SAPHE - Simple Accelerometer based wireless Pairing with HEuristic trees

Bogdan Groza
Faculty of Automatics and Computers
Politehnica University of Timisoara
Bd. V. Parvan nr. 2, Timisoara, Romania
bogdan.groza@aut.upt.ro

Rene Mayrhofer
Department for Mobile Computing
University of Applied Sciences Upper Austria
Softwarepark 11, Hagenberg, Austria
rene.mayrhofer@fh-hagenberg.at

ABSTRACT

Accelerometers provide a good source of entropy for bootstrapping a secure communication channel in autonomous and spontaneous interactions between mobile devices that share a common context but were not previously associated. We propose two simple and efficient key exchange protocols based on accelerometer data that use only simple hash functions combined with heuristic search trees. Using heuristics such as the Euclidean distance proves to be beneficial as it allows a more effective recovery of the shared key. While the first protocol seems to give just some performance improvements, the second, which we call *hashed heuristic tree*, is more secure than previous proposals since it increases the difference in protocol execution between benign and malicious parties. Nevertheless, the *hashed heuristic tree* is an entirely new approach which has the advantage of allowing different heuristics in the search, leaving plenty of room for future variants and optimizations.

Categories and Subject Descriptors

D.4.6 [**OPERATING SYSTEMS**]: Security and Protection—*Authentication*; K.6.5 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection—*Authentication*

General Terms

Security

Keywords

authentication, accelerometer, key-exchange

1. MOTIVATION

While today's mobile devices embed countless applications that boost up usability, they still heavily rely on manual input for performing day by day routines such as pairing, i.e., bootstrapping a communication channel. Examples for such manual input are the PIN code that needs to be entered for Bluetooth pairing, the WPS password for connecting to many WiFi networks, or a shared password for many secure chat, telephone, or file transfer applications. Depending on user data not only reduces usability, but inextricably gambles security since such input usually lacks the desired entropy for making a connection secure (as evidenced by the standard pairing PIN codes '0000' or '1234' for many Bluetooth headsets).

In contrast, using environmental information is an attractive mechanism for secure pairing between devices in autonomous and spontaneous interactions where previously shared information such as public key certificates does not exists. While manually entered information is time consuming and requires additional input interfaces, using motion, sound, light, etc., seems to be an attractive alternative that provides a fast and more scalable solution that can be also used by unskillful users. A complete survey on different techniques that can be used for this purpose can be found in [17].

In particular, motion characteristics acquired from accelerometers are considered to be a rich source of entropy, easy to produce or re-produce and difficult to guess by outsiders. Mayrhofer and Gellersen explore the use of shaking patterns to pair mobile devices in [22] and apply this technique to secure a bluetooth connection. Bluetooth pairing is known to be particularly vulnerable [13] to man-in-the-middle attacks caused by external adversaries and shaking patterns can be used to remove this problem. The same idea is also explored by Castelluccia and Mutaf in [5], Kirovski et al. in [16] and by Bichler et al. in [3]. All methods are distinct in the way key extraction from sensor data is performed. More specifically, in [22] data is extracted in the frequency domain by using a coherence function, initially used in [18] while the other two proposals extract data in the time domain. Our approach also uses the time domain but the way in which we extract data by using heuristic trees and hashed heuristic trees appears to be entirely distinct from any previous procedure. The closest method to our extraction procedure could be [3], but the concept of hashed heuristic tree that we introduce seems to be entirely new.

In a non-adversarial setting, accelerometer data can be directly used to distinguish whether or not two devices are carried together [14], [18], to recognize activities and gestures [15], [24], [19], [1] or an individual gait [2], [11], [10] by simply exchanging the information acquired from sensors. But it is not straight-forward to establish a common secret

MoMM2012, 3-5 December, 2012, Bali, Indonesia

shared key from such data. The difficulty comes from the fact that information is measured on different sensors from different devices, and thus it is not 100% identical while the communication channel is (yet) insecure. This is distinct to the classical use of acceleration patterns in biometry where the communication channel between the sensor and the matcher is usually assumed to be secure. Therefore, a secure way to compare information and to extract entropy is needed. In [22] two protocols are presented: ShaVe and ShaCK. The first protocol is based on the Diffie-Hellman-Merkle key exchange. By using such public-key primitives it is easy to protect the feature vectors extracted from accelerometers even if they have low entropy. It is commonly known that trapdoor permutations can be used to construct protocols that are resilient to guessing, extensive theoretical results on protecting low entropy secrets in key agreements are available from Goldreich and Lindell [12]. However, public key primitives are by far more computational intensive and may be unsuitable for devices that merge in spontaneous interactions since usually these devices have low computational power, small energy resources, etc. For this purpose a second protocol based entirely on hash functions is proposed in [22]. This protocol is based on a previous proposal called the Candidate Key Protocol (CKP) [20]. In brief, CKP works as follows: each feature vector v_i obtained from the accelerometer is hashed along some random salt value s_i (to prevent a dictionary attack) and sent as $H(v_i, s_i)$ to the other party which searches in its own extracted feature vectors to find a match. If this succeeds, then the vector is added to the corresponding key pool (which is, upon finding a sufficient number of matches, hashed again to compute the shared secret key), otherwise it is dropped.

Contribution in a nutshell. We devise a protocol that can be used to securely exchange low entropy vectors without relying on classical alternatives such as fuzzy cryptography or expensive public-key primitives. The issue that we try to improve is the following: if the hashed feature vectors extracted from the accelerometer are too small, then one can mount an exhaustive search attack while if they are too large the probability that they are dropped increases due to the inability of the second party to find a match. We improve on this as follows. To make it suitable to exchange larger vectors we allow the protocol to use a heuristic search in order to find a match. That is, we use a function to evaluate the probability that a particular value was recorded by both parts. Note however that such a search over regular hashes can also be mounted by an adversary, thus, interestingly enough, the potential improvement of the protocol may be viewed up to some point as a new potential attack. To fix this, we propose a much stronger variant in which heuristics are embedded in the search tree by each party, thus forcing the potentially malicious party to perform in the same manner as the other party (potentially honest). If the pattern matches on both sides, then this gives a higher chance of success. We call these two proposals *heuristic tree* and *hashed heuristic tree*. As heuristic we simply used the Euclidean distance between the extracted bits and some challenge, but we believe there is room for future work as other suitable heuristics can be found. In the experimental section we tested the approach by using two WiiMotes and the results were satisfactory.

2. THE PROTOCOLS

On an abstract level, both protocols have a commitment-challenge-response nature. The steps of both protocols are depicted in Fig. 2. We use standard notations: A, B are the participant identities, also used as subscripts for values to associate them to a participant identity. All notations are summarized in Fig. 1. Thus, in the first stage, both participants commit the future challenges – which take the form of vectors of random threshold values – by exchanging their hashes. Afterwards, data is collected from the accelerometers. In the second stage, these challenges are revealed and in the third stage the responses are computed. If the responses match, then authentication succeeds, otherwise the protocol is dropped, but can be subsequently restarted.

A, B	protocol participants (also used as subscripts to values originating on their sides)
r	randomly generated value
$\overline{v}$	values extracted from accelerometers
$\overline{t}$	threshold to which accelerometer values are compared (thresholds need not be exchanged as they are derived from the commitments)
k	extracted key
$\mathcal{T}$	algorithm used to derive thresholds from random material
$\mathcal{E}$	extraction algorithm used to extract key bits by comparing accelerometer values to threshold values
$\mathcal{O}$	ordering algorithm used in the hashed heuristic tree
H, E	standard hash and encryption functions

Figure 1: Summary of notations

By using the commitment stage, we force the challenge to be established before the features are actually extracted, thus making it impossible for an adversary to select a more convenient challenge (if the challenge is released after data is collected) or to alter data collection in such way that it will be easier to answer the challenge (if the challenge is known before data is collected). Such an attempt may be feasible on CKP since a man-in-the-middle (MITM) adversary can simply detect low entropy vectors by exhaustive search and then replace high entropy vectors with random values making them fail as parts of the key pool. Subsequently, authentication will rely only on weak vectors vectors.

2.1 First Variant

To summarize the proposed key extraction procedure, Fig. 3 shows data collected from two accelerometers and the randomized values used as the challenge threshold. In order to extract one bit of the key, accelerometer data is compared

Device (A)		Device (B)
	Commitment Stage	
1. $r_A \in_R 1^k$		
	$\xrightarrow{H(r_A)}$	
2.		$r_B \in_R 1^k$
	$\xleftarrow{H(r_B)}$	
	Generate Accelerometer Data (Shaking)	
	Challenge-Response Stage	
3. $N_A \in_R 1^k$, $\bar{t}_A \leftarrow \mathcal{T}(r_A)$ $\bar{o}_A \leftarrow \mathcal{O}(\bar{v}_A, \bar{t}_A)$, $k_A \leftarrow \mathcal{E}(\bar{v}_A, \bar{t}_A, \bar{o}_A)$ $c_A = E_{k_A}(N_A)$		
	$\xrightarrow{r_A, \bar{o}_A, c_A}$	
4.		$N_B \in_R 1^k$, $\bar{t}_B \leftarrow \mathcal{T}(r_B)$ $\bar{o}_B \leftarrow \mathcal{O}(\bar{v}_B, \bar{t}_B)$, $k_B \leftarrow \mathcal{E}(\bar{v}_B, \bar{t}_B, \bar{o}_B)$ $c_B = E_{k_B}(N_B)$, $k'_A \leftarrow \mathcal{E}^{-1}(\bar{v}_B, \bar{t}_A, \bar{o}_A)$ $N'_A = E^{-1}_{k'_A}(c_A)$
	$\xleftarrow{r_B, \bar{o}_B, c_B, N'_A}$	
5. if $N_A = N'_A$ then k_A was correctly extracted $k'_B \leftarrow \mathcal{E}^{-1}(\bar{v}_A, \bar{t}_B, \bar{o}_B)$ $N'_B = E^{-1}_{k'_B}(c_B)$		
	$\xrightarrow{N'_B}$	
6.		if $N_B = N'_B$ then k_B was correctly extracted

Figure 2: Modified protocol for the case of hashed heuristic tree

to the value of the threshold and the bit is set to 0 if the accelerometer value is less or equal and 1 otherwise.

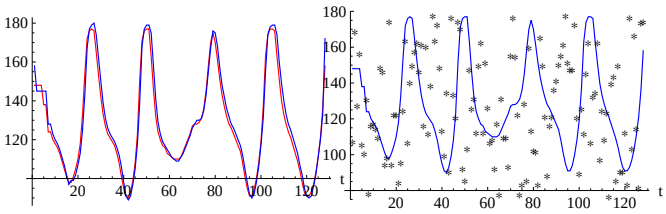

Figure 3: Data collected from two accelerometers shaken together (left) and the challenge threshold (right)

For the first protocol variant, the ordering algorithm $\mathcal{O}$ and its output $\bar{o}$ from Fig. 2 will be ignored. The protocol works as follows. First, the commitments are exchanged between the participants, random values r are generated on each side and their hashes exchanged. Second, after accelerometer data is generated by shaking, challenges are computed. To compute the challenges and responses the following publicly known algorithms, that we depict in calligraphic style, are used: $\mathcal{T}$ is the algorithm used to generate a threshold to which the vector items are compared (in particular a deterministic PRNG which receives random r as a seed can be used), $\mathcal{E}$ is the key extraction algorithm which extracts each key bit by comparing the values obtained from the accelerometer to the corresponding value from the thresholds and $\mathcal{E}^{-1}$ is the inverse of the extraction algorithm which tries to recover the key. Thus, the thresholds are extracted from the committed random values, i.e., $\bar{t} \leftarrow \mathcal{T}(r)$, then the key is extracted, i.e., $k \leftarrow \mathcal{E}(\bar{v}, \bar{t})$ and finally the challenge is computed, i.e., $c = E_k(N)$. To answer the challenge, the nonce is recovered by computing $k' \leftarrow \mathcal{E}^{-1}(\bar{v}, \bar{t})$ and then decrypting $N' = E^{-1}_{k'}(c)$.

The extraction algorithm $\mathcal{E}^{-1}$ works as follows. Instead of simply extracting the bits by comparing the threshold with the accelerometer value, the algorithm produces a search tree by placing the extracted bits in a list and sorting them in a decreasing order of the heuristic function for which we use the Euclidean distance $\Delta(x, y) = |x - y|$. This will lead to a search tree which has closest to the root all those values which are more distant from the threshold and should have better chances for being identically measured on both sides. The reason is that accelerometer values that are more distant from the threshold have a bigger chance to give the same binary output on both sides.

To explain why this is the case, consider the case of one

random threshold value fixed to t_i and the corresponding accelerometer value set to $v_{A,i}$ and $v_{B,i}$ on each side. To extract the same bit on each side, the sign of the distance between values and thresholds has to be the same on each side, i.e., $sgn(v_{A,i} - t_i) = sgn(v_{B,i} - t_i)$. Note that this happens with probability:

$$\Pr[v_{A,i} > t_i \wedge v_{B,i} > t_i] + \Pr[v_{A,i} \leq t_i \wedge v_{B,i} \leq t_i] =$$
$$= 1 - \Pr[v_{A,i} \leq t_i \wedge v_{B,i} > t_i] - \Pr[v_{A,i} > t_i \wedge v_{B,i} \leq t_i].$$

But $\Pr[v_{A,i} > t_i \wedge v_{B,i} \leq t_i] \leq \Pr[\Delta(v_{A,i}, v_{B,i}) > \Delta(v_{A,i}, t_i)]$ since if the values from A and B are on opposite sides of the threshold then the distance between them must be bigger than the distance between any of the values and the threshold. Now as can be seen in Fig. 4, the probability that two accelerometers shaken together drift with a certain distance, i.e., $\Pr[\Delta(v_{A,i}, v_{B,i}) > \Delta(v_{A,i}, t)]$, drops almost exponentially with the distance and so must $\Pr[v_{A,i} > t_i \wedge v_{B,i} < t_i]$. The same holds for the second term $\Pr[v_{A,i} \leq t_i \wedge v_{B,i} \leq t_i]$ and this completes our intuition. To avoid locking $\mathcal{E}^{-1}$ when there are too many elements that do not match, the algorithm must be forced to stop after a predefined number of steps (for example 1000 hashes in our experimental results). An authenticated encryption algorithm is used for E such that $\mathcal{E}^{-1}$ can stop precisely when the correct response is reached.

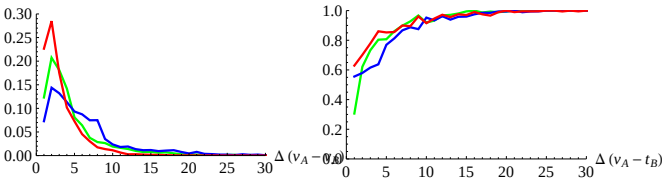

Figure 4: Drift probability from two WiiMotes shaken together (left) and success probability for different thresholds (right)

2.2 Second Variant

As already stated, the search over regular hashes can also be mounted by an adversary. While this will not help much if the entropy is high enough, it can be considered a new potential attack, to best of our knowledge not analysed in related proposals.

We use a small example to clarify our intentions with the hashed heuristic tree. In the first protocol variant one bit is extracted by comparing accelerometer data to the corresponding threshold value. In the second variant we proceed in the same way but with two modifications. First we sort the bits in decreasing order of the distance to the threshold, i.e, $\Delta(v_i, t_i)$. Second, we do a bit-by-bit hashing as suggested in Fig. 5 which illustrates a hashed heuristic tree. To build the hashed heuristic tree we use a fresh random value r (r_A in the case of A and r_B in the case of B) to avoid a precomputed dictionary attack on the hashed tree. Now the hash of r forms the root of the tree.

For example, consider the following three random values returned by A's accelerometer $v_A = \{17, 31, 7\}$ and a three value random threshold $t_A = \{10, 17, 37\}$. The corresponding key bits are $k_A = \{1, 1, 0\}$ and by subtracting the vectors we get the distance $\Delta(v_A - t_A) = \{7, 14, 30\}$ which means that the key bits are reordered before hashing to $\tilde{k}_A = \{0, 1, 1\}$ (note the reverse order). Getting back to Fig. 5, A will follow the path that is marked with continuous line and extract the key k_{011}. Now consider that B's accelerometer collected $v_B = \{8, 29, 9\}$, indeed the key bits for B are $k_B = \{0, 1, 0\}$ but the distance $\Delta(v_B - t_A) = \{2, 12, 28\}$ leads to the same order as in the case of A placing the mismatched first bit as the last one and accordingly at a leaf node in the search tree. This makes B end up with k_{010} which makes it spend only one additional hash to recover the correct key k_{011}. In comparison, if the bit-by-bit hashing would not be performed in decreasing order of the distance, then instead of one additional hash, B would have needed four more hashes to get from k_{010} to k_{110}. This is because of the aforementioned reason: key bits that are more distant to the threshold have a higher chance to be identical on both sides. Thus, if data is indeed collected from accelerometers shaken together, the hashed heuristic tree constructed by one participant will also provide a fast key recovery for the other participant. Note that an adversary cannot decide on the order in which they choose the values to extract the key bits and the heuristic search tree in the previous protocol cannot be used by the adversary. This leads to a more secure variant and we will define its exact security bound in a forthcoming section.

The protocol is outlined in Fig. 2. Now the ordering algorithm $\mathcal{O}$ is used to sort the values according to heuristic h and modified the extraction algorithm. $\mathcal{E}$ and $\mathcal{E}^{-1}$ are used as previously to extract the bits and compute the key, but this time each bit is hashed along with the hash of the previous bit and so on in a predefined order $\overline{o}$ as described above.

3. EXPERIMENTAL RESULTS

As a primary testbed we used two Nintendo Wii Remotes, also known as WiiMotes, which are wide spread devices, easy to use and program. To test the protocol even further, we used the shaking data sets from the Open Source Ubiquitous Authentication Toolkit (`http://www.openuat.org/datasets`). These sets contain a rich amount of shaking values from different individuals but the similarity of the data recorded by the sensors is weaker compared to WiiMotes (notably, the experimental setup was made from off-the-shelf components before WiiMotes reached the market).

3.1 Nintendo WiiMotes

To test the efficiency of the protocol we used two WiiMotes connected via Bluetooth to a notebook. The bit-rate taken from all three axes of the accelerometer is around 256 bits per second which come at a sample rate of around 80–90 samples per second on each of the three axes. Each sample adds 1 bit of entropy to the key by comparing it to the threshold, but indeed the resolution of accelerometer data is around 7 bits (the WiiMotes return a float which we convert to a byte but the value is roughly in the interval $[-64, 64]$). For security reasons, it should be preferable to take a majority function of the protocol outcome from several runs during a fixed time period (for example 5-10 seconds).

Figure 6 provides the success rate taken for keys of 32, 64, and 128 bits extracted with hashed heuristic trees and a search depth limited to 128, 256, 512 and 1024 hashes. Obviously, the success rate increases with the depth limit, but variations are possible since this greatly depends on the way in which accelerometers are shaken by the user. The results are somewhat non-uniform on the three axes and

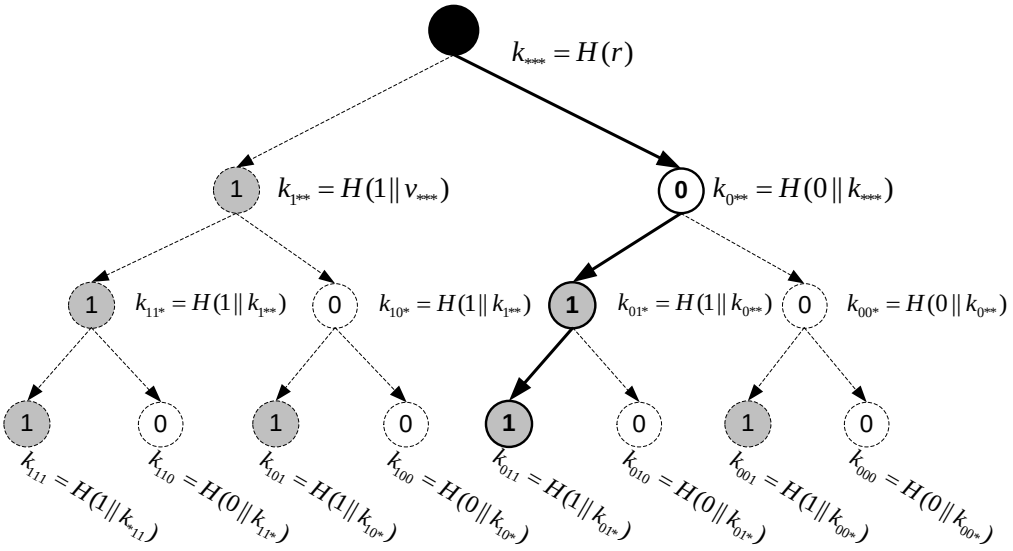

Figure 5: Hashed heuristic tree

this is mostly due to the tendency of individuals to shake devices mostly vertically. The success rate is very high for 32 bits and even for 64 bits if 1024 hashes are allowed the success rate is generally over 75%. However, 32 or 64 bits may be insufficient to be used directly as a symmetric session key. Extracting larger keys of 128 bits is also feasible with good probability if 256–1024 hashes are allowed. If only 128 hashes are allowed in the case of 128 bits, then the success probability is close to 0 (a successful result would require at most 7 out of the 128 bits to be different) but if 256–1024 hashes are allowed then a success probability of 20%–60% is achieved. For the case when only short keys can be extracted, they can be safely used with asymmetric cryptography in an authenticated key establishment protocol, in particular we propose to use these key bits as a (short) shared secret in the confidential short input variant of UA-CAP [21], which then allows a secure, non-interactive key establishment based on this common input.

3.2 Shaking Data from OpenUAT

Indeed the similarity of the data is not as high as in the case of WiiMotes. One reason is that the sensors are not spatially aligned as accelerometers were mounted inside ping-pong balls. Figure 7 shows data collected on the three axes (i), (ii), (iii); it is easy to note that there are significant drifts on the second and the third axes. After carefully merging the three axes by computing the arithmetic mean, the values recorded by the two sensors become more similar as can be seen in Figure 7 (iv). On this particular data the protocol succeeds, but unfortunately there are many samples that are less correlated than this.

The tests done over the Experiment 1 data sets from OpenUAT showed a success rate of around 20%–40% for a key of 64 bits at 1024 hashes. This is significantly reduced compared to the case of WiiMotes. While not efficient from a practical point of view, statistically this still means that

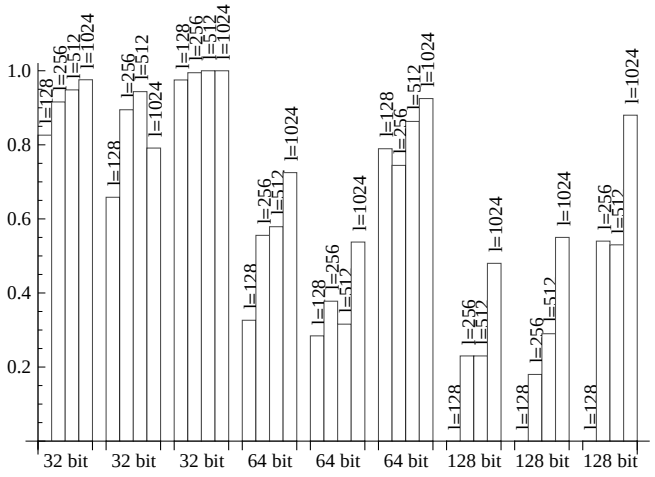

Figure 6: Success rate for 32, 64, and 128 bit keys with search depth limited to 128, 256, 512 and 1024 hashes

the protocol can be used to pair such devices. But indeed, using the coherence function and the protocol from [22] is much more reliable (this at the cost of more computational power). Finally, the coherence function can be embedded in the hashed heuristic tree proposed here and this should increase the reliability of the protocol.

4. SECURITY ANALYSIS, COMPARISON AND FURTHER IMPROVEMENTS

We make a brief security analysis of the proposed method in order to derive a security bound. Then we proceed to compare this proposal with protocols from related work in-

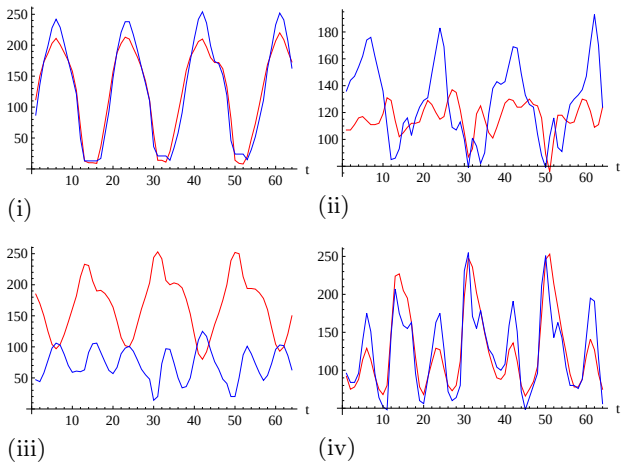

Figure 7: Example of data collected on the three axes (i), (ii), (iii) and the result after preprocessing (iv)

cluding fuzzy cryptography. We find this method to be complementary to related techniques in the sense that hashed heuristic trees can be added on top of the other protocols. We also enumerate some improvements that can be added to our proposal.

4.1 Security and efficiency considerations

For the proposed protocols it is possible to find both the security and efficiency bounds given the vector of probabilities $p =< p_0, p_1, p_2, ..., p_{d-1} >$ where $p_i, i \in [0..d-1]$ is the probability of honest client (or of the adversary if the security level is to be determined) to get the right bit at step i. Here $d-1$ represents the maximum depth of the search tree and it corresponds to the bit-length of the extracted key. Due to the key extraction procedure, in both protocol variants, a search tree is obtained where leaf nodes correspond to one particular value of the key (see Fig. 8 for an example). On each branch of the three, p_i denotes the probability that the bit is correctly identified.

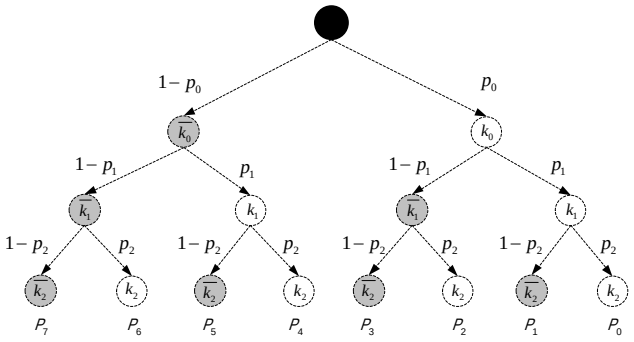

Figure 8: Heuristic search tree

For the general case, we have a search tree of depth d with probabilities $p_0, p_1, p_2, ..., p_{d-1}$ associated to each level from root to leaves. Let $\mathcal{P}_i$ denote the product of probabilities on the i-th path, e.g., $\mathcal{P}_0 = p_0 \cdot p_1 \cdot p_2 ... \cdot p_{d-1}$, $\mathcal{P}_1 = p_0 \cdot p_1 \cdot p_2 ... \cdot (1-p_{d-1})$, etc. For example in the case of the search tree from Figure 8 we have $\mathcal{P}_0 = p_0 \cdot p_1 \cdot p_2$, $\mathcal{P}_1 = p_0 \cdot p_1 \cdot (1-p_2)$, $\mathcal{P}_3 = p_0 \cdot (1-p_1) \cdot p_2$, etc. Each leaf node corresponds to a particular value of the key and the probability that the key is located in the i-th leaf is equal to the product of the path that leads to the leaf and which is $\mathcal{P}_i$. Since the leaf nodes cover all possible values for the key we have $\sum_{i=0}^{i=2^d-1} \mathcal{P}_i = 1$, i.e., the key is certainly in one of the leaf nodes. The minimum number of steps of the searching algorithm is d in the case when the key is located in the rightmost leaf and $d+2^d-1$ in the worst case when the key is in the leftmost leaf node. By definition, the average solving time of the search algorithm is the weighted average of the possible number of steps, i.e.,

$$T = \sum_{i=0}^{i=2^d-1} (d+i) \cdot \mathcal{P}_i = \underbrace{d \cdot \sum_{i=0}^{i=2^d-1} \mathcal{P}_i}_{=d} + \sum_{i=0}^{i=2^d-1} i \cdot \mathcal{P}_i$$

Now let $\mathcal{P}_i^{(n)}$ denote the path to leaf i cut at level $d-n$. Note that in the general case if we take any pair of leaf nodes that descend from the same parent (e.g., in Fig. 8 we have $(\mathcal{P}_0, \mathcal{P}_1)$, $(\mathcal{P}_2, \mathcal{P}_3)$, etc.) we have $\forall i \in [0..2^{d-1}-1]$, $2 \cdot i \cdot \mathcal{P}_{2 \cdot i} + (2 \cdot i + 1) \cdot \mathcal{P}_{2 \cdot i+1} = 2 \cdot i \cdot p_d \cdot \mathcal{P}'_{2 \cdot i} + (2 \cdot i + 1) \cdot (1-p_{d-1}) \cdot \mathcal{P}'_{2 \cdot i} = 2 \cdot i \cdot \mathcal{P}'_{2 \cdot i} + (1-p_{d-1}) \cdot \mathcal{P}'_{2 \cdot i}$. Now we can reduce further derive the average solving time to:

$$\begin{aligned} T &= d + \sum_{i=0}^{i=2^d-1} i \cdot \mathcal{P}_i \\ &= d + \sum_{i=0}^{i=2^{d-1}-1} (2 \cdot i \cdot \mathcal{P}_{2 \cdot i} + (2 \cdot i + 1) \cdot \mathcal{P}_{2 \cdot i+1}) \\ &= d + \sum_{i=0}^{i=2^{d-1}-1} \left(2 \cdot i \cdot \mathcal{P}'_{2 \cdot i} + (1-p_{d-1}) \cdot \mathcal{P}'_{2 \cdot i}\right) \end{aligned}$$

Again $\sum_{i=0}^{i=2^d-1} \mathcal{P}'_{2 \cdot i} = 1$. Now by recurrence it follows:

$$\begin{aligned} T &= d + 1 - p_d + 2 \cdot \sum_{i=0}^{i=2^{d-1}-1} i \cdot \mathcal{P}'_{2 \cdot i} = \\ &= d + 1 + 2 + ... + 2^{d-1} - \\ &- p_{d-1} - 2 \cdot p_{d-2} - ... - 2^{d-1} \cdot p_0 = \\ &= d + 2^d - 1 - \sum_{i=0}^{d-1} 2^i \cdot p_{d-i} \end{aligned}$$

This relation can be further used to calibrate the security level of the protocol having the success probabilities for the honest users and adversaries. In particular the success probabilities of honest users can be easily computed from experimental data as shown in Fig. 4. Note that in the case when $\forall i, p_i = 1$ we get $T = d + 2^d - 1 - (2^d - 1) = d$. In the case of an adversary that has no better chance to guess each bit extracted from the accelerometer value we have $\forall i, p_i = 1/2$ which leads to $T = d + 2^d - 1 - 2^{-1}(2^d - 1) = d + 2^d - 1 - 2^{d-1} + 1/2 = 2^{d-1} + d - 1/2$. That is, an adversary will need roughly around 2^{d-1} computations which are infeasible for $d = 128$ in the case of an 128 bit key.

4.2 Comparison to related methods

A comparison to previous approaches may be relevant. The main difference with ShaVe and ShaCK [22] is in how acceleration data is compared to find a match. In both ShaVe and ShaCK [22] the coherence function is used. The main advantage of this function, initially proposed in [18], is that it is very sensitive allowing to distinguish whether or not the devices are carried by the same person. Indeed this was proved to be feasible in the experimental results from [18] outside the context of spontaneous authentication. However, while this creates a more reliable protocol, there are also two disadvantages. First, this require more processing power, e.g., computing FFT coefficients. Second, the discussion is more complicated when it comes to security since the coherence function works over the power spectrum in the frequency domain. Estimating the exact security of this transformation may be subject of future work for us.

From a computational point of view, ShaVe is based on public key cryptography thus it is more computational expensive. In contrast, ShaCK is based only on symmetric primitives, thus it has somewhat similar computational requirements to SAPHE.

Finally, fuzzy cryptography is the technique of choice when it comes to matching data affected by random noise [7], [8]. This technique is highly efficient as it makes use only of simple error correcting codes and symmetric primitives. However, this technique is commonly used to validate biometric data that are transfered on channels that are generally secure. When it comes to matching data on an insecure channel, care should be taken since if a fuzzy scheme will allow to correct at most n bits out k then an adversary can correct this number of bits as well. Thus, if the adversary is able to recover $k-n$ bits from external observation (such as from a high speed camera) then the protocol is broken. Therefore there is clear dependence of the security level on the degree of similarity between two devices, the guessing probability of an adversary and the recovery rate of the fuzzy scheme.

Bottom line, there are advantages and disadvantages in any of the approaches and indeed fuzzy cryptography may be the most efficient technique. Fuzzy cryptography does not appear to exclude the hashed heuristic trees proposed here since mixing these techniques is possible and more, the hashed heuristic tree can benefit from different optimizations that we discuss next.

4.3 Further Improvements

The protocol variants provided above are simple and efficient, but both the key generation and the key extraction procedures are flexible and can be modified to gain even more efficiency.

The key extraction procedure can be improved to extract more than 1 bit of entropy from each value since the accelerometers return float values that have more than 1 bit of entropy. This can be done simply by comparing the accelerometer value with more than one threshold or using a distinct function. Even more, this can be further strengthen with gesture recognition techniques [4], [6].

Using specific portions of the shaking patterns can also be relevant. In particular the features used in [2] and the Symbolic Aggregate approXimation (SAX) from [23] are of particular interest. Since all these feature are extracted in the time domain it should be easy to embed them in our protocol.

For the heuristic, it may be possible to find better functions. On brief improvement of the Euclidean distance could be the use of the acceleration derivative, indeed, if the derivative is high, then the drift probability is even likely to be higher. Preprocessing techniques can be also used in this respect, a comprehensive list can be found in [9].

5. CONCLUSIONS

Our proposed protocol improves previous results in terms of a simple and efficient way for using accelerometer data for an authenticated key exchange. The solution is also general and allows modification and use with different heuristics that can be more suitable for other kinds of data. Also, the hashed heuristic tree is more secure and efficient as it hinders an adversary from performing a more efficient heuristic search over the candidate key bits and therefore partially addresses the issue of low-entropy input sensor data streams originally identified for the Candidate Key Protocol (CKP) [20]. In contrast to previous approaches, we outline a mathematical bound on the hardness of recovering the keys, which can be further used to configure the protocol parameters, such as the bit-length of the exchanged key. Experiments that we performed in a setting with two Nintendo WiiMotes confirmed the correctness of our approach. This work may open road for using other pattern recognition techniques in wireless device pairing based on accelerometer data. Finally, it is an open question to us whether it is easy or hard to estimate accelerometer values by exterior observations (such as from a high speed camera) and up to what extent pattern recognition techniques may help in this respect. We believe that such a study is highly welcome because it will establish a more concrete bound over the security level of this procedure which is employed in several papers [3], [16], [22].

6. REFERENCES

[1] K. Altun, B. Barshan, and O. Tunçel. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*, 43(10):3605–3620, 2010.

[2] S. Bamberg, A. Benbasat, D. Scarborough, D. Krebs, and J. Paradiso. Gait analysis using a shoe-integrated wireless sensor system. *IEEE Transactions on Information Technology in Biomedicine*, 12(4):413–423, 2008.

[3] D. Bichler, G. Stromberg, M. Huemer, and M. Löw. Key generation based on acceleration data of shaking processes. In *Proceedings of the 9th International Conference on Ubiquitous Computing*, pages 304–317. Springer-Verlag, 2007.

[4] G. Caridakis, K. Karpouzis, A. Drosopoulos, and S. Kollias. Somm: Self organizing markov map for gesture recognition. *Pattern Recognition Letters*, 31(1):52–59, 2010.

[5] 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*, pages 51–64. ACM, 2005.

[6] J. Cheng, C. Xie, W. Bian, and D. Tao. Feature fusion for 3d hand gesture recognition by learning a shared

Protocol	Advantages	Disadvantages	Fix
SAPHE	clear security bound, only symm. functions, flexible heuristic	higher failure rates	high quality sensors
ShaCK	only symm. functions, high success probability	low entropy vectors insecure, less clear security bound	increasing vector size (invariantly leads to higher failure rates)
ShaVe	high success probability	expensive public-key cryptography	N/A
Fuzzy-crypto	reduced failure rate, only symm. functions	special care at choosing system parameters	N/A

Table 1: A comparison between different methods

hidden space. *Pattern Recognition Letters*, 33(4):476–484, 2012.

[7] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer Berlin / Heidelberg, 2004.

[8] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: A brief survey of results from 2004 to 2006. *Security with Noisy Data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*, pages 79–99, 2007.

[9] D. Figo, P. Diniz, D. Ferreira, and J. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, 2010.

[10] D. Gafurov and E. Snekkenes. Gait recognition using wearable motion recording sensors. *EURASIP Journal on Advances in Signal Processing*, 2009:7, 2009.

[11] D. Gafurov, E. Snekkenes, and P. Bours. Spoof attacks on gait authentication system. *IEEE Transactions on Information Forensics and Security,*, 2(3):491 –502, sept. 2007.

[12] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. *Journal of Cryptology*, 19(3):241–340, 2006.

[13] K. Haataja and P. Toivanen. Two practical man-in-the-middle attacks on bluetooth secure simple pairing and countermeasures. *Wireless Communications, IEEE Transactions on*, 9(1):384 –392, january 2010.

[14] L. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Proc. UbiComp 2001*, pages 116–122. Springer-Verlag, September 2001.

[15] H. Junker, O. Amft, P. Lukowicz, and G. Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6):2010–2024, 2008.

[16] D. Kirovski, M. Sinclair, and D. Wilson. The martini synch: joint fuzzy hashing via error correction. *Security and Privacy in Ad-hoc and Sensor Networks*, pages 16–30, 2007.

[17] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing*, 5(6):734–749, 2009.

[18] J. Lester, B. Hannaford, and G. Borriello. "Are You with Me?"–using accelerometers to determine if two devices are carried by the same person. *Pervasive Computing*, pages 33–50, 2004.

[19] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1 –9, march 2009.

[20] R. Mayrhofer. The candidate key protocol for generating secret shared keys from similar sensor data streams. *Security and Privacy in Ad-hoc and Sensor Networks*, pages 1–15, 2007.

[21] R. Mayrhofer, J. Fuss, and I. Ion. UACAP: A unified auxiliary channel authentication protocol. *IEEE Transactions on Mobile Computing*, 2012. *accepted for publication in 2012*.

[22] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Transactions on Mobile Computing*, 8(6):792–806, 2009.

[23] P. Siirtola, H. Koskimäki, V. Huikari, P. Laurinen, and J. Röning. Improving the classification accuracy of streaming data using sax similarity features. *Pattern Recognition Letters*, 2011.

[24] J. Yang, J. Wang, and Y. Chen. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern recognition letters*, 29(16):2213–2220, 2008.