The Martini Synch

Darko Kirovski, Michael Sinclair, and David Wilson
Microsoft Research

Contact: {darkok,sinclair,dbwilson}@microsoft.com

Technical Report MSR-TR-2007-123
September 2007

Microsoft Research
One Microsoft Way Redmond, WA 98052, USA
http://research.microsoft.com

The Martini Synch

Darko Kirovski, Michael Sinclair, and David Wilson

Microsoft Research

Abstract. Device pairing is a significant problem for a large class of increasingly popular resource-constrained wireless protocols such as BlueTooth. The objective of pairing is to establish a secure wireless communication channel between two specific devices without a public-key infrastructure, a secure near-field communication channel, or electrical contact. We use a surprising user-device interaction as a solution to this problem. By adding an accelerometer, a device can sense its motion in a Cartesian space relative to the inertial space. The idea is to have two devices in a fixed, relative position to each other. Then, the joint object is moved randomly in 3D for several seconds. The unique motion generates approximately the same distinct signal at the accelerometers. The difference between the signals in the two inertially conjoined sensors should be relatively small under normal motion induced manually. The objective is to derive a deterministic key at both sides with maximized entropy that will be used as a private key for symmetric encryption. Currently, our prototype produces between 10–15 bits of entropy per second of usual manual motion using off-the-shelf components.

Keywords: device pairing, key exchange, fuzzy hashing, error correction.

1 INTRODUCTION

Establishing a secure session is one of the least efficiently resolved problems with modern low-cost wireless protocols such as BlueTooth [1]. The key challenge is that such protocols do not assume the existence of a trusted authority that can certify public-keys; hence one cannot build a standard public-key infrastructure (PKI) [2]. In a PKI, the public key of the trusted authority would be hardwired into all devices. Each device or user would have a single public-private key-pair along with a certificate that vouches the authenticity of the public key. Key exchange using an underlying public-key cryptosystem such as RSA [5], would involve authentication followed by generation of a common secret, i.e., session key [6]. Using a system of certificates, the central authority could manage the trust in the world-wide network [7]. Unfortunately, this class of solutions is prohibitively expensive for most applications of mobile ad-hoc wireless protocols.

We introduce the first protocol that derives a common secret between two devices based on kinetic user-device interaction. The idea is simple: two devices equipped with 3-axis accelerometers and moved along the same trajectory, should produce approximately similar output from each of the sensors. While similar ideas have been proposed earlier for device notification (e.g., [3,4]), our protocol is the first to derive a common secret from two fuzzy replicas of a common source. We show that in our scenario the difficulty of the traditional fuzzy hashing problem can be successfully overcome as the

participants in the protocol **can communicate** while deriving the common secret. Although the replicas are only probabilistically equivalent to the source, our algorithm uses error correcting codes to produce two equivalent keys on both devices with a certain probability of failure. In the protocol, Alice computes a syndrome based upon a mutually agreed error correcting code and sends it to Bob. Based upon the syndrome, Bob can correct the errors in his sensor readings, i.e., adjust them to equal Alice's sensor readings. The error corrected sensor readings are then used on both sides to set up a private mutually-agreed session key. As high entropy of the session key stems only from the random motion of the two devices, much like shaking a drink of Martini, we have named our protocol The Martini Synch.

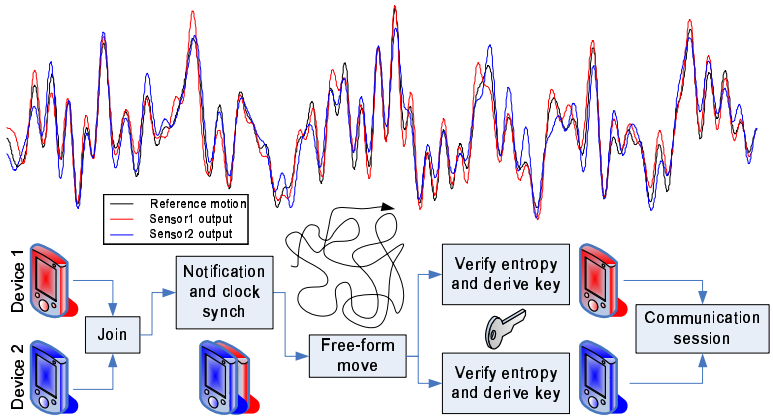

Fig. 1. (top) Sensor output for two devices compared to the acceleration curve for the reference motion. (bottom) Diagram of the basic steps of The Martini Synch protocol.

The Martini Synch is power-efficient as the amount of data exchanged between the devices is actually lower compared to traditional key exchange protocols specified in standards such as the IEEE P1363 [8]. Cost-wise 3D accelerometers should not increase the device price by more than US$1 per axis [9,10], however, we stress that there exist design proposals that could lower this price at least one order of magnitude. As we impose only relative measurement consistency across different sensors, not their absolute accuracy, we believe that such accelerometers can be built at low cost.

To evaluate the platform, we built two prototype devices based upon off-the-shelf accelerometers and BlueTooth transceivers. The devices resulted in key generation rates of 10–15 bits per second under normal manual motion.

2 RELATED WORK

2.1 BlueTooth Security

BlueTooth uses the SAFER+ algorithm for authentication and key generation [11] and the E0 stream cipher is used for encrypting packets [12,13,14]. Simultaneously, fre-

quency hopping makes eavesdropping on BlueTooth-enabled devices more difficult [1]. Still, there are a number of security concerns reported for BlueTooth. Some of the first concerns were raised with respect to certain poor implementations [15] – the security flaw would lead to disclosure of certain personal data. First reverse engineering of the security PIN used for device pairing was revealed in [16]. The first worm program, Cabir, that targeted BlueTooth devices was written in 2004 by a group of virus writers known as 29a; it used the short-range wireless feature of Smart Phones that run the Symbian OS to detect other Symbian phones and then transfer itself to the new host as a package file. Since then both passive [17] and active attacks [18] have been realized. The essence of the problem is in the fact that wireless communication occurs over a public channel, therefore key exchange is prone to the man-in-the-middle attack [19]. Typically, inexpensive protocols do not assume a trusted authority, therefore it is difficult to build a PKI in the system. According to our research, the work proposed in this paper is the first that relies on device motion, not traditional cryptography (asymmetric or symmetric), to establish a shared secret between two devices.

2.2 Gesture-based Device Notification

Several gesture-based techniques have been proposed to date for device notification. In this context, two devices use a gesture to signal demand for mutual communication. Bumping devices as a gesture has been proposed by Hinckley for aligning multiple-screen images [3]. Holmquist et al. have proposed shaking conjoined devices as means of establishing communication. However, their system "Smart-Its Friends" does not incorporate a communication protocol to derive a shared secret [4]. In "Smart-Its Friends" two devices exchange their sensed motion patterns in plain-text as a request for communication. In their work, Lester et al. used motion sensors to identify that in a cloud of mobile devices, two or more are worn by the same person by analyzing the stress patterns due to walking and other activities [20]. Patel et al. proposed a gesture-based communication initiation between a mobile device augmented with accelerometers and a public terminal [21]. Finally, Castelluccia and Mutaf proposed shaking devices together in order to filter out the radio frequency noise stemming from the environment while the relative signal energy between the two devices would stay the same [22]. With the exception of the last technique, none of the previous efforts aimed at generating a shared secret key between the communicating parties: the essential ingredient of private communication. To that extent, our proposal is the first to establish such a secret in accelerometer-equipped devices using a novel fuzzy hashing protocol.

2.3 Fuzzy Hashing

Hashing of fuzzy data has been addressed for several different types of sources: images [23,24,25], audio [26,27,28], textual documents [29,30], biometrics [31], and graphics and protein matching [32]. Relative to our application, in most of the related work, hashing diverse similar structures is efficient if the resulting hashes are within a certain minimal distance. In our application we have an additional relaxation that the encoder and decoder can communicate while agreeing on a mutually equivalent secret. This

relaxation greatly simplifies this otherwise difficult task. For that reason we do not review in detail the fuzzy hashing techniques deployed in the referenced previous work.

3 THE MARTINI SYNCH PROTOCOL

The steps of the proposed protocol are illustrated in Figure 1. The hardware requirements include a 3-axis accelerometer in the participating devices and a reliable wireless communication stack such as BlueTooth. Hardware platforms that contain a hard-drive are typically equipped with accelerometers to detect shock or free fall for data protection – hence, a large class of existing devices already satisfies the hardware requirements. We review the protocol steps in more detail:

- **Notification.** In order to launch the protocol, the two participating devices are initially notified by their users that they should establish a session key. This can be done in several ways including: a physical *push-button* (unless it already exists, an action-specific push-button can be prohibitively expensive), a *proximity test* performed by measuring the energy of a received radio beacon or by detecting a source of near-field communication such as an RFID, or by *bumping the devices* and detecting the bumps in the accelerometers' output [3].
- **Synchronization of internal clocks.** Upon initial discovery, the devices synchronize their internal clocks. Since sensor output is typically sampled at rates lower than 1kHz, millisecond accuracy is both sufficient and inexpensive to establish.
- **The Martini shake.** Next, the devices are mechanically confined to a single object (i.e., held together) and then randomly moved in free-space. The motion is induced manually. While moving, the devices internally estimate the resulting entropy of the collected sensor measurements. When one or both of the devices reaches a desired minimal entropy, they signal to each other the end of the data collection process. Alternatively, the same process can also be stopped using a timer.
- **Joint fuzzy hashing.** If the entropy of the collected signals is sufficient on both sides, the devices finalize the key generation phase by exchanging a set of messages whose purpose is to perform the joint fuzzy hashing of the sensor outputs while securing the integrity of the derived deterministic secret. The protocol for joint fuzzy hashing is detailed in the subsequent section.
- **Secret verification.** Finally, the devices verify that they have derived the same key by exchanging the ciphertext of a known plaintext. If the ciphertexts match, the devices proceed with the secure session.

The key characteristic of the protocol is that it establishes a common secret on two devices based upon an activity in the physical world. Thus, private keys are established on both sides without the assistance from public-key cryptography. Assuming a secure symmetric encryption scheme,[1] the only remaining tool for the adversary is video-taping the motion using high-speed cameras and then deploying 3D computer vision techniques to estimate the motion [33]. Suspecting that, users can take certain straightforward precautions in case they are concerned about this type of attack.

[1] Unfortunately this is not the case with BlueTooth currently [12,13,14].

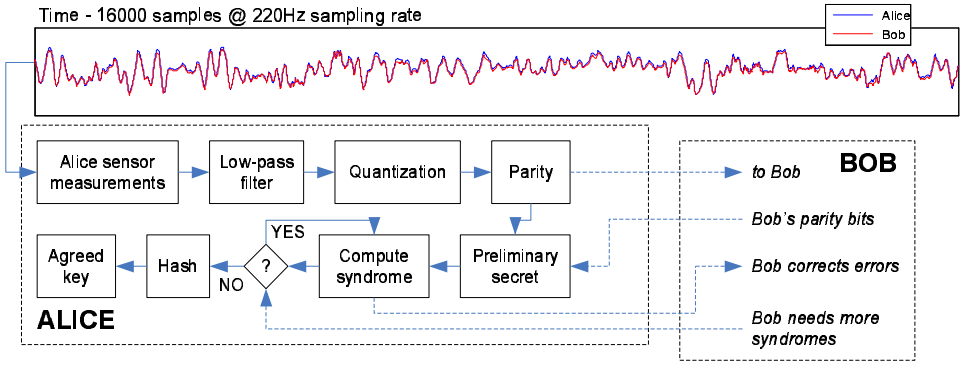

Fig. 2. Block diagram of the Martini Synch protocol. An example of sensor measurements taken from two devices during a 72.7 second Martini Synch at a sampling rate of 220Hz.

4 JOINT FUZZY HASHING

In this section we propose a solution to the joint fuzzy hashing problem. First Alice and Bob convert their sensor measurements into a sequence of 0's and 1's which they will largely agree upon, with possibly some errors. This sequence of 0's and 1's is the "preliminary secret". Then Alice and Bob communicate a small number of bits of information about this preliminary secret, enough bits to figure out where the discrepancies are, but without leaking too many bits about their preliminary secret to an eavesdropper. This is the error correction phase. When Alice and Bob estimate that they have enough entropy in their secrets which was not leaked during the error correction phase, they then hash their corrected preliminary secrets down to a common secret key. The Martini Synch protocol is illustrated in Figure 2.

4.1 Preliminary Secret

After the signals are passed through a low-pass filter to reduce noise, Alice and Bob quantize their respective signals by dividing by a quantization step size Q and rounding the result to the nearest integer. Next Alice and Bob tell each other the parities of each of these quantized values. If the quantization step size Q is large enough, and Alice and Bob agree on the parity of the quantized value, then it is likely that they agree upon the quantized value itself.

Figure 3 shows one possible stream of quantized values that Alice and Bob might measure, and the corresponding parity bits that they would then communicate to each other. An eavesdropper can of course listen to these parity bits. Since the sampling rate is relatively high (220 Hz) and the signals have gone through a low-pass filter, an eavesdropper might reasonably infer that if the parity bits of either Alice or Bob have not changed, then it is likely that the quantized values have not changed either (see e.g. the first 4 measurements in Figure 3). If a secret is made out of all the quantized values, then the parity bits leak some partial information about this secret. Therefore we extract

a preliminary from the agreed-upon quantized values in a different manner, as described in the caption of Figure 3.

Alice measures:	5	5	5	5	4	4	3	2	2	5	4	5	6	6	7	8	5	5	5	3	2
Bob measures:	5	4	4	5	4	4	2	2	2	3	5	5	5	6	7	8	6	5	5	3	2
Alice transmits:	1	1	1	1	0	0	1	0	0	1	0	1	0	0	1	0	1	1	1	1	0
Bob transmits:	1	0	0	1	0	0	0	0	0	1	1	1	1	0	1	0	0	1	1	1	0
Alice records:					D					U				U	U	U		D			D
Bob records:					D					D				U	U	U		D			D

Fig. 3. An example sequence of quantized measurements that Alice and Bob might make, with the parity bits that they would communicate to each other. Shown in gray are those times for which Alice and Bob agree on the parity, and for which the parity is different than for the previous time that Alice and Bob agreed on the parity. At each time shown in gray, Alice and Bob record an "up vs. down" bit which indicates whether their quantized signals went up or down since the last time shown in gray. Since an eavesdropper learns essentially nothing about the "up vs. down" bits from the parity bits, and most of the time Alice and Bob agree on the "up vs. down" bits, we use they bits to form the preliminary secret.

Devices with three accelerometers will produce three data streams, while it is only necessary to produce one secret key, so we splice these three streams of "up vs. down" bits into one preliminary secret.

There are two issues with Alice and Bob's preliminary secret. Due to inertia, the bits within the secret are correlated with one another, so there are fewer bits of entropy than the length of the preliminary secret would indicate. We will need to estimate the entropy. A second issue is that unless Q is rather large, Alice and Bob will disagree on some fraction of the bits in their "common" preliminary secret — these bits are errors. For practical purposes, if any bits are in error, then the mutual secret has no value. But if Q is taken to be so large as to ensure that there are likely no errors, then this significantly reduces the entropy of the mutual secret that Alice and Bob can obtain from their measurements. We deal with this second issue first.

4.2 Error Correction

In the interest of increasing the bits of entropy per second of the Martini Synch, one would like to sample the signals more frequently, and make the quantization intervals correspondingly smaller. Reducing the quantization intervals will necessarily increase the likelihood that the two devices measure a different value. We now describe how Alice and Bob may correct the resulting discrepancies in their preliminary secrets.

The idea is for Alice and Bob to use a parity-check error-correcting code to correct the errors in their measurements. This use of an error-correcting code is somewhat

unusual in that the encoding procedure is skipped, the participants only perform the decoding part. To explain, we introduce some notation. Let $\boldsymbol{a}$ be a column vector of size n containing Alice's preliminary secret, and let $\boldsymbol{b}$ be a column vector containing Bob's preliminary secret. Let $\boldsymbol{e}$ denote the column vector of errors — $\boldsymbol{e} = \boldsymbol{a} \oplus \boldsymbol{b}$. Let H denote the $k \times n$ parity-check matrix of a binary error-correcting code.

In the normal use of an error-correcting code, a message of $m = n - k$ bits is expanded into a vector $\boldsymbol{v}$ of n bits satisfying the property that $H\boldsymbol{v} = 0$. Upon transmitting $\boldsymbol{v}$ over a noisy channel, some of the bits are corrupted, so that the received message is $\boldsymbol{r} = \boldsymbol{v} + \boldsymbol{e}$, where $\boldsymbol{e}$ denotes the errors. The receiver then computes $H\boldsymbol{r} = H(\boldsymbol{v} \oplus \boldsymbol{e}) = H\boldsymbol{e}$, and from this infers where the errors were, and corrects them. Most decoders correct the errors using only $H\boldsymbol{e}$, but sometimes the receiving device outputs not just $\boldsymbol{r}$ but also a vector of reliability estimates that the decoder may make use of when correcting errors [34].

In our application, Alice computes $H\boldsymbol{a}$ and sends the resulting k bits to Bob. Bob computes $H\boldsymbol{b}$ and takes the exclusive-or of the result with what Alice sent — $H\boldsymbol{b} \oplus H\boldsymbol{a} = H(\boldsymbol{b} \oplus a) = H\boldsymbol{e}$. At this point Bob is in the same position as the receiver of an encoded message that was corrupted by a noisy channel, and can determine which bits of his measurements $\boldsymbol{b}$ he needs to flip for them to agree with Alice's measurements $\boldsymbol{a}$.

4.3 Progressive Error Correction

Rather than use an error correcting code with a fixed number of checksum bits, Alice and Bob are at liberty to transmit checksum bits until they decide that they have corrected all the errors. In the event that there is a small number of errors, they may stop communicating checksum bits early so as to avoid leaking data to an eavesdropper. In the event that there are many errors, Alice and Bob many continue to communicate checksum bits until they are satisfied that all the errors have been corrected.

To illustrate this "progressive error correction", where the number of checksum bits depends on the errors that occur, it is instructive to consider a concrete example, such as the scheme that we adopted. For our application we use BCH codes on blocks of length 63 bits [34,35]. The preliminary secret is partitioned into blocks of length 63 which are corrected separately. In a BCH code, the bit positions are indexed by the non-zero elements of a finite field, in our case the field is $\mathbb{F}_{64}$, represented as binary polynomials in $\mathbb{F}_2[x]$ modulo $x^6 + x + 1$. The element x generates the multiplicative subgroup of $\mathbb{F}_{64}$. So for integers p, Alice can transmit $S_p^A := \sum_{i=1}^{63} a_i x^{pi} \bmod x^6 + x + 1$ in six bits, whereupon Bob can compute $S_p^B := \sum_{i=1}^{63} b_i x^{pi} \bmod x^6 + x + 1$ and determine

$$S_p := S_p^A \oplus S_p^B = \sum_{i=1}^{63} e_i \alpha^{pi} \bmod x^6 + x + 1.$$

In the event that there are t errors, knowing S_p for the first t odd positive integers p ($S_1, S_3, \ldots, S_{2t-1}$) is enough to determine the locations of the t errors using the Berlekamp-Massey algorithm [34], which would allow Bob to change his copy of the preliminary secret to agree with Alice's. Of course Alice and Bob do not know beforehand how many errors there will be. But if the first t odd power-sums $S_1, S_3, \ldots, S_{2t-1}$

are consistent with there being significantly fewer than t errors, then Bob can infer that there are in fact fewer than t errors, and that it is not necessary for Alice to transmit additional checksum bits. When t is large, in the event that there are more than t errors, it becomes increasingly likely that the decoding procedure will detect this. Thus for large t there is less need for Alice to send Bob extra checksum bits for Bob to be confident that there are not extra errors. To better take advantage of the fact that the decoding procedure can often detect the presence of too many (random) errors, and to offset the fact that errors are frequently clumped together (not randomly located), we picked and fixed a random permutation on 63 items, and permuted the input bits according to this permutation before doing the error correction. To run the protocol we need to specify a function $f(t) \leq t$ such that, when $S_1, \ldots, S_{2t-1}$ are consistent with $\leq f(t)$ errors, Bob is satisfied that he knows Alice's preliminary secret. We used the function:

$$f(t) = \begin{cases} t-2 & t \leq 4 \\ t-1 & 5 \leq t \leq 10 \\ t & t \geq 11. \end{cases}$$

An eavesdropper listening to Bob's communications will learn about the number of discrepancies between the two preliminary secrets, but Bob's communications do not reveal anything about Alice's version of the preliminary secret, which is the one that will be hashed to form a secret key. Each time that Alice sends a S_p^A she reveals at most six bits about her preliminary secret. It turns out that sending $S_1^A, S_3^A, \ldots, S_{2t-1}^A$ will in general reveal fewer than $6t$ bits about the preliminary secret, since there are some linear relations between the transmitted bits. The number of bits revealed is the rank of the associated checksum matrix, and is fairly well understood [35, Chapt. 9, sec. 3 & 4]. For the BCH code over $\mathbb{F}_{64}$ we summarize the number of leaked bits in Table 1.

Table 1. Number of bits leaked when t syndrome packets are sent from Alice to Bob.

t:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16–31	32–63
# leaked bits:	6	12	18	24	27	33	39	45	45	45	47	53	53	56	56	62	63

4.4 Dealing with Information Leakage

When Alice transmits the k bits of $H\boldsymbol{a}$ to Bob, an outside observer gains some information about Alice's measurements $\boldsymbol{a}$, so Alice and Bob would not want to simply use $\boldsymbol{a}$ itself as their common secret. This problem is easy enough to deal with when all 2^n possible values of $\boldsymbol{a}$ are equally likely. In this case Alice and Bob can simply agree beforehand upon a maximal rank submatrix A of H, and discard the bits of $\boldsymbol{a}$ whose positions correspond to the columns of A. Let us assume that H has rank k, since otherwise the bits of $H\boldsymbol{a}$ corresponding to dependent rows of H give no extra information to either Bob or the outside observer. Regardless of the values of the remaining $n-k$ bits that Alice and Bob keep, since the k discarded bits are uniformly random, and since

their positions correspond to a full-rank submatrix of H, the message $H\boldsymbol{a}$ is uniformly random, and thus contains no information about the $n-k$ bits that Alice and Bob keep.

When the measured values $\boldsymbol{a}$ are not completely independent of one another, more care is needed to ensure that $H\boldsymbol{a}$ does not leak much information about the secret that Alice and Bob derive from $\boldsymbol{a}$. We shall assume that for some $r < 1$, no set of measurements $\boldsymbol{a}$ occurs with probability greater than r^n. Under these circumstances we might hope to extract $n \log_2 r^{-1} - k - O(1)$ nearly uniformly random bits from $\boldsymbol{a}$ which are nearly independent of $H\boldsymbol{a}$, since conditional on the transmitted syndrome, no measurement occurs with probability more than $2^k r^n$. Alice and Bob can do this making use of a random hash function which may be public and known to the outside observer. Let M be a $s \times n$ uniformly random matrix of 0's and 1's which may be public, where $s = n \log_2 r^{-1} - k - O(1)$. The common secret of Alice and Bob is $M\boldsymbol{a}$.

The probability that two different measured values $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ get hashed to to the same secret $M\boldsymbol{a}_1 = M\boldsymbol{a}_2$ is precisely 2^{-s}. Conditional upon the values of $H\boldsymbol{a}$, no value of $\boldsymbol{a}$ occurs with probability greater than $2^k r^n$. It can be shown using the second-moment method, that even after an observer has learned $H\boldsymbol{a}$, the expected total variation distance between the secret $M\boldsymbol{a}$ and uniformly random set of s bits, is no more than $2^{k+s} r^n$ (e.g., see [36] for further explanation). When s is chosen to be $n \log r - k - O(1)$, the outside observer learns essentially nothing about Alice and Bob's common secret $M\boldsymbol{a}$.

4.5 Entropy Estimate

Empirically estimating the entropy of a process is generally difficult, but we need estimates of the entropy to judge the strength of the common secret that Alice and Bob distill from their measurements. As mentioned above, the length of the preliminary secret consisting of "up vs. down" bits is a poor estimate of the entropy, since there tend to be alternating strings of 1's and 0's whose length is longer than what one would find in a uniformly random string. To better estimate the entropy, for each of the three data streams from the three coordinate axes, we let r_i denote the number of runs of 0's and 1's there are of length r_i, and estimate the entropy for a given data stream to be:

$$\sum_i r_i \log_2 \frac{\sum_i r_i}{r_i}.$$

When the sum of the three entropy estimates, minus the number of leaked bits, exceeds a specific security requirement, e.g., 60 bits, Alice and Bob determine that they have enough entropy in the corrected preliminary secret to hash down to form a common secret key.

4.6 Additional Remarks

Since a crucial part of the Martini Synch protocol occurs in the physical world, it is important to stress the constraints related to this process. The difference in $\mathbf{x}$ and $\mathbf{y}$ is influenced by two components:

i noise in the sensors stemming from calibration and other physical influences, and

ii the actual difference in the motion vectors for the two fixed points in the relative Cartesian space where device accelerometers are positioned.

In order to reduce the latter noise, the devices must be designed so that the accelerometers are positioned as closely as possible during the protocol. In our experiments, motion induced using typical manual kinetics caused negligible additional noise compared to the noise collected when both devices are still (e.g., noise type *i*). In the tests, the accelerometers were positioned at a distance of approximately one inch.

4.7 Parameters

We have found the following parameters to generally work well for the Martini Synch. For the noise filtering, we removed the DC component of the signal (to reduce the need for calibration) and convolved the signal with a binomial distribution of order 256 (whose characteristic width is around 16–32). The sensors we used generated 10-bit measurements, and for the quantization parameter Q we found $Q = 40$ to work satisfactory. The performance of the protocol depends in part on how vigorously the user accelerates the devices, but with these parameters 16000 measurements (corresponding to 72.7 seconds) generally produces a preliminary secret of about n =800-1400 bits (per coordinate axis), containing about $n/3$ blocks of 0's and 1's, and about $n/10$ errors. After correcting the errors one can expect to have about 400 bits of entropy for each of the three coordinate axes.

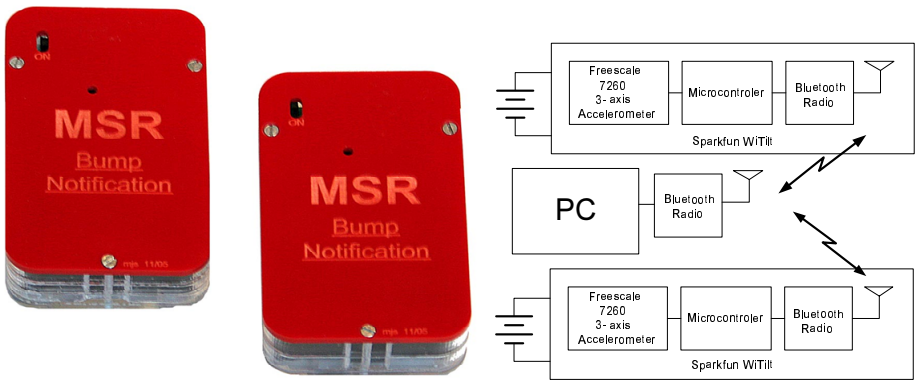

Fig. 4. (left) Photo of the encased prototype devices used in the experiments. (right) Block diagram showing two portable devices and an intermediate PC for processing.

5 EMPIRICAL EVALUATION

The prototypes produced for this research consist of two handheld battery operated devices, each equipped with a 3-axis accelerometer and a BlueTooth radio. The experimental testbed is illustrated in Figure 4. For the prototype, the information exchanged

was mediated by an intermediate BlueTooth-enabled PC. In a real-world scenario, the two devices would process the information locally as well as communicate with each other via their radios. The WiTilt 3-axis accelerometer with BlueTooth radio was purchased from Sparkfun [37]. The accelerometers are model 7260 from Freescale Semiconductors [38]. They are MEMS (micro-electromechanical systems) 3-axis accelerometers with a 1.5-6g acceleration range, 0.5mA operating current with a detection range greater than 1kHz.

5.1 Evaluation of The Martini Synch

In this subsection we quantify the key performance features of the Martini Synch protocol. We collected data for thirteen 3-axis sensor vectors of length 16K samples from a group of 5 users. The sensor vectors were sampled at 220Hz. We did not specifically calibrate the sensors prior to the experiments. The usage of the platform is sufficiently simple so that no other guidelines were issued to the users except that they should randomly move the joined objects.

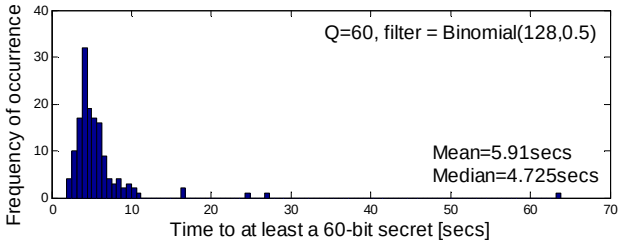

Fig. 5. Distribution of timings required to obtain a shared secret of estimated minimal 60 bits.

We show the first set of results in Figure 5. We illustrate the distribution of timings required to obtain a shared secret of estimated minimal 60 bits. The subjects followed usual manual motion for 11 and severe shaking for 2 out of the 13 benchmarks. As a result, the mean and the median results are 5.9 and 4.7 seconds respectively over the acquired set of sensor measurements. Here we report results for $Q = 60$ and a binomial(128,0.5) filter. It is important to stress that overly energetic shaking of the devices resulted in acceleration that could not be captured accurately with our devices – hence, in these cases users required longer times (> 10 seconds) to generate a strong secret.

In the second set of experiments we evaluate the probability of a false negative and false positive (i.e., produced entropy per second). The left diagram shows a color-coded plot of the probability of a false negative ε_{FN} for various quantization step values and binomial(x, 0.5) filters. One can observe that in a large part of the region of interest we have $\varepsilon_{FN} < 0.02$. For example, this type of performance corresponds to the false negatives produced while typing a textual password. Similarly on the right side we present in

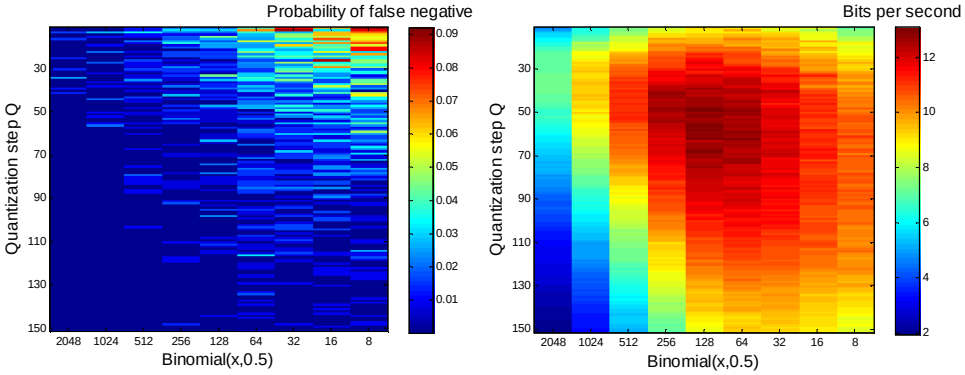

Fig. 6. (left) Color-coded plot of the probability of a false negative ε_{FN}. (right) Color-coded diagram of the entropy per second for correctly resolved shared secrets.

a color-coded diagram the achieved entropy per second for correctly resolved shared secrets. We computed the entropy according to the algorithm presented in Subsection 4.5. One can observe that a relatively large area in the tested parameter set, corresponded to bit-rates in excess of 12 bits per second. Thus, we point to a particular parameter selection $Q = 50$ and a binomial(128,0.5) filter as a good design solution for the experimental platform we developed. For the selected design parameters, individual five second Martini Synch's produced entropies commonly in the range between 10 to 15 bits per second.

6 A User-Study

The simplicity of the proposed user-device interface points to a few unknowns with respect to user acceptance of the Martini Synch. During a demonstration fair we asked 47 persons, most of them with technical background, about the convenience of using the Martini Synch. We conducted the following survey:

1 Is shaking a pair of devices convenient for generating a shared secret?
2 For a given device, no other mechanism is available for generating a shared secret. Would you perform the Martini Synch to accomplish this task or you would deem the device unusable?
3 In your opinion, how many seconds of shaking devices results in a good balance between usability and security?
4 Are you likely to hide the device motion for fear of computer vision attacks?
5 Do you prefer bumping devices vs. software initiation of the key generation protocol?
6 Which mechanical feature is the most effective to lock two devices in place?
7 Is usage of the Martini Synch self-explanatory?
8 In your opinion, is the user-device interface appealing to the following individual age groups?

Table 2. Results of a small user study. In total, 47 persons were surveyed.

Question	Answers				
1	(yes) 39			(no) 8	
2	(yes) 45			(no) 2	
3	< 2s	2-3s	3-4s	4-5s	> 5s
	3	3	18	17	3
4	(yes) 2			(no) 45	
5	(yes) 36			(no) 11	
6	magnet	velcro	joints	hi-friction surf.	
	25	2	1	19	
7	(yes) 45			(no) 2	
8	10-20	21-40	41-60	61+	
(yes)	43	35	20	45	

Responses to the survey are tabulated in Table 2. In summary, the technology was well accepted. All but two participants acknowledged that they would use the technology if available on a low-cost device. A majority recognized the need for shaking the device over a longer period – most of them targeting the 3-5 second period as convenient. Similarly most participants preferred bumping devices to initiate a key generation session as opposed to a point-and-click software-only user interface. Almost all users found the protocol easy to comprehend. Finally, the survey participants estimated that the 40-60 age group is least likely to accept the new user-device interface while its appeal is the strongest to the youngest and elderly. We conclude the report on this informal and simple user study with a disclaimer that the sample of users was not statistically large as well as broad in terms of technical background and acceptance of modern technology. To that extent, we point to a likely discrepancy of the presented results with respect to ground truth.

7 SUMMARY

The Martini Synch protocol establishes a secure wireless communication channel between two specific devices without a public-key infrastructure, a secure near-field communication channel, or electrical contact. It relies on a surprising user-device interaction to achieve its objective. Using an accelerometer, a device can sense its motion in a Cartesian space relative to inertial space. The idea is to have two devices in a fixed, relative position to each other. The joint object is moved randomly in 3D for several seconds. The unique motion generates approximately the same distinct signal at corresponding devices' accelerometers. The protocol uses a novel distributed fuzzy hashing algorithm based upon exchanging error correction syndromes that derives probabilistically the same secret key in both devices based upon the observed joint motion. We developed a prototype platform using off-the-shelf components to show that even in a simple implementation, The Martini Synch protocol can generate between 10 and 15 bits of entropy per second of manual motion.

ACKNOWLEDGMENTS

We would like to thank Kris Andersen, Tom Blank, and Gideon Yuval for insightful discussions that have improved the contents of this manuscript.

References

1. J. Haartsen, M. Naghshineh, J. Inouye, O. Joeressen, and W. Allen. BlueTooth: Vision, goals, and architecture. Mobile Computing and Communications Review, Vol.2, pp.38–45, 1998.
2. IETF PKIX workgroup. Public-Key Infrastructure X.509. Available on-line at `http://www.ietf.org/html.charters/pkix-charter.html`.
3. K. Hinckley. Synchronous Gestures for Multiple Users and Computers. ACM UIST Symposium on User Interface Software & Technology, pp.149–158, 2003.
4. L.E. Holmquist, et al. Smart-its friends: a technique for users to easily establish connections between smart artefacts. UBICOMP, pp.116–122, 2001.
5. R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of ACM, Vol.21, (no.2), pp.120–126, 1978.
6. IETF TLS workgroup. Transport Layer Security. Available on-line at `http://www.ietf.org/html.charters/tls-charter.html`.
7. M. Naor and K. Nissim. Certificate revocation and certificate update. In Proceedings of the 7th USENIX Security Symposium, 1998.
8. IEEE 1363-2000: Standard Specifications for Public Key Cryptography. Available on-line at `http://grouper.ieee.org/groups/1363`.
9. G. Roos. 3-axis accelerometer sets low power benchmark for mobile devices. Wireless Net Design Line, October 31, 2005. Available on-line at `http://www.wirelessnetdesignline.com/products/173402048`.
10. Analog Devices Corp. ADXL330. Available on-line at http://www.analog.com.
11. J. Massey, G. Khachatrian, and M. Kuregian. Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES). NIST AES Proposal, 1998.
12. S. Fluhrer and S. Lucks. Analysis of the E0 Encryption System. SAC, 2001.
13. Y. Lu and S. Vaudenay. Faster correlation attack on BlueTooth keystream generator E0. CRYPTO, pp.407-425, 2004.
14. F. Armknecht. A linearization attack on the BlueTooth key stream generator. Cryptology ePrint Archive, report 2002/191, available from `http://eprint.iacr.org/2002/191`, 2002.
15. A. Laurie, M. Herfurt and M. Holtmann. Hacking BlueTooth enabled mobile phones and beyond. 21st Chaos Communication Congress, 2003.
16. O. Whitehouse. War nibbling: BlueTooth insecurity. @Stake, research report, 2003.
17. F.-L. Wong and F. Stajano. Repairing the BlueTooth pairing protocol. Proceedings of Security Protocols Workshop, 2005.
18. Y. Shaked and A. Wool. Cracking the BlueTooth PIN. International Conference on Mobile Systems, Applications, and Services, pp.39-50, 2005.
19. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. Handbook of applied cryptography. CRC Press, 1997.
20. 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, pp.33–50, 2004.
21. S.N. Patel, J.S. Pierce, and G.D. Abowd. A gesture-based authentication scheme for untrusted public terminals. ACM Symposium on User Interface Software and Technology, pp.157–160, 2004.

22. C. Castelluccia and P. Mutaf. Shake Them Up!: a movement-based pairing protocol for CPU-constrained devices. MobiSys, 2005.
23. V. Monga, A. Banerjee, and B.L. Evans. A Clustering Based Approach to Perceptual Image Hashing. IEEE Transactions on Information Forensics and Security, Vol.1, (no.1), pp.68–79, 2006.
24. M. Johnson and K. Ramchandran. Dither-Based Secure Image Hashing Using Distributed Coding. IEEE International Conference on Image Processing, 2003.
25. A. Swaminathan, Y. Mao and M. Wu. Robust and Secure Hashing for Images. IEEE Transactions on Information Forensics and Security, 2006.
26. T. Kalker, J. Haitsma, and J. Oostveen. Robust audio hashing for content identification. International Workshop on Content Based Multimedia Indexing, 2001.
27. C.J. Burges, J.C. Platt, and S. Jana. Distortion discriminant analysis for audio fingerprinting. IEEE Transactions on Speech and Audio Processing, Vol.11, (no.3), pp.165-174, 2003.
28. M. Gruhne. Robust audio identification for commercial applications. Fraunhofer IIS, AEMT, 2003.
29. A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. 25th VLDB Conference, 1999.
30. R. Weber, H.-J. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. 24th VLDB Conference, pp.194-205, 1998.
31. Y. Dodis, L. Reyzin and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. EUROCRYPT, pp.523–540, 2004.
32. H.J. Wolfson and I. Rigoutsos. Geometric Hashing: An Overview. IEEE Computational Science and Engineering, Vol.4, (no.4), pp.10–21, 1997.
33. J. Neumann, C. Fermuller, and Y. Aloimonos. Polydioptric camera design and 3d motion estimation. IEEE Conference on Computer Vision and Pattern Recognition, Vol.II, pp.294–301, 2003.
34. E. R. Berlekamp, Algebraic coding theory. McGraw-Hill Book Co., 1968 pp. xiv+466.
35. F. J. MacWilliams and N. J. A. Sloane, The theory of error-correcting codes. I. North-Holland Mathematical Library, Vol. 16. North-Holland Publishing Co., 1977, pp. xv+369.
36. D. B. Wilson, Random random walks on $\mathbb{Z}_2^d$. Probability Theory and Related Fields, 108(4):441–457, 1997.
37. Sparkfun, Inc. Available on-line at `http://www.sparkfun.com`.
38. Freescale Semiconductors, Corp. Available on-line at `http://www.freescale.com`.