SHAKE: SHared Acceleration Key Establishment for Resource-Constrained IoT Devices

Bejder, Emil; Mathiasen, Adam Krog ; De Donno, Michele; Dragoni, Nicola; Fafoutis, Xenofon

Published in:
Proceedings of IEEE 6th World Forum on Internet of Things

Link to article, DOI:
10.1109/WF-IoT48130.2020.9221263

Publication date:
2020

Document Version
Peer reviewed version

SHAKE: SHared Acceleration Key Establishment for Resource-Constrained IoT Devices

Emil Bejder*, Adam Krog Mathiasen*, Michele De Donno*, Nicola Dragoni*† and Xenofon Fafoutis*
*DTU Compute, Technical University of Denmark, Denmark
Email: {s164161, s164168}@student.dtu.dk, {mido, ndra, xefa}@dtu.dk
†AASS, Örebro University, Sweden

***Abstract*—IoT security for resource-constrained devices is largely based on symmetric block ciphers, such as AES. In such resource-constrained contexts, and particularly in the case of large-scale IoT deployments with multiple devices, the installation of encryption keys can pose a significant challenge. This paper presents SHAKE (SHared Acceleration Key Establishment): a convenient means to generate and install secret keys in IoT devices during deployment. Using SHAKE, an IoT deployment technician can generate and install a shared encryption key on two devices by holding them together and *shaking* them. SHAKE, operating on each of the devices, captures these movements from an on-board accelerometer and generates a secret key based on the shared acceleration profile. We provide a proof-of-concept implementation of SHAKE for the Contiki-NG operating system and assess its security against mimic attacks, that is the scenario whereby an eavesdropper with a clear line of sight to the deployment technician tries to mimic the random movements to generate the same key. Finally, we assess the energy requirements for generating a 128-bit key with SHAKE and we compare it against state-of-the-art methods for key generation.**

***Index Terms*—Secret key generation, IoT security, IoT deployments, Resource-constrained devices, Internet of Things**

I. Introduction

In the emerging era of the Internet of Things (IoT), there are numerous use cases, in which networks of low-end devices, such as sensors and actuators, support critical infrastructures with security requirements, such confidentiality, integrity and authentication, amongst others [1]. Whilst these security requirements are largely similar to a traditional wireless network, the key challenge in low-end IoT devices is their severe resource constraints in terms of processing power, memory, and energy [2]. Indeed, traditional security schemes are not directly applicable in such resource-constrained contexts, and this challenge has triggered interest in lightweight alternatives.

In networks of resource-constrained IoT devices, the requirements for confidentiality and authentication are mainly addressed through lightweight encryption. Traditionally, operating systems for low-end IoT devices incorporate lightweight software implementations of symmetric block ciphers, such as AES (see, *e.g.*, the security suites of Contiki OS [3] and TinyOS [4]). More recently, Systems-on-Chip (SoC) for low-end IoT devices incorporate hardware accelerators that can execute encryption primitives more efficiently than in software (*e.g.* CC1350 incorporates an AES hardware accelerator [5]).

Although very efficient (even in such resource-constrained contexts), symmetric cryptography has the inherent chicken-and-egg problem of key distribution: sharing a secret key is a prerequisite to communicating securely, yet having the means to communicate securely is a prerequisite to sharing the key. On the Internet, this challenge is addressed in a centralised manner, namely using public key infrastructure and trusted certification authorities. Wireless systems, on the other hand, largely depend on Pre-Shared Keys (PSK), *i.e.* secret keys that are shared manually during the installation of the network. PSK solves the key distribution problem but naturally does not scale well with the large number of devices in IoT networks.

An alternative to manual key installation is generating secret bit sequences on the devices using shared environmental observations that are not accessible to third parties [6]. In this paper, we propose SHAKE (SHared Acceleration Key Establishment), a scheme for generating encryption keys from shared movements. SHAKE is able to install keys without the need of transmitting them directly over a wired or wireless channel, operating as follows. Consider an IoT technician deploying a network of IoT devices. A unique pair-wise key with the IoT Gateway needs to be installed in each device. Before mounting each device, the IoT technician holds it in contact with the IoT Gateway and performs a random movement for a few seconds. SHAKE leverages the shared acceleration patterns to generate the same key in both devices.

In summary, the contributions of this paper are the following: (i) we propose SHAKE, a friendly and efficient way to install encryption keys on low-end IoT devices during deployment; (ii) we provide a proof-of-concept implementation of SHAKE that is based on the Contiki-NG operating system; (iii) we evaluate its resilience against mimic attacks; (iv) lastly, we evaluate the energy-efficiency of SHAKE in comparison to other state-of-the-art schemes for key generation.

The remainder of the paper is structured as follows. Section II briefly summarises the related work. Section III elaborates on SHAKE and its mechanics. Section IV provides implementation details and experimentally evaluates SHAKE in terms of security and energy-efficiency. Lastly, Section V provides concluding remarks.

II. Related Work

From Wi-Fi to cellular networks, wireless networks typically depend on Pre-Shared Keys (PSK) for implementing security. Although lightweight public-key infrastructure architectures have been proposed in the literature [7], [8], to date,

all major operating systems for resource-constrained wireless embedded devices primarily depend on PSK for their security services. For instance, Contiki-NG currently supports end-to-end security with PSK-based TinyDTLS [9] and link-layer security via a full implementation of the PSK-based IEEE 802.15.4 security stack [10].

PSK naturally introduces challenges in deploying IoT sensing technology at large scale. A good example is the SPHERE deployments [11]. SPHERE has deployed Health IoT sensing technology in 50+ residential properties in Bristol, UK. The security of their 700+ low-end IoT devices (*i.e.* wearables and environmental sensors) is based on pre-shared keys that need to be installed on the devices by non-expert technicians. To address the challenges of key installation, SPHERE developed a tool that automatically generates custom firmware for each device with the necessary encryption key pre-installed [12]. Yet, handling a custom firmware image for each device introduces managerial overhead, is error-prone, and reduces the flexibility of the deployment technicians.

An alternative solution is presented in [13], which proposes an RFID-based secret key installation system that introduces RFID readers on the resource-constrained devices. In turn, the deployment technician carries an RFID tag and uses it to install keys securely and without the need of wires. The disadvantage of the solution is that the RFID readers consume significant battery resources (*i.e.* 1 mA) even when deactivated [13].

A different approach to the problem of key installation is generating secret keys locally on the devices. Secret key generation techniques build on measurements of some physical property of the environment that is: (i) sufficiently random (*i.e.* high entropy), (ii) shared among the legitimate parties, and (iii) inaccessible to adversaries. The wireless channel, indeed, has these properties and has been used as a source of shared randomness for secret key generation [6]. This is based on the theory of reciprocity of electromagnetic radiation, which states that the wireless channel between two devices is symmetrical within the coherence time and distance. Hence, the devices can exchange unencrypted messages, individually measure the channel attenuation (*i.e.* the Received Signal Strength, RSS), and use the measurements to generate secret bit streams. The individually generated bit streams may contain bit errors, thus error correction techniques are employed, as well as privacy amplification techniques (*e.g.* hashing) to account for information leakage in the previous stages [6]. Examples of RSS-based secret key generation schemes for resource-constrained IoT devices can be found in [14], [15] and [16].

In [17], the authors present a key generation scheme that is based on anonymous broadcasting and source indistinguishability. The proposed scheme leverages induced movements to protect the anonymity of the broadcast packets against RSS analysis attacks. Our proposed scheme also incorporates induced movements, yet differently from [17], we use them as a source of shared randomness. Indeed, as we discuss in Section IV, [17] and the aforementioned RSS-based secret key generation schemes require extensive use of the radio unit; thus, they are less energy-efficient than our proposed scheme that uses an ultra low power accelerometer instead.

In wearables and body-area networks in general, biometrics have also been used as a source of shared randomness for secret key generation. The idea is that two body-worn (or implanted) IoT devices can generate shared secret keys from some physical property of the user's body. Walkie-Talkie [18] and BANDANA [19], for instance, generate keys based on the gait. Similarly, H2H [20] and H2B [21] generate keys based on the heartbeat. Our proposed secret key generation scheme, SHAKE, is in a similar spirit to these works, yet it is based on shared movements induced by the deployment technician and, therefore, it is not limited to body-worn sensors.

Finally, acceleration profiles have also been used to prevent information leakage in wearables [22], as well as for security in different application domains that are far less resource-constrained, including smartphone pairing [23], driver identification in smart cars [24], and smart bicycle locks [25].

III. SHAKE

In this section, we illustrate SHAKE and its operations, contextualising it in a real-world setting.

Consider an IoT technician who needs to deploy an IoT device, D, and establish an encryption key with the IoT gateway, G, using SHAKE. Like other secret key generation schemes, SHAKE is composed of five main stages: *sampling*, *quantisation*, *reconciliation*, *privacy amplification*, and *confirmation*, as illustrated in Fig. 1.

1) Sampling: The technician holds D and G together, initiates SHAKE (by, *e.g.*, clicking a button), and begins a series of random movements. In turn, D and G enable their on-board tri-axial accelerometers and collect N acceleration samples. Let us denote these acceleration matrices as $\mathbf{A}_D$ and $\mathbf{A}_G$ respectively. In these $3\times N$ matrices each raw corresponds to the acceleration on the x, y and z axis respectively, and each column corresponds to a sample. The key challenge of this stage is synchronisation of the measurements; *i.e.* better synchronisation reduces the bit mismatches at the next stage.

We highlight that SHAKE assumes that gateway can be moved during deployment and that all devices are equipped with an ultra-low-power accelerometer, such as [26].

2) Quantisation: In this stage, D and G process their respective acceleration matrices, $\mathbf{A}_D$ and $\mathbf{A}_G$, individually. The process is similar on both devices, yet for clarity, let us focus on D. Initially, D applies filtering on each of the three axes using a low-pass filter and a running mean filter. Filtering reduces the impact of high-frequency noise on bit mismatches. Next, D calculates the magnitude of the acceleration. This step is necessary because D and G will likely have their accelerometers' axes not perfectly aligned. Next, D normalises the signal. Finally, the normalised magnitude of the acceleration is quantised into a binary array. The quantisation step operates using hysteresis thresholding. In particular, two thresholds are defined: an upper and a lower threshold. When a sample is above the upper threshold, it is quantised as 1; when a sample is below the lower threshold, it is quantised as 0; the samples in between the two thresholds are quantised as

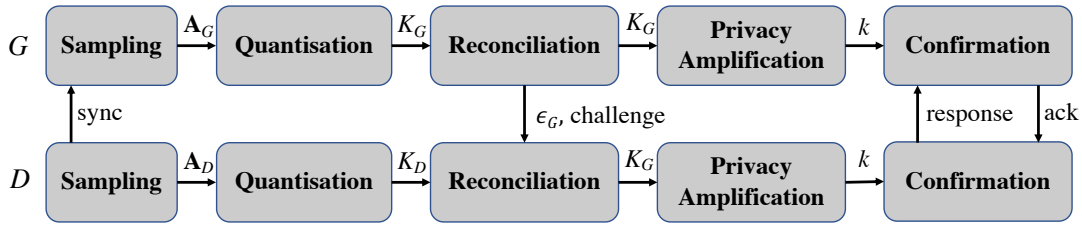

Fig. 1. **Secret key generation with SHAKE.** An IoT deployment technician establishes a secret key between a device, D, and the gateway, G, using SHAKE. Upon initialisation, the technician holds the devices together and performs random movements. The devices individually capture the movements using an on-board accelerometer (*Sampling*). Then, they quantise the raw acceleration measurements into a binary series, K_D and K_G, respectively (*Quantisation*). Discrepancies between the generated bit series are corrected (*Reconciliation*) and encryption keys are generated after privacy amplification (*Privacy Amplification*). Finally, G and D confirm that they have agreed on the same secret key, k, via a Challenge-Response mechanism (*Confirmation*).

the previous sample. The output of the quantisation stage is a binary array of size N, K_D. Similarly, G applies the same steps on $\mathbf{A}_G$ and generates K_G.

3) Reconciliation: K_D and K_G are expected to have some bit mismatches. These are due to small differences in the induced movements of the D and G, noise and synchronisation errors in sampling, and quantisation errors in processing. In the reconciliation stage, the miss-matches are corrected using error correction codes. It operates as follows. The gateway, G, generates an error correction code (ϵ_G) from K_G and transmits it to D over the wireless channel. In turn, D applies ϵ_G to K_D and flips the bits that do not match. At the end of the reconciliation stage, both D and G agree on a shared secret, namely K_G. (We note that it is equivalent if D and G changed roles during this stage.) The coding rate, R, is particularly important. On one hand, it controls how many errors can be fixed, therefore the reliability of SHAKE. On the other hand, the larger the size of the code, the more information about the shared secret leaks to a potential eavesdropper. We further investigate this trade-off in Section IV-B2.

4) Privacy Amplification: Reconciliation improves reliability at the cost of secrecy. To address this problem, we employ privacy amplification [27] using a hash function. (It is worth noting that recent systems, such as the CC1352, incorporate a SHA2 hardware accelerator.) Indeed, reconciliation reduces the number of secret bits by a ratio equal to the rate of the error correction code, R [6]. Therefore, for a key length of L bits, the samples N should be equal to $N = L/R$. The output is an encryption key, k, of high entropy that can be used as-is or as a master key.

5) Confirmation: Finally, D and G confirm that the keys are identical with a Challenge-Response (CR) mechanism. In this version, we opt for a basic HMAC-based CR mechanism that operates as follows. G generates a *nonce* (challenge) and sends it to D together with the error correction code during the reconciliation phase. In turn, D, hashes the challenge together with the generated key, k, and responds with the digest (response). G repeats the process locally and confirms that the digests match (ack). Different CR mechanisms can also be adopted to best suit every scenario [28].

TABLE I
TYPES OF MOVEMENTS

Title	Description
Simple Shake	Regular translation along the x-axis
Rough Shake	Same as *Simple Shake* but with greater force
Roller Coaster	Moving the device along a path in space
Circular Motion	Moving the device in a circle along x and y axes

IV. EVALUATION

In this section, we first prototype SHAKE and provide implementation details, then, we experimentally evaluate it, both in terms of security and energy-efficiency.

A. Proof-of-Concept Implementation

For the purposes of the evaluation, we provide a prototype implementation of SHAKE using Contiki-NG.

In particular, we interface an ADXL362 accelerometer to the launchpad CC1350 [5]. ADXL362 is an ultra-low-power accelerometer that consumes less than 2 μA when active and 10 nA on standby [26]. Although SHAKE is designed to run on the embedded device, for convenience and quick experimentation, in this implementation, we use the prototype for sampling and we do the signal processing off-board.

The accelerometer is configured with a sampling frequency of 25 Hz and a sampling range of ± 4 g. The low-pass filter is a fourth-order filter with a cut-off frequency of 2 Hz. The running mean filter has a window of 2. The upper and lower quantisation thresholds are set to 0.6 and 0.4. These configuration values are chosen empirically upon experimentation. In the reconciliation stage, we employ Reed-Solomon codes.

Using two prototypes, we collect data as follows. An individual is instructed to hold two launchpads together and execute SHAKE, *i.e.* perform random movements in order to generate a key. Specifically, we consider four types of movements, as detailed in Table I. The process is repeated 10 times for each type of movement (40 iterations in total). The user is instructed to vary the movements with each repetition.

B. Security

In this section, we evaluate the security of SHAKE. First, we define the adversary model considered, then, we describe

the mimic attack, a possible threat for SHAKE, and we discuss its impact based on experimental results.

1) Adversary Model: In order to evaluate SHAKE from the security perspective, we need to define the adversary model in scope. The scenario we consider is the same presented in Section III: an IoT technician uses SHAKE to deploy an IoT device, D, and establish an encryption key with the IoT gateway, G. The main goal of the adversary is to extract the key being established between D and G using SHAKE. To this aim, the adversary has: (i) clear line of sight of all actions and movements performed by the technician throughout SHAKE; and (ii) the possibility to eavesdrop the full communication between D and G. In other words, the attacker is in close proximity of the IoT technician during key installation and has direct sight on them.

We consider this adversary model particularly strong, as it is difficult for an attacker to be in such a privileged spot in a real-world setting, for two reasons. Firstly, SHAKE is designed for indoor deployments, as its mechanics assume that adversaries do not have physical access to the devices; in this spirit, we anticipate that SHAKE shall be used behind closed doors and far from the eyes of potential attackers. Secondly, SHAKE automatically configures the radio to the lowest transmission power setting for the transmission of the synchronisation message, error correction code, and challenge/response; given the fact that the devices are in contact during the operation of SHAKE, a higher transmission power is not only wasteful (*i.e.* unnecessary energy consumption), but it also unnecessarily increases the exposure distance to potential eavesdroppers.

Lastly, we do not address any adversary who exploits (locally or remotely) any vulnerability of the involved devices to gain the encryption key *after* it has been generated and stored on the devices, since it is out of the scope of SHAKE.

2) Mimic Attack: Given the mechanics of SHAKE and the adversary model we consider, the main security threat resides in what we call *mimic attack*. A mimic attack is an attack where the adversary, Eve, tries to extract the key established between two devices, D and G, by mimicking the motions of the deployment technician during key installation.

The mimic attack can be described as follows: (i) the legitimate key generation process, between D and G, starts while an adversary, Eve, is eavesdropping their communication; (ii) Eve mimics the motion of the deployment technician and generates the bit sequence K_E; (iii) Eve intercepts ϵ_G sent from G to D and applies it to K_E, generating the shared secret between D and G, denoted as K'_G; (iv) if K'_G is identical to K_G (*i.e.,* the legitimate secret between D and G) the mimic attack is successful, as Eve can now perform the privacy amplification function to obtain the final key, k.

3) Experimentation: We test the effectiveness of the mimic attack using our prototype. For each of the movement types detailed in Table I we perform a mimic attack, acting like an attacker that is positioned less than one meter from the technician and has a clear line of sight, as well as the ability to capture all transmitted packets. To ensure the repeatability of the results and to avoid the presence of outliers, we have performed the attack 5 times for each type of movement. Table II compares the similarity of the generated bit sequences

TABLE II
SHAKE: SIMILARITY OF GENERATED BIT-SEQUENCES

Similarity (%) of Legitimate Attempts (K_D and K_G)			
Movement Type	Mean	$\pm 95\%$ CI	
Simple Shake	85.12	82.68	87.56
Rough Shake	69.53	56.23	82.83
Roller Coaster	83.34	72.47	94.21
Circular Motion	48.31	25.87	70.74
Similarity (%) of Malicious Attempts (K_E and K_G)			
Movement Type	Mean	$\pm 95\%$ CI	
Simple Shake	51.99	37.94	66.04
Rough Shake	41.13	27.31	54.94
Roller Coaster	52.18	28.12	76.23
Circular Motion	68.57	15.36	100.0

before the stage of reconciliation. In particular, in the upper half of the table, we show the similarity of K_D and K_G, whilst in the lower half we show the similarity of K_E and K_G. The similarity is expressed as a percentage of bit matches and the table shows the mean and 95% Confidence Intervals (CI).

A key aspect that affects the success rate of a mimic attack is the maximum percentage of bits that can be fixed using the error correction code: a higher degree of error correction will make it easier to compute the right key, both for legitimate and malicious users; a lower degree of error correction will make it also harder for the legitimate users to succeed in the key generation. Indeed, the ideal level of error correction would fall above the upper bound of the malicious attempts and below the lower bound of the legitimate attempts, since it gives the adversary a small chance of randomly hitting the key while the intended user has a high chance of succeeding at the first attempt. Looking at the data, in most movement types, there is a clear separation between legitimate and malicious attempts. This means that it is possible to generate an error correction code that is able to fix legitimate errors without enabling mimic attacks. This observation holds for three of the considered movement types. Indeed, 'Circular Motion' is not characterised by sufficient separation. As a result, it is not secure against mimic attacks and, thus, it is henceforth not considered any further.

In Fig. 2, we plot the success rate of the legitimate and malicious attempts against various error correction rates for the three movement types with good separation, *i.e.,* excluding the 'Circular Motion'. Based on these results, we adopt an error correction rate that is able to fix up to 20% of the bits. In this setting, over 60% of the legitimate attempts succeed on the first attempt, while none of the mimic attacks is successful.

To conclude, although the mimic attack described in Section IV-B2 is theoretically possible, we have shown that, with a 20% of error correction, it is very unlikely to succeed. In plain words, the K_E generated by the attacker is not similar enough to K_G generated by the gateway, thus, even using the eavesdropped information ϵ_G is not sufficient for the correct generation of the shared secret, K_G.

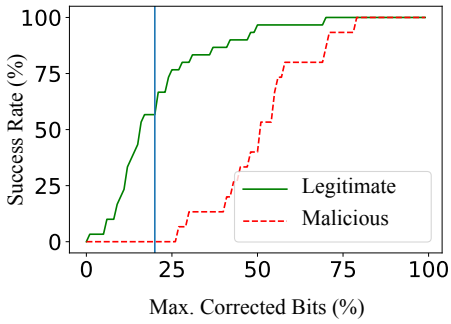

Fig. 2. Percentage of successful legitimate key generation attempts (in green) and mimic attacks (in red) against the maximum percentage of bits that the code can correct.

C. Energy Consumption

To evaluate the energy consumption of SHAKE, we have identified the consumption of sampling, communication, and processing using software-based estimation [29]. In particular, we have monitored the CPU-cycles of each stage using the Contiki-NG tool `energest` [30]. In turn, we have approximated the energy consumption by combining the time measurements with current measurements.

Based on the conclusion of Section IV-B3, we consider an error coding rate that can correct up to 20% of the generated bits. Hence, for L secret bits after the error correction, we generate N bits from $N = L/0.8$ samples [6].

Since the proof-of-concept implementation performs only data sampling on-board, an auxiliary program is written to estimate the power usage of processing. This program performs data processing on the CC1350 in a loop. An analogous python script is written to accompany it. Both are then executed and the timing differential is used as runtime scalar, such that the on-board processing time can be estimated. The python code written to handle the data processing off-board is then timed and the scalar is applied to estimate the on-board running time.

The energy consumption of communication assumes the use of the IEEE 802.15.4 TSCH (Time-Slotted Channel Hopping) protocol [31] at 2.4 GHz, and that the packet transmissions are acknowledged at the link layer. Given that the two devices are in close proximity during the key generation, we use the lowest transmission power setting (*i.e.*, -9 dBm) to limit the exposure to potential eavesdroppers. Finally, we assume the maximum packet size (*i.e.*, 98 bytes of payload). The estimates are based on measurements from [32] and [31].

Table III summarises the energy cost for generating a key of $L = 128$ bits (*i.e.* $N = 160$). Each stage was measured 10 times and the mean is used in the estimation. The standard deviation of all measurements was less than 2% of the mean. Furthermore, Table IV shows the estimates of the energy required for generating keys of 192 and 256 bits as well.

Finally, Table V compares SHAKE to other proposed methods for generating symmetric 128-bit keys in energy-constrained devices. In terms of energy consumption, SHAKE performs better than the other schemes; SHAKE needs 77% of the energy required by the low-power configuration of SKYGlow [16] that ranks second. This is primarily due to the low usage of the radio unit. The energy consumption of [14] and [15] are taken from [16]. H2B [21] uses 20.6 mJ for sampling and 83.8 mJ for processing (Samsung Smartwatch), resulting in 104.4 mJ in total. The table does not include [17] because the authors do not provide energy measurements. Yet, [17] requires the transmission and reception of one packet per two generated bits (*i.e.* 64 packets for a 128-bit key) per device. In contrast, SHAKE requires the transmission and reception of just 2 packets per device. Overall, we note that the comparison is only indicative as the adopted hardware and communication protocol also affect the energy needed to generate a key. Lastly, we also note that the various schemes have important qualitative differences that must also be considered (*e.g.* H2B only works on body sensors).

TABLE III
SHAKE: ENERGY BREAKDOWN FOR GENERATING A 128-BIT KEY

Stage	Time (ms)	Current (mA)	Energy (mJ)
Sampling	472	4	6.24
Tx (2x pkts)	20	5.39	0.36
Rx (2x pkts)	20	4.97	0.33
Processing	1816	4	23.97
Total	2329	-	30.9

TABLE IV
SHAKE: ENERGY CONSUMPTION AGAINST KEY LENGTH

Key Length (bits)	Time (s)	Energy (mJ)
128	2.33	30.9
192	3.46	45.9
256	4.62	61.1

TABLE V
ENERGY COMPARISON FOR GENERATING A 128-BIT KEY

Key Generation Scheme	Energy (mJ)
Ali *et al.* [14] (estimated by [16])	161.8
Li *et al.* [15] (estimated by [16])	79.8
SKYGLOW.a (Indoor, 2.4 Ghz) [16]	36.3
Walkie-Taklie (Moto E2) [18]	85.4
H2B (Samsung Smartwatch) [21]	104.4
SHAKE	**30.9**

V. CONCLUSION AND FUTURE WORK

Lightweight security for resource-constrained IoT systems largely depends on symmetric block ciphers and pre-shared keys. This paper has presented SHAKE, a means to install encryption keys on IoT devices during deployment. SHAKE generates keys from acceleration measurements that capture random movements performed by the deployment technician. The process incorporates a quantisation stage, a reconciliation stage for error correction, and a privacy amplification stage for increasing the entropy and addressing information leakage.

We have provided a proof-of-concept implementation of SHAKE for the CC1350 system using Contiki-NG. Using

the prototype implementation, we have generated keys from several types of movements, such as simple shake, rough shake, roller coaster, and circular motion. In addition, we have performed a series of mimic attacks whereby an adversary with a clear line of sight and ability to capture all the transmitted packets attempts to replicate the movements of the deployment technician from a short distance. The experimental results, used to fine-tune the reconciliation stage, have demonstrated that there is sufficient separation between the legitimate and adversarial generated bit streams that allows for the correction of up to 20% of the bits. This applies to all but the circular motion, which is not recommended for usage.

Finally, we have measured the energy required for generating keys using SHAKE and compared it against other key generation schemes in the literature. The results suggest that SHAKE is more energy-efficient than the alternatives; yet, we appreciate that the employed hardware and networking protocols also have a significant impact on energy consumption.

The performance of SHAKE can be further improved in future work with better time synchronisation. Indeed, a time-synchronous communication protocol, such as IEEE 802.15.4 TSCH, can provide microsecond level synchronisation between the clocks of the devices [33]. Yet, the ADXL362 accelerometers do not operate using the device clock; instead, they use their own embedded clocks. For better synchronisation, sampling should be triggered externally, from a pulse-wave generated by CC1350. At the cost of additional energy consumption, this approach would improve the similarity of the generated signals, enabling the generation of more than one bits per sample during quantisation.

REFERENCES

[1] S. Ravi, S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 461–491, Aug. 2004.

[2] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating systems for low-end devices in the internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 720–734, Oct 2016.

[3] L. Casado and P. Tsigas, "ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System," in *Identity and Privacy in the Internet Age*, 2009, pp. 133–147.

[4] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," in *Proc. 2nd Int. Conf. Embedded Networked Sensor Systems*. ACM, 2004, pp. 162–175.

[5] Texas Instruments. LaunchPad CC1350. [Online]. Available: http://www.ti.com/tool/LAUNCHXL-CC1350

[6] Y. E. H. Shehadeh and D. Hogrefe, "A survey on secret key generation mechanisms on the physical layer in wireless networks," *Security and Communication Networks*, vol. 8, no. 2, pp. 332–341, 2015.

[7] R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," in *Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2004.

[8] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proc. 1st IEEE Int. Conf. Sensor and Ad Hoc Communications and Networks (SECON)*. IEEE, 2004, pp. 71–80.

[9] O. Bergmann, S. Gerdes, and C. Bormann, "Simple keys for simple smart objects," in *Workshop on Smart Object Security*, 2012.

[10] N. Sastry and D. Wagner, "Security considerations for ieee 802.15.4 networks," in *Proc. 3rd ACM Workshop on Wireless Security*, 2004.

[11] A. Elsts, X. Fafoutis, P. Woznowski, E. Tonkin, G. Oikonomou, R. Piechocki, and I. Craddock, "Enabling Healthcare in Smart Homes: The SPHERE IoT Network Infrastructure," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 164–170, 2018.

[12] X. Fafoutis, A. Elsts, G. Oikonomou, and R. Piechocki, "SPHERE Deployment Manager: A Tool for Deploying IoT Sensor Networks at Large Scale," in *Int. Conf. Ad-Hoc Networks and Wireless*. Springer, 2018, pp. 307–318.

[13] M. Cetinkaya, J. Dede, and A. Förster, "An RFID Based Secure Key and Configuration Distribution for Contiki," in *Proc. 2018 Int. Conf. Embedded Wireless Systems and Networks*, 2018, pp. 189–190.

[14] S. T. Ali, V. Sivaraman, and D. Ostry, "Zero reconciliation secret key generation for body-worn health monitoring devices," in *Proc. 5th ACM Conf. on Security and Privacy in Wireless and Mobile Networks*. ACM, 2012, pp. 39–50.

[15] Z. Li, Q. Pei, I. Markwood, Y. Liu, and H. Zhu, "Secret key establishment via RSS trajectory matching between wearable devices," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 802–817, 2017.

[16] G. Margelis, X. Fafoutis, G. Oikonomou, R. Piechocki, T. Tryfonas, and P. Thomas, "Efficient DCT-based secret key generation for the Internet of Things," *Ad Hoc Networks*, vol. 92, 2019.

[17] C. Castelluccia and P. Mutaf, "Shake them up!: A movement-based pairing protocol for cpu-constrained devices," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. ACM, 2005, pp. 51–64.

[18] W. Xu, G. Revadigar, C. Luo, N. Bergmann, and W. Hu, "Walkie-Talkie: Motion-Assisted Automatic Key Generation for Secure On-Body Device Communication," in *Proc. 15th ACM/IEEE Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2016, pp. 1–12.

[19] D. Schrmann, A. Brsch, S. Sigg, and L. Wolf, "BANDANA – Body area network device-to-device authentication using natural gAit," in *Proc. IEEE Int. Conf. Pervasive Computing and Communications (PerCom)*, 2017, pp. 190–196.

[20] M. Rostami, A. Juels, and F. Koushanfar, "Heart-to-heart (H2H): authentication for implanted medical devices," in *Proc. ACM SIGSAC Conf. Computer & Communications Security*. ACM, 2013, pp. 1099–1112.

[21] Q. Lin, W. Xu, J. Liu, A. Khamis, W. Hu, M. Hassan, and A. Seneviratne, "H2B: heartbeat-based secret key generation using piezo vibration sensors," in *Proc. 18th Int. Conf. Information Processing in Sensor Networks (IPSN)*. ACM, 2019, pp. 265–276.

[22] X. Fafoutis, L. Marchegiani, G. Z. Papadopoulos, R. Piechocki, T. Tryfonas, and G. Oikonomou, "Privacy leakage of physical activity levels in wireless embedded wearable systems," *IEEE Signal Processing Letters*, vol. 24, no. 2, pp. 136–140, 2017.

[23] R. Mayrhofer and H. Gellersen, "Shake well before use: Intuitive and secure pairing of mobile devices," *IEEE Trans. Mobile Computing*, vol. 8, no. 6, pp. 792–806, 2009.

[24] L. Marchegiani and I. Posner, "Long-term driving behaviour modelling for driver identification," in *21st Int. Conf. Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 913–919.

[25] A. Arno, K. Toyoda, and I. Sasase, "Accelerometer assisted authentication scheme for smart bicycle lock," in *Proc. IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 520–523.

[26] Analog Devices. ADXL362. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl362.pdf

[27] C. H. Bennett, G. Brassard, C. Crepeau, and U. M. Maurer, "Generalized privacy amplification," *IEEE Trans. Information Theory*, vol. 41, no. 6, pp. 1915–1923, 1995.

[28] G. Ginesu, M. L. Lobina, and D. D. Giusto, "Property Protection and User Authentication in IP Networks through Challenge-Response Mechanisms: Present, Past and Future Trends," in *Inform. Tech. Intellectual Property Protection: Interdisciplinary Advancem.*, 2012, pp. 133–163.

[29] P. Hurni, B. Nyffenegger, T. Braun, and A. Hergenroeder, "On the accuracy of software-based energy estimation techniques," in *Wireless Sensor Networks*. Springer, 2011, pp. 49–64.

[30] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based online energy estimation for sensor nodes," in *Proc. 4th Workshop on Embedded Networked Sensors*, 2007, pp. 28–32.

[31] S. Duquennoy, A. Elsts, B. Al Nahas, and G. Oikonomou, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *13th Int. Conf. Distributed Computing in Sensor Systems (DCOSS)*, 2017.

[32] X. Fafoutis, E. Tsimbalo, W. Zhao *et al.*, "BLE or IEEE 802.15.4: Which Home IoT Communication Solution is more Energy-Efficient?" *EAI Endorsed Trans. Internet of Things*, vol. 2, no. 5, 12 2016.

[33] A. Elsts, S. Duquennoy, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "Microsecond-accuracy time synchronization using the ieee 802.15.4 tsch protocol," in *Proc. IEEE 41st Conf. Local Computer Networks Workshops (LCN Workshops)*, 2016, pp. 156–164.