 0 & (\text{otherwise}) \end{cases}$$

where $P(a,h) = (2a+1)(a+1)^h - 2\sum_{k=0}^{a} k^h$ is the normalizing constant. $p_D(0)$ can be estimated as

$$\hat{p}_D(0) = \frac{1}{n}\sum_{i=0}^{a} \frac{n_i[(a+1)^h - i^h]}{P(a,h)},$$

where n_i is the number of samples whose distance is i. The asymptotic mean integrated squared errors are given by:

$$\mathrm{AMISE}(a,h) = \frac{(a+1)^h}{nP(a,h)} + \frac{1}{4}[V(a,h)]^2 \sum_{x\in\mathbb{N}} [p''(x)]^2,$$

where $V(a,h)$ is the variance of $T_{a,h,x}(X)$ as

$$V(a,h) = \frac{1}{P(a,h)}\left[\frac{a(2a+1)(a+1)^h}{3} - 2\sum_{k=0}^{a} k^{h+2}\right]$$

and $p''(x)$ is the second derivative of $p(x)$ as

$$p''(x) = \begin{cases} p(2) - 2p(1) + p(0) & (x=0) \\ \frac{p(3)-p(2)-p(1)+p(0)}{2} & (x=1) \\ \frac{p(x+2)-2p(x)+p(x-2)}{4} & (\text{otherwise}) \end{cases}.$$

$\mathrm{AMISE}(a,h)$ is increasing with respect to a; thus, $a^* = 1$ is the optimal parameter for a. The optimal parameter h^* for h is given by:

$$h^* = \arg\min_{h>0} \mathrm{AMISE}(1,h).$$

We acquired 90 strings, which are the characteristic values of accelerometers described in section 3.1, from 15 of the same wearables (LG Watch Urbanes). Each string ranges from 00000 to 99999 and can be encoded with a 17-bit binary value. We exhaustively compared the binary values and studied the distribution of the Hamming distance.

Table 3: Distribution of Hamming distance.

Distance	Frequency
0	0
1	2
2	5
3	24
4	92
5	193
6	394
7	646
8	711
9	729
10	572
11	354
12	187
13	70
14	22
15	3
16	1
17	0
Total	4,005

Table 3 shows the distribution of the Hamming distance, and it is similar to the binomial distribution. Thus, we used $p(x)$ as the probabilistic mass function of the binomial distribution $B(17,1/2)$ to find the optimal parameter h^*. $\mathrm{AMISE}(1,h)$ is increasing with respect to h; however, $\hat{p}_D(0)$ goes to 0 as $h \to 0$. Thus, $a^* = 1$ and $h^* = 1/12$ are used as the optimal parameters. The estimated Quadratic Renyi entropy of the artificially generated histogram of the binomial distribution $B(17,1/2)$ is 16.19 bits using the optimal parameter setting.

We can estimate $\hat{p}_D(0) = 2.52 \times 10^{-5}$, from the distribution of the Hamming distance in Table 3. Thus, the Quadratic Renyi entropy is $\hat{H}_2(B) = \log_2 \hat{p}_D(0)$ or 15.28 bits. The device identifier based on the accelerometer consists of six-dimensional data, and the total entropy is 91.66 bits.

We discuss the trade-off between entropy and robustness in the case where the sensor-based PUF uses the fuzzy extractor. The entropy loss of each characteristic value of the sensors is at least $\log_2 \sum_{k=0}^{t} \binom{n}{k}$ bits according to the Hamming bound. n is the bit length of the encoded characteristic value, and t is the correction capability of the error-correcting code. The total entropy loss is at least $6\log_2 \sum_{k=0}^{t} \binom{n}{k}$ bits. Table 4 shows the relationship between the correction capability of the error-correcting code and the upper bound of the total entropy of the device identifier generated by the sensor-based PUF. The device identifier generated by the accelerometer-based PUF can achieve more than 60 bits of entropy when the fuzzy extractor is not used, or when the fuzzy extractor uses the error-correcting code with one-bit correction capability.

Table 4: Entropy loss due to error correction.

Capability	Entropy
0	91.66
1	66.64
2	48.06
3	33.43
4	21.75
5	12.47

5.3 Robustness

The sensor-based PUF using an accelerometer and gyroscope can generate the identical device identifier within the same device without the fuzzy extractor. We confirmed that the device identifier generated by the accelerometer-based PUF is consistent by the fact that the same user generates the same identifier more than 1,000 times. Furthermore, more than ten users can generate the same identification within the same device. The accelerometer-based PUF generates the same device identifier regardless of the surrounding temperature. Finally, it generates the same identifier in -5 degrees Celsius and 90 degrees Celsius and at 2000 meters of altitude on a mountain.

6 CONCLUSION

In this paper, we proposed a sensor-based PUF. The sensor-based PUF utilizes the accelerometer and gyroscope that are widely available in smartphone and IoT devices. We implemented the proposed PUF on a smartwatch and showed that the accelerometer-based PUF achieves good usability, extreme robustness, and a high entropy of 91.66 bits.

REFERENCES

3GPP2 (2000). Removable user identity module (R-UIM) for cdma2000 spread spectrum systems. http://www.3gpp2.org/public_html/specs/CS0023-0.pdf.

Apple (2013). What's New in iOS 7.0 – Apple Developer. https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS7.html.

Chopra, J. and Colopy, R. (2009). SRAM Characteristics as Physical Unclonable Functions. Worcester Polytechnic Institute Electric Project Collection, http://www.wpi.edu/Pubs/E-project/Available/E-project-031709-141338/.

Dodis, Y., Ostrovsky, R., Reyzin, L., and Smith, A. (2008). Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM Journal on Computing*, 38(1):97–139.

Gassend, B., Clarke, D., Lim, D., van Dijk, M., and Devada, S. (2004). Identification and Authentication of Integrated Circuits. *Concurrency and Computation: Practice and Experience,*, 16(11):1077–1098.

Gassend, B., Clarke, D., van Dijk, M., and Devadas, S. (2002). Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS2002*, page 148.

Goldreich, O. and Ostrovsky, R. (1996). Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473.

Google (2015). Android 6.0 changes, access to hardware identifier. http://developer.android.com/intl/ja/about/versions/marshmallow/android-6.0-changes.html#behavior-hardware-id.

Google (2015). Brillo. https://developers.google.com/brillo/.

Hidano, S., Ohki, T., Komatsu, N., and Takahashi, K. (2010). A metric of identification performance of biometrics based on information content. In *Proceedings of 11th International Conference on Control, Automation, Robotics and Vision, ICARCV2010*, pages 1274–1279.

Hidano, S., Ohki, T., and Takahashi, K. (2012). Evaluation of security for biometric guessing attacks in biometric cryptosystem using fuzzy commitment scheme. In *Proceedings of 2012 International Conference of the Biometrics Special Interest Group, BIOSIG*, pages 1–6.

IDC (2015). Smartphone OS Market Share, 2015 Q2. http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

Keller, C., Gurkaynak, F., Kaeslin, H., and Felber, N. (2014). Dynamic memory-based physically unclonable function for the generation of unique identifiers and true random numbers. In *Proceedings of IEEE International Symposium on Circuits and Systems*, volume 3, pages 2740–2743.

Kokonendji, C. C., Kiesse, T. S., and Zocchi, S. S. (2007). Discrete triangular distributions and nonparametric estimation for probability mass function. *Journal of Nonparametric Statistics*, 19:241–254.

Krishna, A. R., Narasimhan, S., Wang, X., and Bhunia, S. (2011). MECCA: A robust low-overhead PUF using embedded memory array. In *Proceedings of the Cryptographic Hardware and Embedded Systems, CHES2011*, pages 407–420.

Kumar, S. S., Guajardo, J., Maes, R., Schrijen, G. J., and Tuyls, P. (2008). The Butterfly PUF protecting IP on every FPGA. In *Proceedings of 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST2008*, pages 67–70.

Lafortune, E. (2002). ProGuard. http://proguard.sourceforge.net/.

Lee, J., Lim, D. L. D., Gassend, B., Suh, G., Dijk, M. V., and Devadas, S. (2004). A technique to build a secret key in integrated circuits for identification and authentication applications. In *Proceedings of 2004 Symposium on VLSI Circuits.*, pages 176–179.

Liu, W., Zhang, Z., Li, M., and Liu, Z. (2014). A trustworthy key generation prototype based on DDR3 PUF for wireless sensor networks. In *Proceedings of 2014 International Symposium on Computer, Consumer and Control, IS3C 2014*, pages 706–709.

Maes, R., Tuyls, P., and Verbauwhede, I. (2009a). A soft decision helper data algorithm for SRAM PUFs. In *Proceedings of IEEE International Symposium on Information Theory, ISIT2009*, pages 2101–2105.

Maes, R., Tuyls, P., and Verbauwhede, I. (2009b). Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In *Proceedings of the Cryptographic Hardware and Embedded Systems, CHES2009*, pages 332–347.

Maes, R. and Verbauwhede, I. (2010). Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. *Towards Hardware-Intrinsic Security*, pages 3–37.

Nakano, Y., Cid, C., Kiyomoto, S., and Miyake, Y. (2012). Memory access pattern protection for resource-constrained devices. In *Proceedings of Smart Card Research and Advanced Application Conference, CARDIS2012*, pages 188–202.

Open Handest Alliance (2010). Open Handest Alliance. http://www.openhandsetalliance.com/index.html.

Ostrovsky, R. (1990). Efficient computation on oblibious RAMs. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, STOC1990*, pages 514–523.

Pappu, R., Recht, B., Taylor, J., and Gershenfeld, N. (2002). Physical One-Way Functions. *Science*, 297:2026–2030.

Statista (2015). Number of available applications in the Google Play Store from December 2009 to November 2015. http://www.statista.com/statistics/266210/.

Suzuki, D. and Shimizu, K. (2010). The Glitch PUF : A New Delay-PUF. In *Proceedings of the Cryptographic Hardware and Embedded Systems, CHES2010*, pages 366–382.

Tehranipoor, F., Karimina, N., Xiao, K., and Chandy, J. (2015). DRAM based Intrinsic Physical Unclonable Functions for System Level Security. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI, GLSVLSI '15*, pages 15–20.

The 3rd Generation Partnership Project (3GPP) (1990). Specification of the subscriber identity module - mobile equipment (sim-me) interface. http://www.3gpp.org/ftp/Specs/html-info/1111.htm.

Trust Computing Group (2016). Trusted Platform Module. http://www.trustedcomputinggroup.org/developers/trusted_platform_module.

Tuyls, P., Schrijen, G.-J., Škorić, B., van Geloven, J., Verhaegh, N., and Wolters, R. (2006). Read-Proof Hardware from Protective Coatings. In *Proceedings of the Cryptographic Hardware and Embedded Systems, CHES2006*, pages 369–383.

```
@Override
public void onSensorChanged
                    (SensorEvent event) {
  if (event.sensor.getType() ==
         Sensor.TYPE_ACCELEROMETER) {
    boolean upd = false;
    float x = event.values[0];
    float y = event.values[1];
    float z = event.values[2];

    if (x > maxX) { maxX = x; upd = true; }
    if (x < minX) { minX = x; upd = true; }
    if (y > maxY) { maxY = y; upd = true; }
    if (y < minY) { minY = y; upd = true; }
    if (z > maxZ) { maxZ = z; upd = true; }
    if (z < minZ) { minZ = z; upd = true; }
    if (upd) {
        // Generate device identifier based
        // on updated tentative values
        deviceId = generateDeviceId();
    }
  }
}
```

Figure 6: Acquisition of maximum and minimum values of the accelerometer.

APPENDIX

Figure 6 shows a sample implementation to acquire the maximum and minimum values of the accelerometer in an Android device. The fields *maxX*, *minX*, *maxY*, *minY*, *maxZ*, and *minZ* are fields that store the tentative maximum and minimum values of accelerations along the x, y and z-axis. The method *onSensorChanged* is called when the sensor values have changed. We retrieve the event from the accelerometer by using the *if* statement. The acceleration along the x, y, and z-axis is stored in the *values* array. The same code where `Sensor.TYPE_ACCELEROMETER` is replaced with `Sensor.TYPE_GYROSCOPE` can acquire the maximum and minimum values of angular velocities around each axis.

The sensor-based PUF needs to set the highest sampling frequency on the sensors so that we can efficiently acquire the maximum and minimum values. We can set the sampling frequency through the *registerListener* method in Android. The method registers *SensorEventListener* that is used to receive notifications from the *SensorManager* when the sensor values have changed. The notification frequency is highest, and the period is a few milliseconds if `SENSOR_DELAY_FASTEST` is passed to the method.