Commitment-based device pairing with synchronized drawing

Mohit Sethi*†, Markku Antikainen†, Tuomas Aura†
*NomadicLab, Ericsson Research, Finland
†Department of Computer Science, Aalto University, Finland
firstname.lastname@aalto.fi

***Abstract*—Secure device pairing is a widely studied problem. Local wireless connections such as Bluetooth and WiFi typically rely on user-entered secret keys or manually verified authentication codes. Several recent proposals replace these with contextual or location-dependent sensor inputs, which are assumed to be secret from anyone not present at the location where the pairing takes place. These protocols have to cope with a *fuzzy* secret, i.e. noisy secret input that differs between the devices. In this paper, we overview such protocols and propose a new variation using time-based opening of commitments. Our protocol has the advantage of treating the fuzzy secret as one piece of data rather than requiring it to be partitioned into time intervals, and being more robust against variations in input entropy than those based on error correction codes. The protocol development is motivated by the discovery of a novel human source for the fuzzy secret: synchronized drawing with two fingers of the same hand on two touch screens or surfaces. Metrics for measuring the distance between the drawings are described and evaluated. We implement a prototype of this surprisingly simple and natural pairing mechanism and show that it accurately differentiates between true positives and man-in-the-middle attackers.**

I. Introduction

Secure pairing of devices to enable easy creation of long-term associations as well as ad-hoc transactions is an important and widely studied problem [17]. 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 [3], WiFi [1], 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 ongoing search 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 use input such as sound or movement, for device pairing. 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 [20], [25], [29] or from simultaneous sensing of user actions. As an example, Mayrhofer [21] and well as Kirovski et al. [16] 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 noisy and, this, differing measurements of sensor data. 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.

We start in Section II by providing an overview and taxonomy of existing key-establishment methods that bootstrap the authentication from fuzzy shared secrets. In Sec. III, we derive a new time-based variant of the commitment-based key establishment. Sec. IV presents the novel pairing mechanism that uses synchronized drawings as the fuzzy shared secret. We discuss various metrics for comparing the drawings, implement a prototype and evaluate the security of this pairing mechanism. Sec. V discusses the security of the proposed scheme. Finally, Sec. VI concludes the paper.

II. Background

This section provides a survey and taxonomy of existing solutions for *authenticated key-establishment with fuzzy data*. We consider protocols that tolerate errors in the data from which they derive a shared secret key.

More precisely, two devices communicate over an insecure network, such as a wireless link, and need to establish a shared secret key. They 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.

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 [11]. 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.

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

978-1-4799-3445-4/14/$31.00 ©2014 IEEE

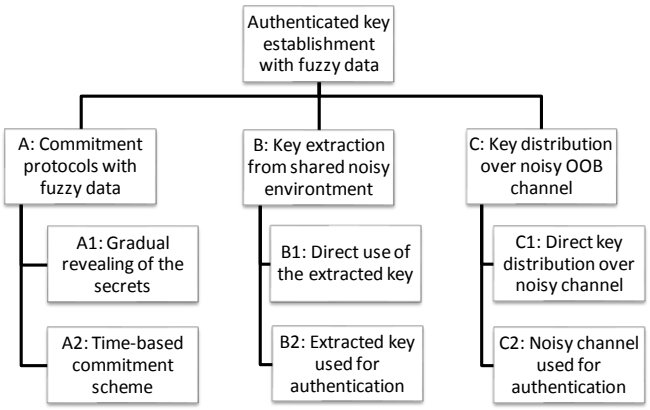


Figure 1: Taxonomy of key-establishment with fuzzy data

A. Commitment-based protocols

Commitment-based protocols (category A in Figure 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 [27]. 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 (but yet-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 an 6-digit number, distributed over an out of band channel. In *commitment phase* of the authentication, the devices exchange cryptographic commitments to the values of the short secret and the 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 *opening phase*, both sides reveal their random numbers r and verify that hash sent by the other party in the first phase is correct.

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 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, 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 [2], [18], [32]. In this protocol, the two devices A and B reveal the one-time secret $p = p_1|p_2|\ldots|p_n$ piecewise in n rounds and perform separate commitment and opening for each round. Figure 2 shows one protocol round. By spoofing

Device A	Message	Device B
$r_{A,i} \in_R \{0,1\}^*$		$r_{B,i} \in_R \{0,1\}^*$
$h_{A,i} :=$ $H(1, k, P_i, r_{A,i})$	$\xrightarrow{h_{A,i}}$ $\xleftarrow{h_{B,i}}$	$h_{B,i} :=$ $H(2, k, P_i, r_{B,i})$
	$\xrightarrow{r_{A,i}}$	Accept if $h_{A,i} =$ $H(1, k, P_i, r_{A,i})$
Accept if $h_{B,i} =$ $H(2, k, P_i, r_{B,i})$	$\xleftarrow{r_{B,i}}$	

Figure 2: ith round in MANA III protocol when the protocol is used to authenticate Diffie-Hellman key k

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). Varshavsky et al. [34] 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 very similar mechanism is proposed by Soriente et al. [30] where the user inputs the fuzzy data by simultaneously pressing buttons on the devices. Protocols in these proposals differ from the standard MANA III so 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 synchronization. Varshavsky et al. solved the correspondence issue by synchronizing the input intervals on the two devices. In our case, we record the fuzzy data without accurate synchronization and, in the most CPU-saving version of our scheme, without using timing data, which makes is difficult to partition the data into subintervals in the same way on both devices without first seeing the full inputs from both sides. To avoid the above limitations, we wanted to find a one-round scheme where the devices commit to, reveal and compare the fuzzy secrets all at once, without the need to split them into parts. Our time-based commitment scheme (A2) in Section III achieves this.

B. 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 ob-

Protocol	Soure of the shared secret	Protocol class
Amigo [34]	Radio environment	A1
BEDA (Button-to-button) [30]	Timing of button presses	A1
D-H and Interlock* [22]	Accelerometer data	A1
Generating key from accelerometer data [7]	Accelerometer data	B1
Candidate Key Protocol [21], [22]	Accelerometer data	B1
AdhocPairing [24], [25], [29]	Audio	B1
Secure Ad-hoc Pairing with Biometrics (SAfE) [8]	Biometric data	B1
Proximate [20]	Radio environment	B1
Feeling-is-Believing [9]	Biometric data (fingerprint)	B1
Secure communication based on ambient audio [28]	Ambient audio	B1
Human-Assisted Pure Audio Device Pairing (HAPADEP) [31]	Device generated secret sent over lossy audio-OOB channel	C1
BEDA (other variants) [30]	Device generated secret sent over lossy user controlled channel	C2
Loud-and-Clear [12]	Device generated secret sent over lossy audio-OOB channel	C2
Seeing-is-Believing [23]	Device generated secret sent over lossy visual-OOB channel	C2

Table I: Comparison of the proposed fuzzy key-establishment protocol

servations 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). The protocols described in the literature appear to be of this type. 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). This is 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 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 [8], [9], [33]. Several protocols using shared sensory input such as movement or ambient audio have also been based on the same idea [16], [24], [28], [29]. 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 inputs can be corrected.

The security of fuzzy cryptography depends on the error correction code not leaking any information about the secret fuzzy data [6], [10], [14]. 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 [6].

Other pairing protocols derive an encryption key directly from sensor data without using fuzzy cryptography. Here, we analyze two such protocols, namely CKP [22] and SAPHE [13]. 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 hashes are sent to the other device and compared. Only the feature vectors with identical hash values are used to create the encryption key. The protocol requires low environmental noise and accurately calibrated sensors to produce enough identical vectors between the devices. There is also no clear rule for how large fraction of the feature vectors can be ignored without endangering security. sensors. The authors also brush aside the question on how a large fraction of the feature vectors can be ignored due to unequal values, thereby increasing the time required to securely pair. Simple Accelerometer Based Wireless Pairing with Heuristic Trees (SAPHE [13]) similarly generates keys directly from the sensor inputs. Differences in the inputs are handled with a brute-force search for differing bits in the keys. The search is made efficient with 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.

C. Secure but noisy OOB channel

It may also be instructive to look at 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 a 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 [31]) offers a good illustration: an encryption key with error correction is sent over an audio channel, which the human user monitors. There are also many ways to use the noisy out-of-band channel for verifying the result of a key exchange. For example, Soriente et al. [30] 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 [12] and 2D bar codes in Seeing-is-believing [23]. Such protocols offer us examples of the available OOB channels but no ready solution as we ask the user to produce rather than compare the two fuzzy inputs.

III. 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 (Sec. IV). 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

Device A	Message	Device B
1. Diffie-Hellman key exchange		
	1a: $g^x \rightarrow$	
$k := g^{xy}$.	$\leftarrow$ **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: $c_A \rightarrow$	
Display "commitment received".	$\leftarrow$ **3b**: c_B	Display "commitment received".
4. Opening the commitments		
Wait for user approval to continue.		Wait for user approval to continue.
	4a: $E_k(A, v_A, r_A) \rightarrow$	
	$\leftarrow$ **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 $D(v_A, v_B) \leq d$.

(a) Basic protocol: explicit synchronization by user

Device A	Message	Device B
1. Diffie-Hellman key exchange		
	1a: $g^x \rightarrow$	
$k := g^{xy}$.	$\leftarrow$ **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: $c_A \rightarrow$	
Abort if message 3b received at time $> t_{A,0} + \Delta_1$ by A's local clock.	$\leftarrow$ **3b**: c_B	Abort if message 3a received at time $> t_{B,3} + \Delta_1$ 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: $E_k(A, v_A, r_A) \rightarrow$	
	$\leftarrow$ **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 $D(v_A, v_B) \leq$ d.

(b) Final protocol: implicit time-based synchronization

Figure 3: Commitment-based key establishment with fuzzy data

correction code in fuzzy cryptography (Sec. II-B) 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 the MANA III variant protocol (Sec. II-A) 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, the easier it is to deploy the protocol and the safer to deploy 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 it differs from those in the literature.

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
Δ_1	delay allowed for sending the commitment and receiving the other device's commitment
Δ_2	safety period between the last time to receive commitments and the time to open one's own

Table II: Summary of variables and functions at device A

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 fresh random number, k is the shared secret key produced in step 1, and H is a cryptographic hash function. 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 input. The commitments are verified by recomputing the other side's hash value and 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 Sec. IV 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, both devices should reveal their values v_A and r_A 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 (Figure 3a), 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 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.

There are ways of reducing the risk of user carelessness. First, the user interface could be designed to encourage (but 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. 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.

After these preliminaries, the time-based commitment scheme becomes easy to describe. The idea is inspired by the TESLA [26] and Guy Fawkes [5] broadcast authentication protocols, in which the sender opens a kind of commitment at a fixed time. However, the previous schemes use the time-based opening of the commitments directly for message authentication, while we use it for key establishment and in combination with fuzzy secrets.

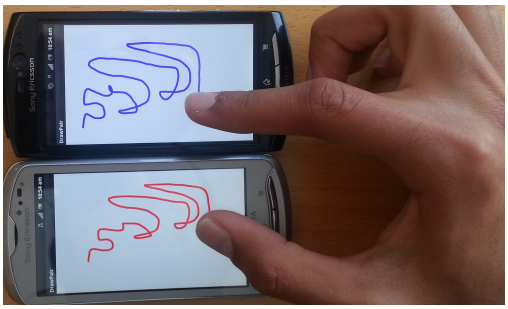


Figure 4: Synchronized drawing for device pairing

The final protocol (Figure 3b) 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 Sec. V). 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. It opens the commitments at the time $t_{A,0} + \Delta_1 + \Delta_2$, where Δ_2 allows for a safety period between the commitments and their opening. Naturally, device B performs the equivalent operations using its own local clock.

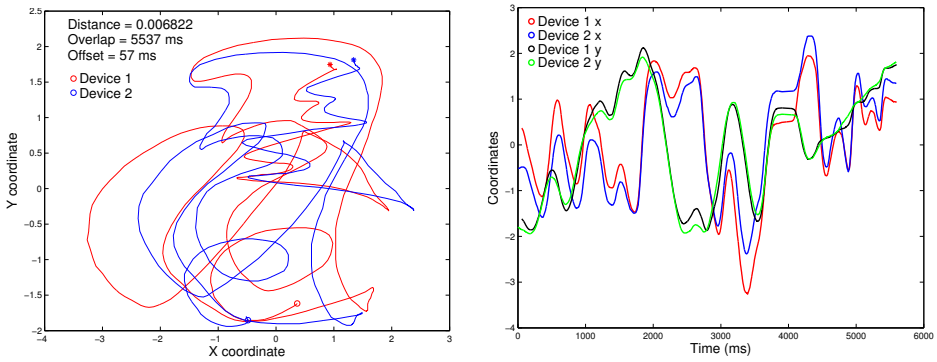
The above protocol depends critically on the timing, and the time constants need to be fixed in the specification. We have set the time periods Δ_1 and Δ_2 to 200 ms, which is sufficient for message transfer over any modern wireless or wired network if the network connection has been established in advance. We set Δ_3 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.

IV. Synchronized Drawing

This section describes a novel type of shared noisy input for devices with a touch screen on 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. 4. The picture also shows sample drawings on two devices. This shared input is used as the fuzzy shared secret data for 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 a smart phone or another device to a laptop computer.

A. Location metric

In order to compare the drawings, we need to define a *distance metric* and a *threshold value* for accepting the



(a) Centered and scaled drawings

(b) x and y coordinates as function of time

Figure 5: Location metric for comparing the drawings

drawings as the same. The same metric and threshold should work for all devices and users, so that the authentication can be used without a training period or setting parameters. Our experiments showed that such metrics and threshold can be found easily and that the comparison is not sensitive to minor changes in the parameters of the algorithms.

The most obvious metric is based on *location as a function of time*. Fig. 5a 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, but that proved unnecessary in practice, except for $90°$, $180°$, and $270°$ turns (touch-screen devices are typically rectangular and are placed next to each other with random sides touching). Fig. 5b 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, which is very close to the time shift that also gives the highest cross-correlation. In this case, the offset between the drawings is 57 ms from the start of the drawings, which is quite a typical value. Fig. 5a also indicates the start and end of the overlapping time period, over 5 seconds in this case. The location-based distance metric is computed from the centered, scaled and interpolated data as the Euclidian norm of the difference between the x and y coordinate vectors:

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

Above, the norm is 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 seconds) and a maximum difference for the lengths of the drawings (500 ms).

As can be seen in Fig. 6, the location metric clearly separates pairs of drawings that have been drawn synchronously with two fingers from pairs of drawings that have no such relation. A threshold value can be easily chosen (e.g. 0.017) so that there are no false positives and only few false negatives for the authentication. Moreover, the latter were caused by obvious user mistakes such as moving the finger for prolonged periods outside the touch-sensitive area of the surface.

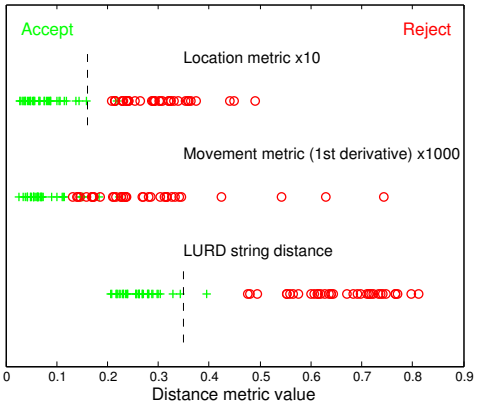


Figure 6: Comparison of distance metrics for drawings

B. 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 the first and second derivatives of the location, i.e. velocity and acceleration, but were not able to derive useful metrics from these. The main reason is that the event-based API 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 speed or acceleration of the finger. 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.

C. LURD string distance

While the location-based metric is otherwise excellent 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 reasons, we wanted a distance metric for the drawings that takes little computing power, does not require accurate time synchronization or computing

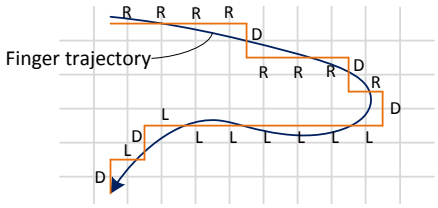


Figure 7: LURD-string encoding

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 [15], [19]. These algorithms encode the drawings into a string, which can be then compared using approximate string comparison algorithms such as the Levenshtein distance.

Our string-based metric is called the *LURD string 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 32×32 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" (see Fig. 7). In the very rare cases where the crossing happens exactly at (or within the rounding error from) the junction of two grid lines, the crossing of the vertical line is recorded first. The substrings from the recorded movement events are concatenated to represent each drawing. The two drawings are then compared using a string distance metric that gives weight 1 to each deletion or insertion (and 2 to substitution) of a character that is required to convert one of the strings to the other.

As can be seen in Fig. 6, 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. There are simple and efficient algorithms for computing such string distances [35], and these proved to work well on the mobile devices.

V. 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. While the protocol was motivated by the particular type of fuzzy secret, it can be used with other types of noisy user or environmental input. We verified the security of the key establishment with ProVerif to avoid any crude mistakes (see Appendix A). 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. Another limitation of our time-based scheme is that the user input should provide a reference event that can be used to reliably synchronize the commitment and opening phases.

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. 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 for this synchronization is equal to the time constant Δ_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 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.

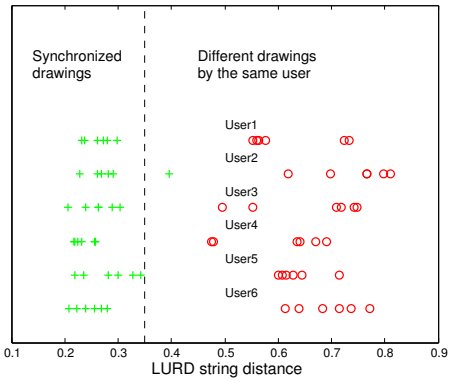
Another requirement for security is naturally that the fuzzy secret must have sufficient entropy. As mentioned earlier, it is difficult to measure the entropy of the drawings. Fortunately, we are using the fuzzy secret for authenticating the key exchange, which means that u bits of uncertainty reduces an active attacker's guessing probability to 2^{-u}. Uncertainty of only $u = 20 \ldots 30$ bits is typically sufficient because 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 seconds, 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/second for a drawing with extended length, encoded as a LURD string, but such results should be treated sceptically in security-critical 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 in real time the drawing, 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 most types of user input, environmental input, and

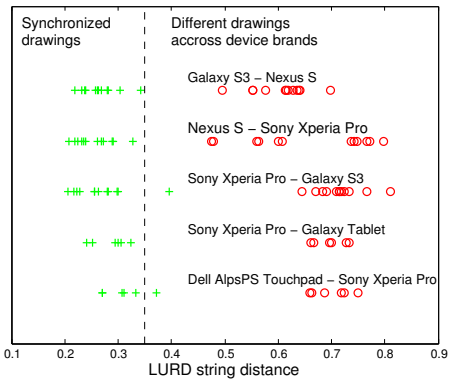
location-limited channels. Fortunately, the security implications of spying are rather intuitive for the user to understand. To make the spying of the drawings slightly more difficult, we implemented the drawing display as a fading trail (disabled in Fig. 4 for illustrative purposes).

Experiments (e.g. Fig. 6) 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. 8a) and different devices (Fig. 8b). 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 two wiggles for each pair of devices being experimented. 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 OS 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.

Finally, the fuzzy data sent in the protocol 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. The LURD metric has parameters, such as the number of directions and the size of the grid, which have not yet been full explored. For our current purposes, the simple string metric presented in this paper seems to work well.



(a) Variation between users



(b) Drawings on different devices

Figure 8: Variation of the LURD distance metric

VI. Conclusion

The goal of this paper is to develop a secure and natural mechanism for pairing devices. We focus on devices with a touch screen or other touch-sensitive surface. 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 is this paper is the end of user input.

Second, in addition to introducing the idea of synchronized drawing, we discuss potential metrics for comparing the drawings and show that both location metrics and an easier-to-compute LURD string distance measure are suitable for the task. They produced no false positives and few false negatives in the authentication experiments. A prototype of the authentication was implemented on Android smart phones, a tablet device, and a laptop 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. While the security of the scheme has been analyzed in this paper, more data could still be collected to maximize the error margins and to evaluate the usability of the scheme outside the laboratory.

VII. Acknowledgements

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] IEEE Std. 802.11 WG, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,.

[2] ISO/IEC 1st CD 9798-6, Information technology – Security techniques – Entity authentication – Part 6: Mechanisms using manual data transfer – Protocol 3b, December 2003. International Organization for Standardization.

[3] 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). 2005.

[4] ZigBee specification. pages 344–346, 2006.

[5] R. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. Needham. A new family of authentication protocols. *ACM SIGOPS Operating Systems Review*, 32(4):9–20, 1998.

[6] L. Ballard and M. K. Reiter. The Practical Subtleties of Biometric Key Generation. pages 61–74.

[7] D. Bichler, G. Stromberg, M. Huemer, and M. Löw. Key generation based on acceleration data of shaking processes. *UbiComp 2007: Ubiquitous Computing*, pages 304–317, 2007.

[8] I. Buhan, J. Doumen, P. Hartel, and R. Veldhuis. Secure Ad-hoc Pairing with Biometrics : SAfE.

[9] I. Buhan, J. Doumen, P. Hartel, and R. Veldhuis. Feeling is believing: a secure template exchange protocol. *Advances in Biometrics*, 2007.

[10] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *Advances in cryptology-Eurocrypt 2004*, 2004.

[11] D. Dolev and a. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, Mar. 1983.

[12] M. Goodrich and M. Sirivianos. Loud and clear: Human-verifiable authentication based on audio. *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on Distributed Computing Systems*, 2006.

[13] B. Groza and R. Mayrhofer. SAPHE: simple accelerometer based wireless pairing with heuristic trees. *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, pages 3–5, 2012.

[14] T. Ignatenko and F. M. J. Willems. Information Leakage in Fuzzy Commitment Schemes. *IEEE Transactions on Information Forensics and Security*, 5(2):337–348, June 2010.

[15] S. Kaygin and M. Bulut. Shape recognition using attributed string matching with polygon vertices as the primitives. *Pattern Recognition Letters*, 23(1-3):287–294, Jan. 2002.

[16] D. Kirovski, M. Sinclair, and D. Wilson. The martini synch. Technical report, Technical report, Microsoft Research Technical Report, MSR-TR-2007-123, 2007.

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

[18] J.-O. Larsson. Higher layer key exchange techniques for bluetooth security. In *Open Group Conference, Amsterdam*, 2001.

[19] M. Maes. Polygonal shape recognition using string-matching techniques. *Pattern Recognition*, 24(5):433–440, Jan. 1991.

[20] S. Mathur, R. Miller, and A. Varshavsky. Proximate: proximity-based secure pairing using ambient wireless signals. *Proceedings of the 9th international conference on Mobile systems, applications, and services*, 2011.

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

[22] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. *Pervasive computing*, pages 144–161, 2007.

[23] J. McCune, A. Perrig, and M. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. *Proceedings. 2005 IEEE Symposium on Security and Privacy*, (May), 2005.

[24] N. Nguyen, S. Sigg, A. Huynh, and Y. Ji. Pattern-Based Alignment of Audio Data for Ad Hoc Secure Device Pairing. *2012 16th International Symposium on Wearable Computers*, pages 88–91, June 2012.

[25] N. Nguyen, S. Sigg, A. Huynh, and Y. Ji. Using ambient audio in secure mobile phone communication. *Pervasive Computing and Communications Workshops (PERCOM Workshops)*, (March):431–434, 2012.

[26] A. Perrig, D. Song, and R. Canetti. The TESLA Broadcast Authentication Protocol The TESLA Broadcast Authentication Protocol. Technical report, Carnegie Mellon University, Department of Engineering and Public Policy, 2005.

[27] R. L. Rivest and A. D. I. Shamir. How to Expose an Eavesdropper. *Communications of the ACM*, 27(4):393–395, 1984.

[28] D. Schurmann and S. Sigg. Secure communication based on ambient audio. 12(2):358–370, 2011.

[29] S. Sigg and Y. Ji. AdhocPairing : Spontaneous audio based secure device pairing for Android mobile devices. 2012.

[30] C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button-enabled device association. 2007.

[31] C. Soriente, G. Tsudik, and E. Uzun. HAPADEP: human-assisted pure audio device pairing. *Information Security*, 2008.

[32] J. Suomalainen, J. Valkonen, and N. Asokan. Standards for security associations in personal networks: a comparative analysis. *International Journal of Security and Networks*, 4(1):87–100, 2009.

[33] U. Uludag, S. Pankanti, and A. K. Jain. Fuzzy vault for fingerprints. In *Audio-and Video-Based Biometric Person Authentication*, pages 310–319. Springer, 2005.

[34] A. Varshavsky and A. Scannell. Amigo: Proximity-based authentication of mobile devices. *UbiComp 2007: Ubiquitous Computing*, pages 253–270, 2007.

[35] L. Yujian and L. Bo. A normalized levenshtein distance metric. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1091–1095, 2007.

APPENDIX A
PROTOCOL VERIFICATION WITH PROVERIF

We verified the security of our commitment protocol with the PROVERIF model-checking tool. The comments `P1`,`P3`,`P4` indicate the corresponding states used in Sec. III. Step 2 is omitted from the model because the processes receive the fuzzy secret as an input parameter. The approximate comparison is modeled as equality. The exact timing is not included in this model, but the statement `phase 1;` models the movement from the commitment phase to opening phase.

```
(* Channels *)
channel c.
free oob:channel [private].
(* Types and constants *)
type G. type exponent. type key. type tag.
const DEVICEA, DEVICEB: tag [data].
(* Diffie-Hellman *)
const g: G [data].
fun exp(G, exponent): G.
equation forall x: exponent, y: exponent;
    exp(exp(g, x), y) = exp(exp(g, y), x).
(* Hash function *)
fun H(bitstring):key.
(* Shared key encryption *)
fun enc(key, bitstring): bitstring.
reduc forall x: bitstring, y: key; dec(y, enc(y,x)) = x.
(* Secrecy assumptions *)
free secretMsg: bitstring [private].
free drawing: bitstring [private].
query attacker(secretMsg).

let pairing(toId:tag, fromId:tag, drawing:bitstring) =
    new dhsecretA: exponent;                        (* P1 *)
    let dhpublicA = exp(g, dhsecretA) in
    out(c, (toId, dhpublicA));
    in(c, (=fromId, dhpublicB: G, sign: bitstring));
    let K = exp(dhpublicB, dhsecretA) in

    new rA: bitstring;                              (* P3 *)
    out(c, H((toId, drawing, rA, K)));
    in(c, cB: key);

    phase 1;
    new synchronizationMsg1: bitstring;
    out(oob, synchronizationMsg1);
    in(oob, synchronizationMsg2: bitstring);

    out(c, (drawing, rA));                           (* P4 *)
    in(c, (vB: bitstring, rB: bitstring));

    if H((fromId, vB, rB, K)) = cB && vB = drawing then(
        out(c, enc(H((toId, K)), secretMsg));0
    ).

process
    ( (!pairing(DEVICEA, DEVICEB, drawing)) |
      (!pairing(DEVICEB, DEVICEA, drawing)) )
```