Antikainen, Markku; Sethi, Mohit; Matetic, Sinisa; Aura, Tuomas

# Commitment-based device-pairing protocol with synchronized drawings and comparison metrics

*Published in:*
Pervasive and Mobile Computing

*DOI:*
10.1016/j.pmcj.2014.10.006

Published: 01/01/2015

*Document Version*
Publisher's PDF, also known as Version of record

Pervasive and Mobile Computing 16 (2015) 205–219

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

# Commitment-based device-pairing protocol with synchronized drawings and comparison metrics☆

Markku Antikainen [a,*], Mohit Sethi [a,b], Sinisa Matetic [a], Tuomas Aura [a]

[a] Aalto University, Finland

[b] Nomadiclab, Ericsson Research, Finland

ARTICLE INFO

*Article history:*
Available online 31 October 2014

*Keywords:*
Security
Device pairing
Commitment protocol
Edit distance

ABSTRACT

This article presents a new method for pairing devices securely. The commitment-based authentication uses a fuzzy secret that the devices only know approximately. Its novel feature is time-based opening of commitments in a single round. We also introduce a new source for the fuzzy secret: synchronized drawing with two fingers of the same hand on two touch screens or surfaces. The drawings are encoded as strings and compared with an edit-distance metric. A prototype implementation of this surprisingly simple and natural pairing mechanism shows that it accurately differentiates between true positives and man-in-the-middle attackers.



## 1. Introduction

Secure device pairing, which enables easy creation of long-term associations as well as ad-hoc transactions between personal computing devices, is an important and widely studied area [1]. The key challenge in device pairing is to allow two devices to securely identify and authenticate each other without having any a-priori shared information. Several communication technologies, such as Bluetooth [2], WiFi [3], and ZigBee [4], require the user to enter the same key string or authentication code to both devices or to compare and approve codes displayed by the devices. While these methods can provide reasonable security, they require user interaction that is relatively unnatural and often considered a nuisance. Thus, there is an ongoing quest in pervasive computing research for more natural ways of pairing devices. The method proposed in this paper, synchronized drawing with two fingers on touch-sensitive surfaces, continues this work in a world where touch screens and surfaces have become increasingly ubiquitous.

To reduce the amount of user interaction required in the pairing process, several proposals use contextual or location-dependent information, or natural user input such as sound or movement, for the authentication. The proposed protocols perform pairing by utilizing a shared *fuzzy* secret that the devices only know approximately. This shared secret may be extracted, for example, from ambient audio or radio signals [5–7] or from simultaneous sensing of user actions. As an example, Mayrhofer [8] as well as Kirovski et al. [9] derive a shared fuzzy secret from the user shaking or moving the two devices together. In such protocols, a major challenge is to establish an exact shared cryptographic key starting from two noisy and, thus, slightly differing measurements of sensor data. In this paper, we overview the existing methods for establishing a shared secret from noisy input and propose a new protocol, which is a single-round, time-based variant of the existing commitment protocols.

☆ This is an extended version of a paper that appeared in PerCom 2014 (Sethi et al. (2014)).

* Corresponding author.

*E-mail addresses:* markku.antikainen@aalto.fi (M. Antikainen), mohit.sethi@aalto.fi (M. Sethi), sinisa.matetic@aalto.fi (S. Matetic), tuomas.aura@aalto.fi (T. Aura).

http://dx.doi.org/10.1016/j.pmcj.2014.10.006

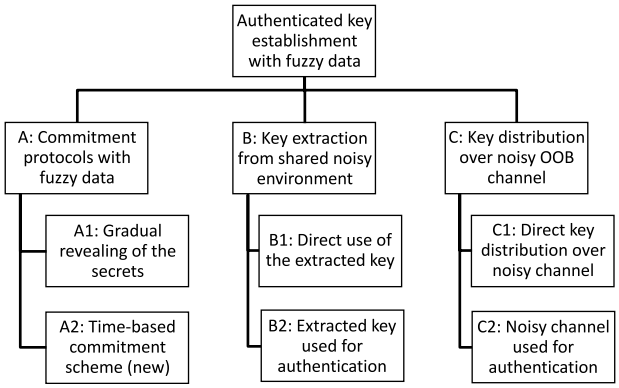



**Fig. 1.** Taxonomy of key-establishment with fuzzy data.

We start in Section 2 by providing an overview and taxonomy of existing key-establishment methods that use fuzzy shared secrets for authentication. In Section 3, we derive a new time-based variant of the commitment-based key establishment. Section 4 presents the novel human-assisted pairing mechanism that uses synchronized drawings as the fuzzy shared secret. We implement a prototype and evaluate the security of this pairing mechanism. In particular, we evaluate various metrics for comparing the drawings and find an efficient metric based on string edit distance. Section 5 discusses the security of the proposed scheme. Finally, Section 6 concludes the paper.

## 2. Background

This section explains the threat model used throughout the paper (Section 2.1) and provides a survey and taxonomy of existing solutions for *authenticated key-establishment with fuzzy data* (Section 2.2). We consider protocols that tolerate errors in the data from which they derive a shared secret key.

### 2.1. Threat model

The use case for authenticated key-establishment with fuzzy data is relatively simple: two devices that communicate over an insecure network, such as a wireless link, need to establish a shared secret key. The devices also have access to an out-of-band (OOB) channel, which has a high error rate. The errors may be caused, for example, by external noise on the channel or by differences in the sensory inputs of the two devices. The noisy channel may be used either to send messages between the devices or to receive the same external signal. A wide range of key-establishment protocols have been proposed based on different types of noisy channels such as ambient sound or radio signals [5,10,11].

The goal of a pairing mechanism is to establish a shared secret key that can be used to secure subsequent communication over the insecure network. The security analysis of this paper assumes a powerful Dolev–Yao type attacker on the insecure network [12]. The noisy OOB channel, on the other hand, is assumed to provide some inherent protection for the confidentiality and integrity of data. This may be, for example, because the channel is location limited and under the direct supervision of the user. The attacker aims to subvert the authentication and access the subsequent communication over the insecure network either by passively eavesdropping it or by actively impersonating one or both of the devices.

### 2.2. Related work

In the following, we divide the related work into the categories illustrated in Fig. 1.

#### 2.2.1. Commitment-based protocols

Commitment-based protocols (category A in Fig. 1) are the closest equivalent to the new protocol proposed in this paper. The earliest protocol to use commitment for protecting the integrity of communication against man-in-the-middle attacks was the interlock protocol by Rivest and Shamir [13]. While it makes use of encryption rather than hash functions, the principle is similar to the modern protocols. Later, commitment-based protocols were developed for authentication with a short one-time secret. As we will see, these can be modified to work equally well with a fuzzy (i.e. noisy) secret.

These protocols begin with an unauthenticated key exchange, such as Diffie–Hellman or public-key encryption without certificates. The resulting strong, fresh (but initially unauthenticated) shared key $k$ is then authenticated to prevent spoofing and man-in-the-middle (MitM) attacks. For this purpose, the devices share a short secret $p$, such as a 6-digit number, distributed over an out of band channel. In the *commitment phase* of the authentication, the devices exchange cryptographic commitments to the values of the short secret and the fresh shared key. A commitment is a cryptographic hash $H(k, p, r)$, where $r$ is a fresh random number needed for blinding the secrets. In the *opening phase*, both sides reveal their random numbers $r$ and verify that hash sent by the other party in the first phase is correct.

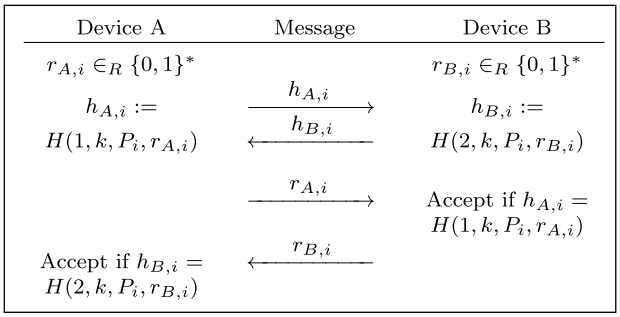



**Fig. 2.** *i*th round in MANA III protocol when authenticating a Diffie–Hellman key *k*.

The security of the commitment schemes described above depends critically on the timing of the messages: both commitments must be received before either side reveals its $r$. Otherwise, a man-in-the-middle attacker could find the value of the short secret $p$ within seconds by brute-force trial. One way to enforce the time order is to require the user to confirm the completion of the commitment phase on both devices before moving on to the opening phase. Unfortunately, it is difficult to design an easy-to-use interface where the user could not take a shortcut and confirm the completion without actually checking both devices, which would expose the protocol to MitM attacks.

One clever solution to enforcing the time order of the phases is the gradual release of the secret, as in the MANA III variant [14–16]. In this protocol, the two devices *A* and *B* reveal the one-time secret $p = p_1|p_2|\cdots|p_n$ piecewise in $n$ rounds and perform a separate commitment and opening for each round. Fig. 2 shows one protocol round. By spoofing one of the commitments $H_{x,i}$, the MitM attacker can learn and replay one part $p_i$ of the short secret. However, it will not be able to continue to the next round if the spoofed commitment was not correct.

The reason why we have given this much space to the commitment protocols is that the short secret can be replaced with fuzzy data (A1 in Fig. 1). Varshavsky et al. [11] demonstrate the use of a MANA-III-like protocol for this purpose. The fuzzy shared secret in this case is a recording of the shared ambient radio environment of two devices that are in close proximity. A similar mechanism is proposed by Soriente et al. [17] where the user inputs the fuzzy data by simultaneously pressing buttons on the devices. Protocols in these proposals differ from the standard MANA III in that the fuzzy secrets $p_i$ are revealed along with the random values $r_i$. Revealing the secrets is necessary since the devices initially know slightly different versions of the fuzzy data and, without the exact information, neither device could verify the hash of the other. Each device compares the observed and received values of $p_i$. For this purpose, they need a suitable distance metric and threshold for acceptance.

A limitation of the gradual revealing methods (A1) is that the fuzzy shared secret $p$ must be partitioned into the round secrets $p_i$ so that (1) the partitions of the data on the two devices correspond to each other, (2) entropy is divided roughly evenly between the parts $p_i$, and (3) the parts are independent from each other so that the earlier parts $p_i$ do not reveal information about the later parts $p_j$, $j > i$. Such partitioning of the data is not easy to achieve with all types of fuzzy data, especially if the data has no natural time axis or if it is recorded without accurate time information. Varshavsky et al. solved the correspondence issue by synchronizing the input intervals on the two devices. In our case, we record the fuzzy data with only rough synchronization at the end and, in the most CPU-saving versions of our scheme, without recording any timing data, which makes is difficult to partition the data into matching subintervals on the two devices without either device first seeing the full inputs from the other side. To avoid the above limitations, we propose a new type of commitment scheme, *time-based commitment scheme* (A2), that uses only one commitment round and does not require that the secret is split into parts. We discuss details of this protocol in Section 3.

### *2.2.2. Key extraction from shared noisy environment*

Another class of device-pairing protocols (B in Fig. 1) extracts the shared secret key from fuzzy data such as observations of a shared environment. If the created secret key has sufficient entropy to prevent offline brute-force attacks, it may be used directly as an encryption key (B1). Most of the protocols described in the literature appear to be of this type (see Table 1). However, as the entropy of the fuzzy input typically is uncertain, it would be safer to use the extracted key as a transitory secret for authenticating a strong key exchange such as Diffie–Hellman (B2). While there are only a few protocols of type B2 in the literature [26,22], all protocols of type B1 can be easily converted to the safer type B2. Using the potentially weak shared secret for authenticating a stronger one is also one of the design principles of our protocol.

The challenge in the key extraction from fuzzy data is that the keys computed by the two devices must be exactly the same even when the inputs are noisy. Typical cryptographic key-generation mechanisms, such as hash functions, aim for the exact opposite, i.e. even a one-bit difference in the inputs will result in completely unrelated keys. The solutions to the key extraction are called *fuzzy cryptography*, which has been developed mainly for key derivation from biometric data [22,20,27]. Several protocols using shared sensory input such as movement or ambient audio have then been based on the same techniques [9,21,5,10]. While the details of the protocols vary, the main idea is to exchange error correction codes – in plaintext over the insecure network – for the secret fuzzy data, so that differences in the fuzzy inputs can be corrected.

**Table 1**
Comparison of the proposed fuzzy key-establishment protocols.

| Protocol | Source of the shared secret | Class |
|---|---|---|
| Amigo [11] | Radio environment | A1 |
| BEDA (button-to-button) [17] | Timing of button presses | A1 |
| D–H and Interlock* [18] | Accelerometer data | A1 |
| AdhocPairing [5,6,10] | Audio | B1 |
| Generating key from accelerometer data [19] | Accelerometer data | B1 |
| Candidate key protocol [8,18] | Accelerometer data | B1 |
| Proximate [7] | Radio environment | B1 |
| Feeling-is-believing [20] | Biometric data (fingerprint) | B1 |
| Secure communication based on ambient audio [21] | Ambient audio | B1 |
| Secure Ad-hoc Pairing with Biometrics (SAfE) [22] | Biometric data | B2 |
| Human-Assisted Pure Audio Device Pairing (HAPADEP) [23] | Device generated secret sent over lossy audio-OOB channel | C1 |
| BEDA (other variants) [17] | Device generated secret sent over lossy user-controlled channel | C2 |
| Loud-and-clear [24] | Device generated secret sent over lossy audio-OOB channel | C2 |
| Seeing-is-believing [25] | Device generated secret sent over lossy visual-OOB channel | C2 |

The security of fuzzy cryptography depends on the error correction code not leaking any information about the secret fuzzy data [28–30]. This is most critical in systems that use long-term fuzzy secrets such as biometrics, but also affects the security of short-term secrets such as environmental input. It may be difficult to estimate the actual entropy of the fuzzy input and, thus, the safe amount of error correction data. Problems may arise, in particular, if the attacker has auxiliary information such as statistical knowledge about the input data, which together with the error correction code might reveal more than expected [28].

Other pairing protocols derive an encryption key directly from sensor data without using fuzzy cryptography. Here, we analyze two such protocols, namely CKP [18] and SAPHE [31]. In Candidate Key Protocol (CKP), devices generate an encryption key from sensor data by extracting feature vectors from the sensory inputs. Each feature vector is hashed with a standard hash function and the hashes are sent to the other device and compared. Differences in the sensory inputs are handled by simply ignoring the feature vectors that are not equal. Only the feature vectors with identical hash values are used to create the encryption key. There is, however, no clear rule for how large fraction of the feature vectors can be ignored without endangering security. Another problem is that two feature vectors that are close to each other but not identical are ignored in the same way as ones that are far apart from each other. Thus, the protocol requires low environmental noise and accurately calibrated sensors to produce enough identical vectors between the devices. Simple Accelerometer -Based Wireless Pairing with Heuristic Trees (SAPHE [31]) similarly generates keys directly from the sensor inputs. Differences in the inputs are handled with a brute-force search for the differing bits in the keys. The search is made efficient by public threshold values that enable the devices to determine which key bits are most likely to be wrong. Unfortunately, the threshold values also leak a significant amount of information about the key, giving roughly the same performance gain to the brute-force attacker as to the honest parties. In conclusion, the direct key extraction methods do not seem as secure or reliable as fuzzy cryptography.

#### *2.2.3. Secure but noisy OOB channel*

For completeness of the discussion, it is instructive to consider also protocols where the end nodes communicate with each other over a secure but noisy out-of-bad channel (C in Fig. 1). The OOB-channel may either be used directly for key distribution (C1) or for authentication of a key exchange that takes place on the insecure network (C2). Protocols that use an out-of-band channel directly for key distribution are relatively simple and typically secure if the attacker cannot eavesdrop the OOB-channel and if active attacks are detected. Human Assisted Pure Audio Pairing (HAPADEP [23]) offers a good illustration: an encryption key with error correction is sent over an audio channel, which the human user monitors. Active attacks are detected as unexpected sound from an external source. There are also many ways to use the noisy out-of-band channel for verifying the result of a key exchange (e.g. unauthenticated Diffie–Hellman). For example, Soriente et al. [17] simply authenticate the keys by sending a message authentication code over the OOB channel. Other protocols rely on the user for comparison of the noisy signals, such as audio in Loud-and-clear [24] and 2D bar codes in Seeing-is-believing [25]. For the purposes of this paper, these protocols offer examples of the available OOB channels but no complete solution because, in the scenarios that we consider, the user produces rather than compares the two fuzzy inputs.

## 3. Key-establishment with a shared fuzzy secret and time-based commitment

This section describes a new variation of the commitment-based key establishment protocol for use with fuzzy secrets. The need for the new protocol arose when implementing our device-pairing scheme based on synchronized drawing (Section 4). In particular, we found no reliable way to measure the entropy of the drawings. Without knowing the entropy of the fuzzy input, we could not be certain that the error correction codes in fuzzy cryptography (Section 2.2.2) would not leak any information about the secret to an attacker who may have a good statistical model of the drawings. We also looked at

**Table 2**
Summary of variables and functions at device $A$ (similar for $B$).

| | |
|---|---|
| $k$ | Diffie–Hellman shared secret |
| $v_A$ | $A$'s version of the fuzzy data i.e. noisy input to device $A$ |
| $r_A$ | Random number generated by $A$ |
| $H$ | Cryptographic hash function |
| $A$ | Identifier of device $A$ |
| $E_k$ | Encryption with a key derived from $k$ |
| $D$ | Distance function for the fuzzy data |
| $d$ | Upper limit for acceptable distance between the two fuzzy inputs |
| $t_{A,0}$ | Local time of device $A$ at the reference event |
| $\Delta_1$ | Delay allowed between sending the commitment and receiving the other device's commitment |
| $\Delta_2$ | Safety period between the last time to receive commitments and the time to open one's own |

the MANA III variant protocol (Section 2.2.1) and were met with a difficult tradeoff in deciding the number of rounds for the gradual revealing of the fuzzy secret. If the number of rounds is small, one needs to partition the input data into the small number of pieces, which must all have high entropy and be independent of each other. On the other hand, if the number of rounds is high, the communication overhead grows and the partitions made by the two devices need to be accurately synchronized. These practical problems in implementing the existing protocols pushed us to look for a new solution where the inputs are communicated and compared in one piece and the security depends only on the total entropy of the fuzzy data. The fewer security-critical parameters, such as the number of rounds or length of the error-correction code, there are, the easier it is to deploy the protocol and the safer to adapt it to new applications.

In the following, we first describe a straight-forward commitment protocol where the user is responsible for signaling the transition from the commitment phase to the opening phase. The explicit user interaction will then be replaced with an implicit time-based mechanism. Explained this way, the protocol appears almost trivial, yet compared to those in the literature, it solves the problems explained above.

The commitment-based key establishment consists of the following steps:

1. Asymmetric-key exchange: e.g. Diffie–Hellman.
2. Noisy shared input to both devices: e.g. our synchronized drawing scheme or shared environmental input.
3. Commitment phase: the devices exchange cryptographic commitments to the shared secret key from step 1 and the fuzzy secret data from step 2.
4. Opening phase: the devices open the commitments and verify their correctness, as well as exchange and compare their fuzzy data.

The commitments are of the form $H(A, v_A, r_A, k)$, where $A$ is the device's identifier, $v_A$ is the device's version of the fuzzy secret, $r_A$ is a fresh random number, $k$ is the shared secret key produced in step 1, and $H$ is a cryptographic hash function (see Table 2 for the summary of variables). The opening of the commitments means that each device sends its random number $r_A$ and its version of the fuzzy secret $v_A$ to the other over an unauthenticated channel. We encrypt the opening phase, which will not prevent a man-in-the-middle attack but will prevent a passive attacker from gathering statistical information about a specific user's fuzzy inputs. The commitments are verified by recomputing the other side's hash value and by comparing it with the value received earlier.

The comparison of the fuzzy data from the two devices is based on some distance function $D(v_A, v_B)$ and a threshold distance $d$ under which the distance must be. The definition of these depends on the nature of the fuzzy data. We will discuss this more closely in Section 4 in relation to the synchronized drawings.

The protocol provides secure authentication for the shared key $k$ under the assumption that the devices move to the opening phase only after the commitment phase has been completed. That is, the devices should reveal their values $v_A$, $r_A$, $v_B$ and $r_B$ only after both devices have received the other's commitment. If this were not the case, an active attacker would be able to replay the fuzzy secret revealed by one device in the opening phase to the other device that is still in the commitment phase. The rest of this section revolves around *how to ensure the strict time separation of the two protocol phases*.

It is instructive to start by looking at our initial solution (Fig. 3), in which the user has the responsibility of enforcing the time separation. After each device has sent and received a commitment, it waits for the user's approval to continue. The user should give this approval only when both devices have reached the same state. If users can be educated to follow this policy, the protocol is secure. However, if an impatient user presses on the "continue" button on one device without checking the state of the other, the security of the authentication is compromised and a man-in-the-middle attack may go undetected.

There are ways of reducing the risk of the rushing-user behavior [32]. First, the user interface can be designed to encourage (although not to enforce) the correct practice. The user can be given instructions, and when there is no attacker present, the devices can detect the user's carelessness about giving approval too early and admonish him. This type of feedback has been shown to improve user behavior in security critical tasks [33]. Second, when the attacker is successful, only one of the devices will appear to complete the protocol successfully. The other device will fail because of false input or timeout, and the user may detect this. (Indeed, the original MANA III protocol asks the user to confirm the completion of the protocol rather than approve the move from the commitment phase to the opening phase.) Nevertheless, the risk of a security failure can be reduced further by avoiding the explicit user involvement, as we will explain below.

| Device A | Message | Device B |
|---|---|---|
| 1. Diffie-Hellman key exchange | | |
| | **1a**: $g^x$ → | |
| $k := g^{xy}$. | ← **1b**: $g^y$ | $k := g^{yx}$. |
| 2. Fuzzy data | | |
| $v_A :=$ noisy input **2a**. | | $v_B :=$ noisy input **2b**. |
| 3. Commitment | | |
| Fresh random number $r_A \in_R \{0,1\}^{128}$. Compute commitment $c_A := H(A, v_A, r_A, k)$. | | Fresh random number $r_B \in_R \{0,1\}^{128}$. Compute commitment $c_B := H(B, v_B, r_B, k)$. |
| | **3a**: $c_A$ → | |
| Display to user "commitment received". | ← **3b**: $c_B$ | Display to user "commitment received". |
| 4. Opening the commitments | | |
| Wait for user approval to continue. | | Wait for user approval to continue. |
| | **4a**: $E_k(A, v_A, r_A)$ → | |
| | ← **4b**: $E_k(B, v_B, r_B)$ | |
| Accept $k$ if $A \neq B$, $c_B = H(B, v_B, r_B, k)$, and $D(v_A, v_B) \leq d$. | | Accept $k$ if $B \neq A$, $c_A = H(A, v_A, r_A, k)$, and $D(v_B, v_A) \leq d$. |

**Fig. 3.** Basic protocol: explicit synchronization by user.

After these preliminaries, the time-based commitment scheme becomes easy to describe. The idea is inspired by the TESLA [34] and Guy Fawkes [35] broadcast authentication protocols, in which the sender opens a kind of commitment at a fixed time. However, these previous schemes use the time-based opening of the commitments directly for message authentication, while we use it for authenticating key establishment and in combination with fuzzy secrets.

The final protocol (Fig. 4) uses the local clocks and one synchronized *reference event* for synchronizing the movement from the commitment phase to the opening phase. The reference event in our scheme is the *end of the user input*, which in the synchronized drawing tends to be almost simultaneous between the devices (see Discussion in Section 5). We denote the reference event time, read from the local clock of device *A*, by $t_{A,0}$. Device *A* sends its commitment message 3a some time after $t_{A,0}$. Device *A* insists on receiving the other device's commitment by the time $t_{A,0} + \Delta_1$; otherwise, the authentication fails. Device *A* opens its commitment at the time $t_{A,0} + \Delta_1 + \Delta_2$, where $\Delta_2$ allows for a safety period between the commitments and their opening. Naturally, device *B* performs the equivalent operations using its own local clock. We have modeled this protocol and its security properties in [36].

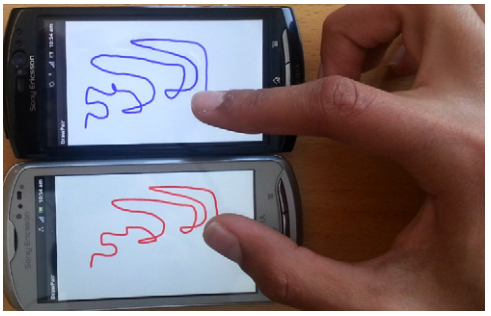
The above protocol depends critically on the timing, and the time constants need to be fixed in the specification. We have set the time period $\Delta_1$ to 200 ms, which is sufficient for message transfer over any modern wireless or wired network if the network connection has been established in advance. We set $\Delta_2$ at the fairly high value 500 ms because this constant allows for error in the synchronization of the reference event on the two devices. Overall, the authentication process takes less than one second after the end of the user input.

## 4. Synchronized drawing and comparison metrics

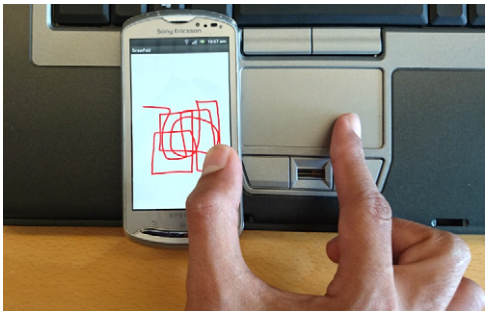
This section describes a novel type of shared noisy input for devices with a touch surface. The basic idea is quickly explained: the user draws the same figure synchronously on the two devices using two fingers of the same hand. Typically, users choose to use the thumb and index finger, as illustrated in Fig. 5. The picture also shows sample drawings on two devices. This shared input is used as the fuzzy shared secret data for the secure key establishment. The method can be applied to smart phones, laptop touchpads, touch-sensitive control panels, drawing pads, and certain types of mice. The shared input can be used, for example, to pair two smart phones or to pair a smart phone, tablet or a separate touch pad to a laptop computer. Our prototype implementation works for any pair of a smart phone, tablet and laptop computer.

| Device A | Message | Device B |
|---|---|---|
| 1. Diffie-Hellman key exchange | | |
| | **1a**: $g^x$ $\longrightarrow$ | |
| $k := g^{xy}$. | $\longleftarrow$ **1b**: $g^y$ | $k := g^{yx}$. |
| 2. Fuzzy data | | |
| $v_A :=$ noisy input **2a**. $t_{A,0} :=$ A's local time at the end of input. | | $v_B :=$ noisy input **2b.** $t_{B,0} :=$ B's local time at the end of input. |
| 3. Commitment | | |
| Fresh random number $r_A \in_R \{0,1\}^{128}$. Compute commitment $c_A := H(A, v_A, r_A, k)$. | | Fresh random number $r_B \in_R \{0,1\}^{128}$. Compute commitment $c_B := H(B, v_B, r_B, k)$. |
| | **3a**: $c_A$ $\longrightarrow$ | |
| Abort if message 3b received at time $> t_{A,0} + \Delta_1$ by A's local clock. | $\longleftarrow$ **3b**: $c_B$ | Abort if message 3a received at time $> t_{B,0} + \Delta_1$ by B's local clock. |
| 4. Opening the commitments | | |
| Wait until time $t_{A,0} + \Delta_1 + \Delta_2$. | | Wait until time $t_{B,0} + \Delta_1 + \Delta_2$. |
| | **4a**: $E_k(A, v_A, r_A)$ $\longrightarrow$ | |
| | $\longleftarrow$ **4b**: $E_k(B, v_B, r_B)$ | |
| Accept $k$ if $A \neq B$, $c_B = H(B, v_B, r_B, k)$, and $D(v_A, v_B) \leq$ d. | | Accept $k$ if $B \neq A$, $c_A = H(A, v_A, r_A, k)$, and $D(v_B, v_A) \leq$ d. |

**Fig. 4.** Final protocol: implicit time-based synchronization.



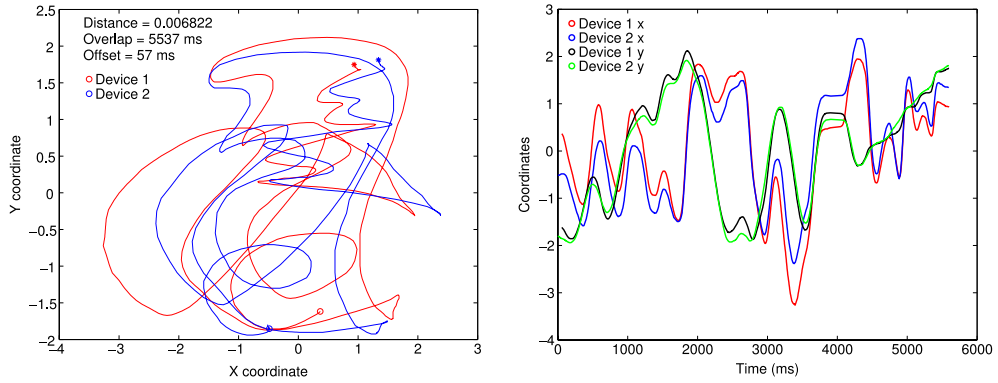
(a) Pairing two phones.



(b) Pairing a phone with a laptop.

**Fig. 5.** Synchronized drawing for device pairing.

While the basic principle is simple, the details of the scheme require careful work. In order to compare the drawings, we need to define a *distance metric* and a *threshold value* for accepting the drawings as a match. The same metric and threshold should work for all devices and users, so that the authentication can be used without a training period and without setting parameters. Our experiments showed that such metrics and threshold values can be found easily and that the comparison is not sensitive to minor changes in the threshold value or other parameters of the algorithms. We experimented with various metrics in order to find one that is inexpensive to compute and close to optimal in terms of accuracy. Some of the possible metrics will be explored in the following sections, and in Section 4.5, we take a closer look at how to choose the best one.




(a) Centered and scaled drawings.

(b) $x$ and $y$ coordinates as function of time.

**Fig. 6.** Location metric for comparing the drawings.

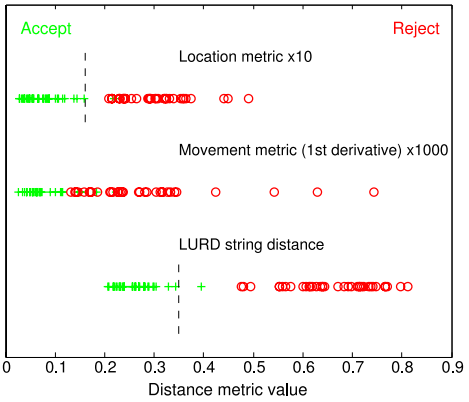



**Fig. 7.** Comparison of distance metrics for drawings. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

### *4.1. Location metric*

The most obvious metric is based on *location as a function of time*. This is because the application programming in interfaces (API) on touch-screen devices produce events with $x$ and $y$ coordinates. Fig. 6(a) shows two matching drawings. Since different touch surfaces differ in their dimensions, pixel density, aspect ratio and frequency of the sensed movements, we (1) center the drawings so that the mean value of the $x$ and $y$ coordinates is zero, (2) scale them so that the mean distance from the center along each axis is one unit, and (3) interpolate the location data to one-millisecond frequency.

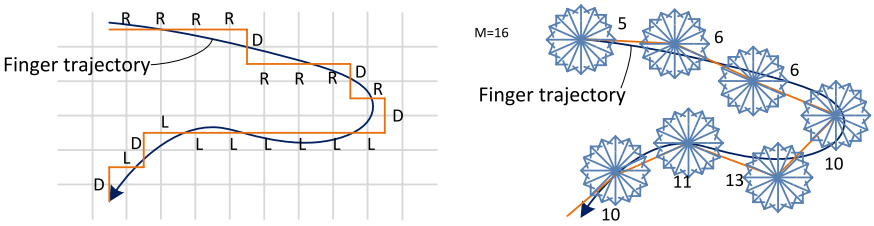
We also experimented with rotating the drawings, in case the two drawings are at an angle. A linear search at the granularity of 20° finds the approximate angle, after which a bisection method can determine the exact rotation. However, the rotation proved unnecessary in practice, except for 90°, 180°, and 270° turns because the touch-screen and pad devices are typically rectangular and are placed next to each other with random sides touching.

Fig. 6(b) represents the same drawings as function of time. The graphs have been aligned on the time axis to minimize their distance by the chosen metric. The optimum time shift is very close to the value that also gives the highest cross-correlation between the two drawings. Since cross-correlation is more efficient to compute than the distance metrics, it should be used in practical implementations for aligning the two inputs. In this case, the offset between the drawings is 57 ms from the start of the drawings, which is quite a typical magnitude. Fig. 6(a) also indicates the start and end of the overlapping time period, over 5 s in this case. The location-based distance metric is computed from the centered, scaled and interpolated data as the Euclidean norm of the difference between the $x$ and $y$ coordinate vectors:

$$D(\text{drawing}_A, \text{drawing}_B) = \frac{1}{n}\left(\sum_{i=1}^{n}((x_{A,i} - x_{B,i})^2 + (y_{A,i} - y_{B,i})^2)\right)^{1/2}.$$

Above, the norm is divided by the length of the drawing to avoid bias towards shorter drawings. The length of the drawings was taken into account separately by setting a minimum drawing length (4 s) and a maximum difference for the lengths of the drawings (500 ms).

Fig. 7 shows location-metric values from an initial experiment plotted on a line. The location metric clearly separates pairs of drawings that have been drawn synchronously with two fingers (green cross) from pairs of drawings that have no such




(a) LURD string encoding of the drawing. (b) Angle string encoding of the drawing.

**Fig. 8.** String encodings of the drawing. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

relation (red circle). A threshold value can be easily chosen so that there are no false positives and no or few false negatives for the authentication. In the experiments, there were no false positives, and false negatives were caused by obvious user mistakes such as moving one finger for prolonged periods outside the touch-sensitive area of the surface. The histogram in Fig. 9(a) shows the distribution of the metric values for a different dataset.

### 4.2. Movement metrics

The distance metric could also be based on the movement, i.e. speed and direction of drawing as a function of time. In some applications, such as handwriting recognition, good results have been achieved by comparing the direction of strokes. We experimented with metrics based on the first and second derivatives of the location, i.e. velocity and acceleration, but were not able to derive useful metrics from these. As can be seen from Figs. 7 and 9(b), there is no threshold value that would separate the green and red plots.

The main reason for the motion metric not working is that the event-based touch-screen APIs on the smart phones (e.g. *MOTION_EVENT* on Android and *touchesMoved* on IOS) do not give frequent enough readings for accurate calculation of the velocity or acceleration of the finger. While one could intuitively expect velocity (speed and direction of moment) to most accurately represent the user drawing actions, in practice, the metric that uses the measurement data the least-processed form seems to work best. Here the devices produce location data, i.e. events with $x$ and $y$ coordinates. In comparison, the pairing schemes based on shaking the devices together work well with acceleration metrics because the sensor data for these schemes is taken from an accelerometer.

Another potential error source is that the touch-screen APIs do not provide accurate timing of the events, which is not needed for most touch-screen applications. However, as we will explain in the following, a coarser direction-based metric that ignores the exact timing of the movement proved to work well.

### 4.3. LURD string edit distance

While the location-based metric is otherwise acceptable for our purposes, it is fairly expensive to compute on a mobile device. The main cost is to find the exact time offset that best aligns the two drawings. For this reason, we wanted a distance metric for the drawings that takes little computing power, does not require accurate time synchronization or computing the exact time offset between the drawings, and nevertheless accurately compares noisy drawings. Shape recognition algorithms that are based on approximate string matching are known to have these properties [37,38]. These algorithms encode the drawings into a string, which can be then compared using approximate string comparison algorithms such as the Levenshtein distance and other, generalized edit distances. The edit distance is the minimum cost of the insertion, deletion and substitution operations needed to transform one of the strings to the other.

Our string-based metric is called the *LURD string edit distance*. Its basic idea is to convert the movement of the finger into a string of letters (L = left, U = up, R = right, D = down) and to compute a string distance between the two drawings. The conversion into a string is done as follows. The area covered by each drawing is divided into a grid of $32 \times 32$ squares. Each movement of the finger that crosses at least one grid line is mapped to a string that has one character for each crossed grid line. The characters indicate the direction of the crossings. For example, if the movement crosses one vertical grid line towards right and then one horizontal grid line in the downward direction, the string will be "RD". For efficiency, the implementation does not try to accurately compute the order of crossings when they happen very close to the junction of two grid lines; in such cases, the crossing of the vertical grid line is prioritized. The substrings from the recorded movement events are concatenated to represent each drawing. Fig. 8(a) illustrates the string encoding; the red line shows how encoding approximates the original drawing.

The two strings, one for each device's observation of the drawing, are then compared by computing their edit distance. In the edit distance, weight 1 is given to the deletion or insertion of a symbol, 1 for the substitution of an adjacent direction, and 2 for the substitution of the opposite direction. Initially, we used uniform weights [36], but the results were slightly improved, with no increase in the computational cost, by making the cost of the substitution dependent on the angle between the movements.

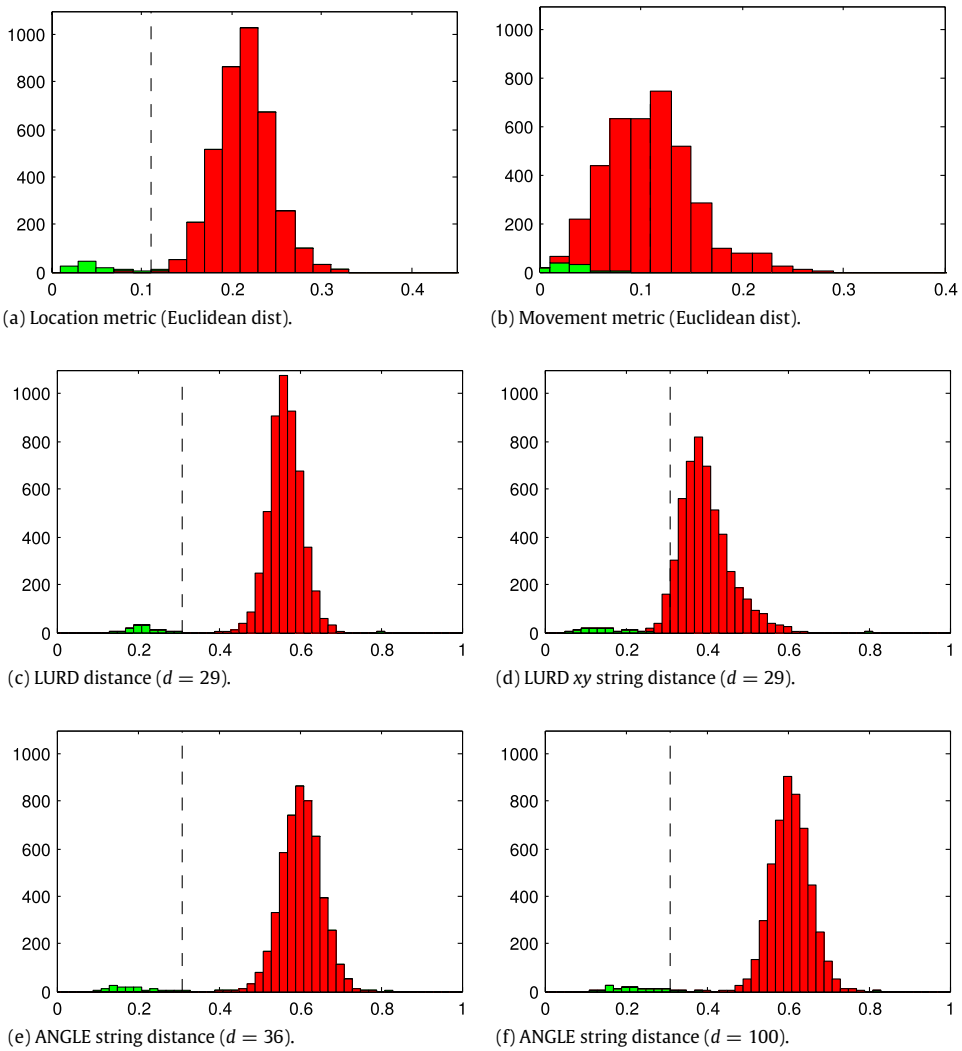



**Fig. 9.** Distribution of metric values for pairs of matching drawings (green) and non-matching drawings (red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

As can be seen in Fig. 7 and in the histogram in Fig. 9(c), the string distance metric satisfactorily separates the synchronized drawings from pairs of unrelated drawings. Moreover, it automatically synchronizes the two drawings by inserting characters to the beginning of the one that starts later, so that no time synchronization needs to be done. There are simple and efficient algorithms for computing such string distances [39], and these proved to work well on the mobile devices. Since the cost of these algorithms grows quadratically with the length of the strings, the string encodings of the drawings needs to be less than about 1000 symbols long. This is achieved by keeping the granularity of the grid relatively coarse (e.g. $32 \times 32$) and by limiting the user input to 5–10 s.

In a simpler version of the metric, the movement in the $x$ and $y$ directions is encoded separately to substrings. This *LURD xy metric* (Fig. 9(d)) did not work well. This can be explained by the loss of information about the interleaving of the vertical and horizontal movements. We experimented with many other variations of the basic LURD string edit metric, and the LURD $xy$ metric is given here as an example of what the histograms looked like for the less good variants.

Since the LURD metric is both efficient to compute and produced zero false positives in all experiments, it was the obvious choice for the final implementation. We wanted, nevertheless, to understand what trade-offs are made and, thus, continued to experiment with its generalized versions.

### *4.4. Angle string metric*

The angle metric is a generalization of the LURD metric. Instead of encoding the drawing as fixed-length movements in four possible directions, more directions are allowed. For example, in Fig. 8(b), there are $M = 16$ possible directions. At

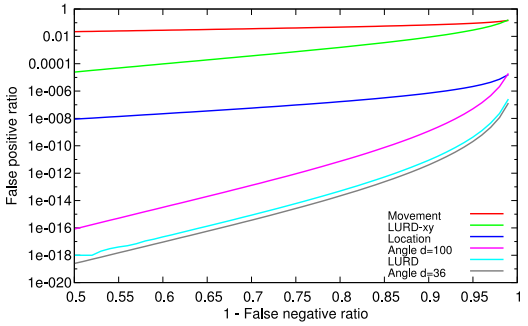



**Fig. 10.** Estimated FPR as a function of FNR (by varying the distance threshold value).

each step, the direction that most closely follows the observed drawing is selected. Thus, the drawing is encoded as a string with an alphabet of $M = 16$ symbols. The red line shows how the encoding approximates the original drawing.

By experimenting, we found that there is an optimum fixed step length that best captures the similarities and differences of the two drawings. Both too short and too long steps make the metric less accurate. After making this observation, we found the optimal fixed step length ($d = 36$ in a scaled coordinate system) by exhaustive search and using the methodology described in the next subsection. A histogram of the metric values is shown in Fig. 9(e). The angle string metric is overall the most accurate metric that we were able to develop. Since the smaller steps create longer string encodings and the cost of computing the edit distance grows quadratically with the string length, there is no reason to use steps shorter than the optimum. The accuracy is not very sensitive to small changes in the step length, but when it grows to 2–3 times the optimum, then the green and red distributions of the histogram start to get visibly closer to each other (e.g. Fig. 9(f)). On the other hand, increasing the alphabet size, i.e. the number of angles $M$, can only lead to more accurate metrics, and there is no computational cost associated with the higher values of $M$. Experimentation showed that there is no observable improvement to accuracy when using more than $M = 32$ different angles, which is the value used in the reported experiments.

### 4.5. Comparison of the metrics

In the previous sections, we have compared the metrics visually, by looking at plots of their distributions. This method enables us to weed out bad metrics where the green and red distributions clearly overlap. However, as can be seen in Fig. 9, visual comparison between the better metrics is not easy. The standard numerical measures of accuracy are the FPR and FNR, but in most cases the threshold value can be chosen so that the *observed* FPR and FNR values are zero. Therefore, we needed a more systematic method for understanding how good the metrics are and for choosing the best one.

The data in Fig. 7 (and also in Fig. 11) is from our initial experiment with six users, each making approximately six or more matching pairs of drawings. The data in the histograms Fig. 9 is from a second, independently conducted experiment, again with six users, each drawing 12–17 matching pairs of drawings. The pairs of different drawings were created from the same data by taking some or all pairs of non-matching drawings. The users were university students and staff on two different campuses. The differences in the threshold values used in the two studies are due to the grid and step sizes being optimized for the second study. The results of the first study are nevertheless interesting because a variety of devices were used (phones, tablet, laptop touchpad) in variable combinations. In the latter study, the devices were an Android touch screen phone and an Apple laptop with a touchpad. In the following, we will use data from the second study.

From the histograms, we can see that the metrics for both the true positive and true negative cases are approximately normally distributed. It can be argued informally that the central limit theorem applies here. The variations in the metric values are caused by the sum of small random events along the line drawings. Assuming the normal distribution, we can estimate the FPR and FNR for any threshold value. Moreover, we can compute the threshold value and thus FPR, even when it is very small, for different values of FNR. This allows us to compare the metrics even when there is no obvious visual difference in the histograms.

The comparison is shown in Fig. 10. The values that are interesting in practice are at the right end of the curves (1-FNR $>$ 0.95). The figure confirms our early observation that the LURD metric performs surprisingly well (for FNR $=$ 1%, FPR $=$ 2.6E$-$07). It is clearly better than the location metric, which we took as the initial baseline (for FNR $=$ 1%, FPR $=$ 1.6E$-$05). Since the angle metric is a generalization of LURD, it is naturally more accurate (for FNR $=$ 1%, FPR $=$ 1.3E$-$07), but not so much that it would justify the higher computational cost. The LURD and angle metrics were the most accurate metrics found in our exploration. They are both based on encoding the drawing as a string and then computing a string edit distance. No other types of metrics came even close to them in accuracy.

It should be noted that some variations of the edit-distance metrics have a visibly skewed (non-normal) distribution. The LURD *xy* metric (Fig. 9(d)) is shown as an example. We have not been able to explain the skewness, although we expect there to be a simple mathematical explanation. However, since the skew in all observed cases was towards the right, using a normal distribution instead will produce conservative estimates of the FPR.

## 5. Discussion

We have described both a key-establishment protocol based on a fuzzy shared secret and a novel method of creating such secrets from user input on touch-sensitive surfaces. We implemented a prototype for pairing smart phones, tablets, and laptops. The same method could also be used to pair other similar devices such as large touch screen tables (e.g. Microsoft Pixelsense[1]) and even Bluetooth enabled stylus pens (e.g. YuFu UltraFine Stylus[2]). Our approach provides new design options for pairing-mechanism developers. On devices where the touch surface is the only user interface, it clearly is more secure than the currently deployed approaches. One current approach for devices with limited user interface is to use no key or a fixed pin code (e.g. 0000), which also provides no security.

While the protocol was motivated by the particular type of fuzzy secret, namely freehand drawings, it can be used with other types of noisy user or environmental input. As an example, it may be possible to use user-generated audio in a similar fashion as we now use the drawings. In that case, the devices would have to detect when the audio ends and use this as the reference event for synchronizing the commitment and opening phases of the protocol.

We verified the security of the key establishment with ProVerif to avoid any crude mistakes (see [36] for the ProVerif model). The formal modeling and verification force protocol designers to define precisely the protocol's security goals and assumptions, which could lead to the discovery of security flaws. In this case, the model was developed alongside the protocol, and its main benefit was increased clarity about the reasoning behind the protocol. The formal model, of course, is an abstraction and usually does not fully cover all subtleties of the design. The rest of this section discusses the more subtle points of the protocol design.

One limitation of our protocol, in common with other commitment schemes, is that the fuzzy data needs to be revealed when the commitments are opened. This kind of protocols require fresh fuzzy secrets for each protocol run. Thus, they will not work with long-term fuzzy secrets such as biometric data. Encryption of the opened commitments in our protocol prevents a passive attacker from collecting a large body of drawing samples from the same users and using this as statistical knowledge to guess the drawings. This gives some protection to even users who (against advice) draw repeatedly almost the same predictable pattern.

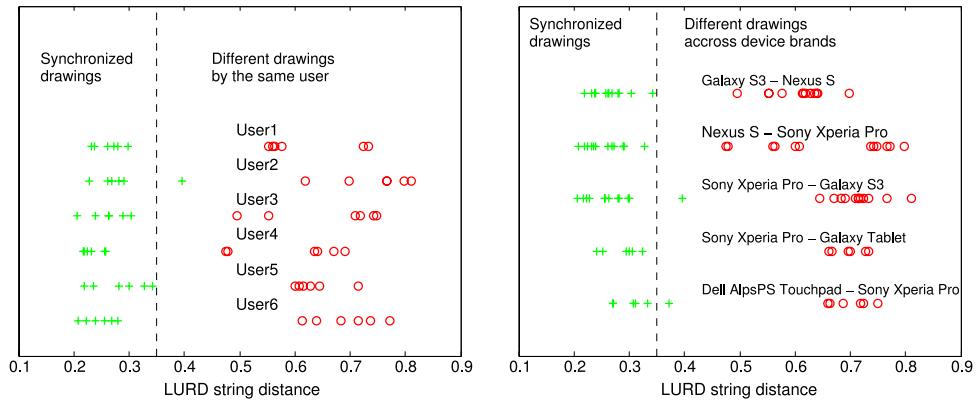
Since the security of the commitment protocol depends on maintaining the strict time order of the commitment and opening phases between the two devices, it is important to consider where this ordering might fail. In our time-based commitment opening, the user input provides a reference event that is used to synchronize the commitment and opening phases. We used the end of the user input (lifting the finger from the touch surface) as the reference event for the timing. The margin of error allowed for this synchronization is equal to the time constant $\Delta_2$, which we set at 500 ms. Normally, the end of user input on the two devices happens almost simultaneously, far from this margin of error. However, problems may arise when the user lifts one of the fingers accidentally or moves it outside the touch-sensitive area. This could cause one device to detect an end of input while the other device continues to record further data. To reduce the probability of such events, we detect the end of input only after 300 ms inactivity, which is longer than any observed accidental interruption.

Obviously, since the security relies on user actions, we cannot completely prevent erratic behavior that creates opportunities for MitM attacks. There is still the possibility of the user lifting only one finger, or lifting both for approximately 300 ms and putting them down on different sides of the time threshold. No such problems were observed in testing, but real users might behave in silly ways. For this reason, we also detect if one input is more than 200 ms longer than the other and reject authentication with warning to the user if that happens. This will not completely prevent attacks because a MitM attacker could pad the first-opened secret input with bogus data, but it will help to educate the careless users while there is no attacker present.

Another requirement for security is naturally that the fuzzy secret must have sufficient entropy. As mentioned earlier, it is difficult to measure the entropy of the drawings. Fortunately, we are using the fuzzy secret for authenticating a Diffie–Hellman key exchange, which means that $u$ bits of uncertainty reduces an active attacker's guessing probability to $2^{-u}$, and an active attack with a new protocol session is required for each guess. Uncertainty of only $u = 20 \ldots 30$ bits is typically sufficient because the unsuccessful active attacks will be detected. While it is easy to believe that a typical pair of synchronized drawings would have this much shared secret information, we employ two rudimentary criteria to check for particularly bad inputs: (1) the minimum drawing length is set to 4 s, and (2) the minimum number of direction changes in the LURD string is set to 90. These minimum values can be adjusted later based on more detailed analysis of the entropy of the typical inputs. The *ent* entropy test tool gave an upper bound of 102 bits/s for a drawing with extended length encoded as a LURD string, but such results should be treated sceptically in security applications.

There are also more traditional threats to consider, such as shoulder surfing and spy cameras. If the attacker is able to observe and copy the drawing in real time, then a man-in-the-middle attack will obviously be possible. We tried such attacks and, in ideal circumstances, it is indeed possible to follow another person's slow-moving hand accurately enough to confuse the distance metrics. The same problem of spying applies to other manual input methods used for secret input and, more generally, to most types of user input, environmental input, and location-limited channels. To make the spying of the drawings slightly more difficult, we implemented the drawing display as a fading trail (disabled in Fig. 5 for illustrative

1 http://www.microsoft.com/en-us/pixelsense/default.aspx.
2 http://hex3.co/products/yufu.




(a) Variation between users. (b) Drawings on different devices.

**Fig. 11.** Variation of the LURD distance metric.

purposes). The drawings have the advantage compared to more traditional key-based input methods, such as passwords and pin codes, that the continuous movement of the finger on an unmarked surface may be more difficult to observe and reproduce in real time from low-quality video than the discrete key presses on clearly labeled keys.

Another potential way to increase the difficulty of spying would be to ignore the broad movements of the finger and use only the high-frequency component of the user input. We were, however, unable to get any useful results from such filtering. This may be explained by the same limitations of the touch-screen APIs that were discussed in Section 4.2 in relation to the movement metrics. Further experiments with a more accurate input device would be needed to determine if the high-pass filtering can be made to work.

Experiments (e.g. Fig. 7) indicate that both the location metric and the LURD string distance are fairly robust: there is broad safety margin to prevent false positives. We also tested how these metrics vary between users (Fig. 11(a)) and different devices (Fig. 11(b)). Several different models of Android phones, a tablet computer, and a laptop-computer touch pad were used for the experiments. Six randomly selected users drew drawings for each pair of devices that were used in the experiment. The LURD string metric is particularly reliable to implement because the strings only depend on the movement path and not on the timing of the movements. The time data on the movement events from a device operating system is not necessarily accurate enough for millisecond-level correlation. This may be one explanation why the LURD string metric performs at least as well as the location metric, which takes the timing information fully into account.

When security protocols are deployed on devices which have life cycles of ten or more years, it is also important to think about future upgrade paths. For this reason, the fuzzy data sent between devices A and B is the original input data. The LURD encoding is done by the receiver before comparison, even though it would be tempting to do it on the fly during the recording. We send the original data because that allows each device to independently choose its distance metric. This makes it possible to deploy improved distance metrics without losing inter-operability with older devices.

Finally, while usability has been one of the motivating factors for our work, we do not claim that the drawing method is inherently more usable than other device-pairing schemes. First, such claims would require extensive usability studies, and the user experiments conducted for this article were focused on optimizing and evaluating the security of our device-pairing scheme. Second, usability depends on the specific application and on past user habits, so that it would be difficult to make absolute comparisons. It is encouraging, though, that we did not observe any major usability issues in our experiments, and many users even found the drawing fun to do. It can be argued that drawing with the finger, unlike for example typing, is a natural way to interact with the touch-sensitive surface. In particular, having an additional input device such as a pin pad just for security would clutter the user interface. Further design and evaluation would be needed to determine how well the drawing integrates into a complete system design. In the end, we, like most research on new key-pairing schemes, *provide secure-system and user-interface designers with new design options*. The drawing method is quite similar to shaking two devices together, and the protocols and discussion in this paper apply equally to both types of user input, but some devices are easier to shake and others easier to draw on. The drawing definitely has its advantages on relatively heavy and stationary devices like laptop computers.

## 6. Conclusion

The goal of this paper is to develop a secure and natural mechanism for pairing devices. We focus on devices with touch screens or other touch-sensitive surfaces. In our solution, the user draws synchronously the same figure on the two devices using two fingers of the same hand. In addition to surveying the existing device-pairing solutions that use noisy input to bootstrap security, this paper makes two major contributions:

First, we presented a secure key-establishment protocol for device pairing, which was motivated by the requirements of the synchronized drawings. It is a one-round variant of commitment-based authentication where the commitments are

opened at a fixed time. Unlike fuzzy cryptography and multi-round commitment protocols, our solution is not sensitive to variations in input entropy, and it does not require synchronization or exact timing data for the input events during the drawing. Instead, we require one reliable reference event that is synchronized between the devices, which in this paper is the end of user input.

Second, in addition to introducing the idea of synchronized drawing, we explore potential metrics for comparing the drawings and show that both location metrics and string edit metrics are suitable for the task. They produced no false positives and few false negatives in the authentication experiments. By comparing the distributions of the metric values for matching and non-matching drawings, we were able to select the best metrics with optimal parameters even when no false positives are actually observed. In particular, the computationally inexpensive LURD string metric proved to be very close to the best metric.

A prototype of the authentication was implemented on Android smart phones, a tablet device, and for laptops with a touch pad. Experiments indicate that, together with the commitment-based protocol, synchronized drawing could provide a new natural way of pairing devices securely.

## Acknowledgments

This work was supported by TEKES as part of the Internet of Things program of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business). We would also like to thank Nicklas Beijar for his help in refining the distance metric.

## References

[1] A. Kumar, N. Saxena, G. Tsudik, E. Uzun, A comparative study of secure device pairing methods, Pervasive Mob. Comput. 5 (6) (2009) 734–749.
[2] IEEE Std 802.15. 1-2005 Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).
[3] IEEE Std. 802.11 WG, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
[4] ZigBee specification, 2006, 344–346.
[5] S. Sigg, Y. Ji, N. Nguyen, A. Huynh, AdhocPairing: spontaneous audio based secure device pairing for Android mobile devices, in: Proceedings of the 4th International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use, IWSSI/SPMU'12.
[6] N. Nguyen, S. Sigg, A. Huynh, Y. Ji, Using ambient audio in secure mobile phone communication, in: Pervasive Computing and Communications Workshops, PERCOM Workshops, 2012, pp. 431–434.
[7] S. Mathur, R. Miller, A. Varshavsky, ProxiMate: proximity-based secure pairing using ambient wireless signals, in: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services.
[8] R. Mayrhofer, The candidate key protocol for generating secret shared keys from similar sensor data streams, in: Security and Privacy in Ad-hoc and Sensor Networks, in: LNCS, vol. 4572, 2007, pp. 1–15.
[9] D. Kirovski, M. Sinclair, D. Wilson, The martini synch, Tech. Rep., Technical Report, Microsoft Research Technical Report, MSR-TR-2007-123, 2007.
[10] N. Nguyen, S. Sigg, A. Huynh, Y. Ji, Pattern-based alignment of audio data for ad hoc secure device pairing, in: 2012 16th International Symposium on Wearable Computers, 2012, pp. 88–91.
[11] A. Varshavsky, A. Scannell, Amigo: proximity-based authentication of mobile devices, in: UbiComp 2007: Ubiquitous Computing, pp. 253–270.
[12] D. Dolev, A. Yao, On the security of public key protocols, IEEE Trans. Inform. Theory 29 (2) (1983) 198–208.
[13] R.L. Rivest, A.D.I. Shamir, How to expose an eavesdropper, Commun. ACM 27 (4) (1984) 393–395.
[14] J. Suomalainen, J. Valkonen, N. Asokan, Standards for security associations in personal networks: a comparative analysis, Int. J. Secur. Netw. 4 (1) (2009) 87–100.
[15] J.-O. Larsson, Higher layer key exchange techniques for bluetooth security, in: Open Group Conference, Amsterdam, 2001.
[16] ISO/IEC 1st CD 9798-6, Information technology—Security techniques—Entity authentication—Part 6: Mechanisms using manual data transfer—Protocol 3b, International Organization for Standardization, December 2003.
[17] C. Soriente, G. Tsudik, E. Uzun, BEDA: button-enabled device pairing, Cryptology ePrint Archive, Report 2007/246, 2007, Available at: http://eprint.iacr.org/2007/246.
[18] R. Mayrhofer, H. Gellersen, Shake well before use: authentication based on accelerometer data, in: Pervasive Computing, in: LNCS, vol. 4480, 2007, pp. 144–161.
[19] D. Bichler, G. Stromberg, M. Huemer, M. Löw, Key generation based on acceleration data of shaking processes, in: UbiComp 2007: Ubiquitous Computing, pp. 304–317.
[20] I. Buhan, J. Doumen, P. Hartel, R. Veldhuis, Feeling is believing: a secure template exchange protocol, in: Advances in Biometrics, in: LNCS, vol. 4642, 2007, pp. 897–906.
[21] D. Schurmann, S. Sigg, Secure communication based on ambient audio, IEEE Trans. Mob. Comput. 12 (2) (2011) 358–370.
[22] I. Buhan, J. Doumen, P. Hartel, R. Veldhuis, Secure ad-hoc pairing with biometrics: SAfE, in: Proceedings of 1st International Workshop on Security for Spontaneous Interaction, IWSSI '07, 2007, pp. 450–456.
[23] C. Soriente, G. Tsudik, E. Uzun, HAPADEP: human-assisted pure audio device pairing, in: Information Security Conference, ISC'07.
[24] M. Goodrich, M. Sirivianos, Loud and clear: human-verifiable authentication based on audio, in: Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on Distributed Computing Systems.
[25] J. McCune, A. Perrig, M. Reiter, Seeing-is-believing: using camera phones for human-verifiable authentication, in: IEEE Symposium on Security and Privacy 2005.
[26] D. Balfanz, D. Smetters, P. Stewart, H. Wong, Talking to strangers: authentication in ad-hoc wireless networks, in: NDSS 2002.
[27] U. Uludag, S. Pankanti, A.K. Jain, Fuzzy vault for fingerprints, in: Audio- and Video-Based Biometric Person Authentication, in: LNCS, vol. 3546, 2005, pp. 310–319.
[28] L. Ballard, S. Kamara, M.K. Reiter, The practical subtleties of biometric key generation, in: USENIX Security Symposium, 2008, pp. 61–74.
[29] T. Ignatenko, F. Willems, Information leakage in fuzzy commitment schemes, IEEE Trans. Inf. Forensics Secur. 5 (2) (2010) 337–348.
[30] Y. Dodis, L. Reyzin, A. Smith, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data, in: Advances in Cryptology—Eurocrypt 2004.
[31] B. Groza, R. Mayrhofer, SAPHE: simple accelerometer based wireless pairing with heuristic trees, in: Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia, pp. 3–5.
[32] N. Saxena, M. Uddin, Secure pairing of "interface-constrained" devices resistant against rushing user behavior, in: Applied Cryptography and Network Security, in: LNCS, vol. 5536, 2009, pp. 34–52.

[33] A. Gallego, N. Saxena, J. Voris, Exploring extrinsic motivation for better security: a usability study of scoring-enhanced device pairing, in: Financial Cryptography and Data Security, pp. 60–68.
[34] A. Perrig, D. Song, R. Canetti, The TESLA broadcast authentication protocol the TESLA broadcast authentication protocol, Tech. Rep., Carnegie Mellon University, Department of Engineering and Public Policy, 2005.
[35] R. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, R. Needham, A new family of authentication protocols, ACM SIGOPS Oper. Syst. Rev. 32 (4) (1998) 9–20.
[36] M. Sethi, M. Antikainen, T. Aura, Commitment-based device pairing with synchronized drawing, in: Proceedings of IEEE International Conference on Pervasive Computing and Communications, PerCom 2014.
[37] M. Maes, Polygonal shape recognition using string-matching techniques, Pattern Recognit. (5) (2002) 433–440. http://dx.doi.org/10.1016/0031-3203(91)90056-B.
[38] S. Kaygin, M. Bulut, Shape recognition using attributed string matching with polygon vertices as the primitives, Pattern Recognit. Lett. 23 (1–3) (2002) 287–294.
[39] L. Yujian, L. Bo, A normalized Levenshtein distance metric, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007) 1091–1095.