The Candidate Key Protocol for Generating Secret Shared Keys From Similar Sensor Data Streams

Rene Mayrhofer

Lancaster University, Computing Department, South Drive, Lancaster LA1 4WA, UK
`rene@comp.lancs.ac.uk`,
WWW home page: `http://www.comp.lancs.ac.uk/`

Abstract. Secure communication over wireless channels necessitates authentication of communication partners to prevent man-in-the-middle attacks. For spontaneous interaction between independent, mobile devices, no a priori information is available for authentication purposes. However, traditional approaches based on manual password input or verification of key fingerprints do not scale to tens to hundreds of interactions a day, as envisioned by future ubiquitous computing environments. One possibility to solve this problem is authentication based on similar sensor data: when two (or multiple) devices are in the same situation, and thus experience the same sensor readings, this constitutes shared, (weakly) secret information. This paper introduces the *Candidate Key Protocol* (CKP) to interactively generate secret shared keys from similar sensor data streams. It is suitable for two-party and multi-party authentication, and supports opportunistic authentication.

1   Introduction

Secure communication over a wireless channel is a difficult problem, especially for spontaneous interaction. Spontaneous interaction in the sense of ad-hoc communication between devices is often aimed for in ubiquitous computing [1], following its vision of seamlessly interacting with whatever services are currently available and useful. Moreover, many of these proposed devices are small, need to cope with limited resources such as memory, computational power and battery life, and do not have any conventional user interfaces such as key pads or displays. Communication is assumed to happen over shared wireless channels that are open to any device, which is necessary to enable transparent interoperability.

It is difficult to secure such interactions because we can not assume the involved devices to have any a priori information about each other. Creating a secure channel depends on an authentication step. If Alice (A) wants to interact with Bob (B)[1] and does not know anything about Bob a priori, then she will be unable to distinguish a legitimate interaction with Bob from malicious behavior

---

[1] In the context of this paper, we use A, B, and E for describing the devices that interact with each other interchangeably with the established names Alice, Bob,

by Eve (E) — Eve can simply perform a valid protocol run with Alice. Currently, there is no globally trusted public key infrastructure (PKI), and it is doubtful if there will be any. Even if there was one that would be able to sign trusted devices, it would not solve the problem of authenticating spontaneous interaction: Eve could just set up a trusted device E of her own and intercept the communication by getting A to communicate with her device instead of B. We therefore need to individually authenticate the interaction between each communicating pair of devices. Such authentication essentially aims at secret key agreement between A and B.

This problem is amplified as ubiquitous computing is expected to generate far more frequent spontaneous interactions. When using hundreds of different devices each day, conventional authentication methods like passwords or PINs fail to scale. Examples of devices that communicate wirelessly with each other are mobile phones, Bluetooth headsets, networked cameras, printers, in the near future goggles with integrated displays, and many more. We use the practical example of establishing a secure channel between a mobile phone and a Bluetooth headset without loss of generality.

Our approach is to authenticate devices based on shared context, which is manifested by similar sensor readings. Whenever two devices are in the same situation, e.g. being worn by the same person, capturing the same audio environment, or just being close to the same object, their sensors will experience similar time series. These time series can be used to implicitly authenticate a secure channel between the devices. There are multiple possibilities for authentication based on similar time series. The more conventional approach is to perform an unauthenticated (anonymous) key agreement like Diffie-Hellman [2], exchange the time series using the secret shared key via some commitment scheme, and compare if they are similar enough with an appropriate metric to prevent man-in-the-middle (MITM) attacks. However, this approach is computationally expensive and consists of two phases, which introduces an additional delay. We present an authentication protocol, the *Candidate Key Protocol* (*CKP*), which derives cryptographic key material directly from sensor data streams and utilizes only hash functions as cryptographic primitives.

In Section 2, we discuss related work and motivate the need for an authentication protocol based on conventional primitives in spite of more recent research on information theoretic security. After defining the threat scenarios that CKP is designed to deal with in section 3, we explain the approach and detailed specification of CKP in section 4. A first practical implementation using UDP multicast and initial experimental results are described in sections 5 and 6, respectively. We finish with discussing the security properties and possibilities for extending the protocol in section 7.

---

and Eve of the respective users. The reason is that one of the devices might be an infrastructure device, such as a printer or a display, that does not belong to any single user.

2   Related Work

Results from two research areas are relevant to the present paper: information theoretical work in cryptography with influences from quantum cryptography, and authentication protocols inspired by practical issues, mostly from ubiquitous computing research.

Generating keys from noisy channels, or more generally, from (random) correlated information, received some attention in theoretical cryptography research, e.g. [3][4][5][6]. For a good introduction into the topic and for results for public, non-authenticated channels, we refer to [7,8,9]. These publications give interesting information theoretical results on key agreement, which no longer assume the intractability of some computational problem like the discrete logarithm problem, but provide what is often called "unconditional security". The basic concept is that, when two legitimate communication partners either have a noisy communication channel or when they have access to correlated information, then it is possible for them to agree to a secret key even when an adversary has access to their noisy channel or partial knowledge of their shared information. There are two classes of such authentication protocols: interactive, e.g. [7,8,9], and non-interactive, e.g. [6]. Non-interactive protocols have the obvious advantage that they can be used to establish a shared secret when only one-way communication is available. This has additional practical consequences. Even when two-way communication is possible, issues like time delays, packet loss, etc. can be handled more easily with non-interactive protocols. On the other hand, interactive protocols are necessary under the assumption of active adversaries (see e.g. [7, section III.D]). Our proposed protocol is interactive.

Other results [10] seem particularly promising because they describe an authentication protocol based on a weak secret key, which closely matches our real world problem of using sensor time series as a weak secret key.

However, these theoretical results do not yet seem to have been implemented, and practical applicability is therefore still limited. Another problem is that, although the shared secrets may be weak, large secrets are required to guarantee the security properties of these protocols. For small and embedded devices, it is difficult to process large strings of secret data, and it is difficult to find good sources of large secret strings in the first place.

In contrast, we use conventional, i.e. computational, cryptographic primitives based on intractability assumptions which are still assumed to hold. With possible future availability of quantum computers, these assumption may need to be revised. In this paper, we use the terminology of information theoretical cryptography as far as appropriate because of the similar aims and assumptions. When adding the assumption of non-reversibility of cryptographic hash functions, then our proposed Candidate Key Protocol can be seen as an instance of a secret key agreement based on correlated random variables.

It is not obvious how the calculus introduced in [8] for noisy channels could be applied to the case of similar sensor time series that A and B have access

to and which E can get some knowledge about. Future work may use this or a similar calculus to analyze the security of CKP more analytically.

A large number of interactive protocols based on authenticated Diffie-Hellman (*DH*) key exchange [11] have recently been suggested, mostly inspired by practical problems of authentication in real world applications. This is assumed to be computationally, instead of unconditionally secure. The classical interlock protocol [12] can be seen as a predecessor of these, but it already used the notion of committing to values before revealing them. Newer protocols are mostly based on commitment schemes, e.g. the MANA family of protocols for manual string input or verification [13], optimized in [14].

While the "resurrecting duckling protocol" [15] aims at long-lived pairings, Hoepman introduced pairing protocols for short-lived interactions based on manual exchange of secrets [16][17], which scales poorly from a user point of view. The protocol proposed in [16] is very similar to MANA III [13] and seems to have been developed independently. Vaudenay claims [18] that Hoepman's protocol can not be implemented securely due to the lack of known hash functions with properties required by the protocol, and presents a protocol called SAS, which provides the same level of security with shorter shared secrets.

Creese et al. introduce a formal model for verifying authentication protocols that work with empirical verification [19]. They present the analysis of three related pairing protocols and show proofs of their security under their model.

Čagalj et al. describe three other pairing protocols with similar aims, based on short string comparison, distance bounding, and integrity codes [20]. Their second protocol is based on distance measurement, but we suggest that their scheme might be applicable to an interactive challenge-response scheme based on sensor data.

CKP is related to all these protocols because it shares similar aims, but differs in the approach. Instead of authenticating ephemeral session keys or long-term pairings created with DH, CKP creates shared keys by using sensor streams as input.

3 Threat Scenarios

In this section, we briefly outline the threat scenarios that are relevant to a device authentication protocol and to CKP in particular. Typical threats for a communication channel are *eavesdropping*, *replaying* of messages, and *deletion*, *insertion* and *modification* of messages. All of these threats are subsumed in the so-called man-in-the-middle (*MITM*) attack, where E is assumed to be "in between" A and B and have complete control over their communication channel. When an unauthenticated key agreement like Diffie-Hellman is used between A and B, E can delete all messages between A and B and instead perform two independent key agreements, one with A and one with B. In this paper, we explicitly assume an active adversary, and CKP is designed to detect when a MITM attack is being performed and fail to authenticate in this case. However, in the general case, it is not possible to distinguish between a *benign* authentication

failure when the sensor values experienced by A and B are not similar enough and a *malign* authentication failure caused by an attack.

Another typical threat is *denial of service* (DoS). This refers to E making communication, and in the scope of this paper, authentication impossible between A and B. When assuming an active adversary, DoS is easily possible and will therefore not be discussed further. However, the protocol should provide indication to the user when it can not complete, either due to benign communication error or due to a DoS attack. Distinguishing between these two cases is, again, not possible in the general case and we therefore treat them equally.

We also point out that attacks on the involved devices themselves are out of the scope of this paper and assume that the two devices A and B are trusted for the purpose of the interaction. If A trusts B with some document, but B (intentionally or due to an attack) forwards it to E, then authentication between A and B can not prevent this.

To summarize, our main threat scenario is an active attack on the (wireless) channel including full MITM capabilities. We assume that there is some sensor data which both A and B can get with better accuracy than E. Here we use the same argument as applied in [7, Theorem 5]: if Alice and Bob do not share any correlated information, then "from Bob's point of view, Alice has no advantage compared to Eve. If Eve performs the same protocol as Alice would, pretending to be Alice, Bob accepts with the same probability as he would accept a protocol execution with Alice". Assuming an experiment where Alice, Bob, and Eve can receive the same bit string over independent noisy channels, [7] concludes that "secret-key agreement against *active* adversaries is only possible if Alice's and Bob's channels are both less noisy than Eve's channel". This is to be intuitively expected, but in contrast to the results for passive adversaries [3].

We argue that this assumption is justified because, when A and B are in a similar context, their sensor time series should be more similar to each other than to the sensor time series perceived by E, even if only slightly. This can be achieved by measuring *local* physical phenomena which an adversary can not reasonably influence to obtain measurements with higher accuracy than A and B. Examples for appropriate phenomena are acceleration, sound, light, or radio frequency signal strength.

4   The Candidate Key Protocol

The candidate key protocol interactively generates secret shared keys from sensor streams between two (or multiple) devices. Figure 1 shows the relations between A, B and E. All devices are assumed to have full access to a wireless communication channel, and we explicitly assume E to be capable of deleting, inserting, and modifying messages between A and B without them being able to notice at this level. Additionally, A and B are assumed to share aspects of their context and have sensors that can capture these aspects. E is assumed not to share the same context, but be able to access it with (similar or different) sensors with inferior accuracy.
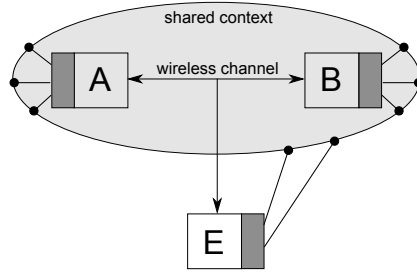
Fig. 1. Assumptions of CKP: A, B, and E share complete access to a wireless channel, and restricted access to the context in which interaction is taking place

4.1 Approach

Our approach to generating secret shared keys from similar, but not equal, sensor time series is based on the concept of *candidates*. When sampling sensor time series, raw samples are typically not used directly, but more meaningful *features* are extracted based on domain specific knowledge. We note that this step is critical for any use of sensor data, although the respective requirements depend on the application. For authentication, i.e. generating cryptographic key material, it is important for the extracted features to have high entropy from an adversary's point of view. In the terms typically used for feature extraction and context-aware systems, high entropy implies that the chosen features must clearly distinguish devices being in the same context from devices being in different contexts. The reverse, however, does not necessarily hold. Good separation in this sense does not imply high entropy, because an adversary is free to choose different features. Therefore, features should be chosen such that an adversary can learn the least amount of information about their specific values. We can not give generally valid recommendations because features are highly application specific. For the scope of this paper, we assume feature vectors to be available as input for authentication.

Even with features that perform well for a given application, there is still room for errors. To generate key material from feature vectors, we need to convert to integer values at some point; simple quantization errors can then lead to different keys even when the feature vectors are very similar. That is, quantization can increase noise. Generally, extracting and comparing feature vectors is a classification problem with the usual trade-off between separation and recall. Making features more distinctive to generate higher entropy will generate more errors in comparing feature vectors from devices in the same context, i.e. *false negatives*. Tuning the features for more robustness will make it easier for an adversary to estimate their values, i.e. generate *false positives*. For the purpose of authentication, false positives must be strictly avoided, but when the false negatives rate is too high, the authentication method may become unusable in practice. Therefore, our approach is to allow the feature extraction step to yield multiple (parallel) feature vectors in each time step. We then use these multiple

different candidates in a way that does not leak additional information to an adversary, and thereby provide a partial solution to this trade-off.
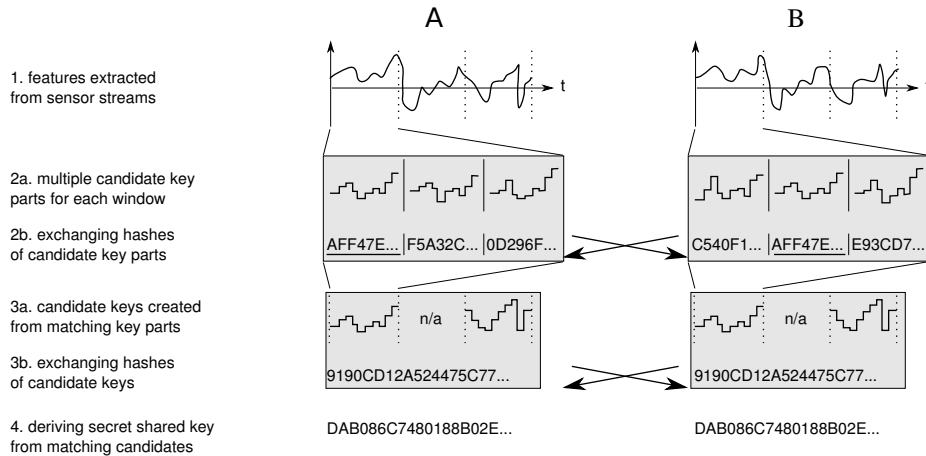

Fig. 2. Approach to generating a secret shared key: candidate key parts are time windows over extracted features from sensor time series and are concatenated to candidate keys

In Fig. 2 we show an overview of CKP, starting with the extracted features. The generation of multiple candidates for each feature vector is again application specific, but there exist general methods. One example is that different offsets for quantization can be used to alleviate the problem of quantization errors and thus solve a large class of false negatives. This method is depicted in Fig. 2. Every feature vector becomes a *candidate key part.* That is, it is a candidate for inclusion in the shared secret, subject to matching with the remote device. We then compute hashes of all candidate key parts for the current time step, which we abstract to a strictly monotonically increasing round number. These hashes are exchanged between A and B to verify which of the candidate key parts, if any, match. Note that transmitting their cryptographic hash values does not reveal any useful information about the candidate key parts themselves, because secure hash functions are assumed to be one-way functions. After a sufficient number of matching key parts, i.e. after accumulating enough entropy, the matching key parts are concatenated to a *candidate key.* With the possibility of multiple matches in each round, there are different ways to concatenate this key. Therefore, hashes of the candidate keys are again exchanged between A and B. When they match, A and B have successfully agreed to a shared secret.
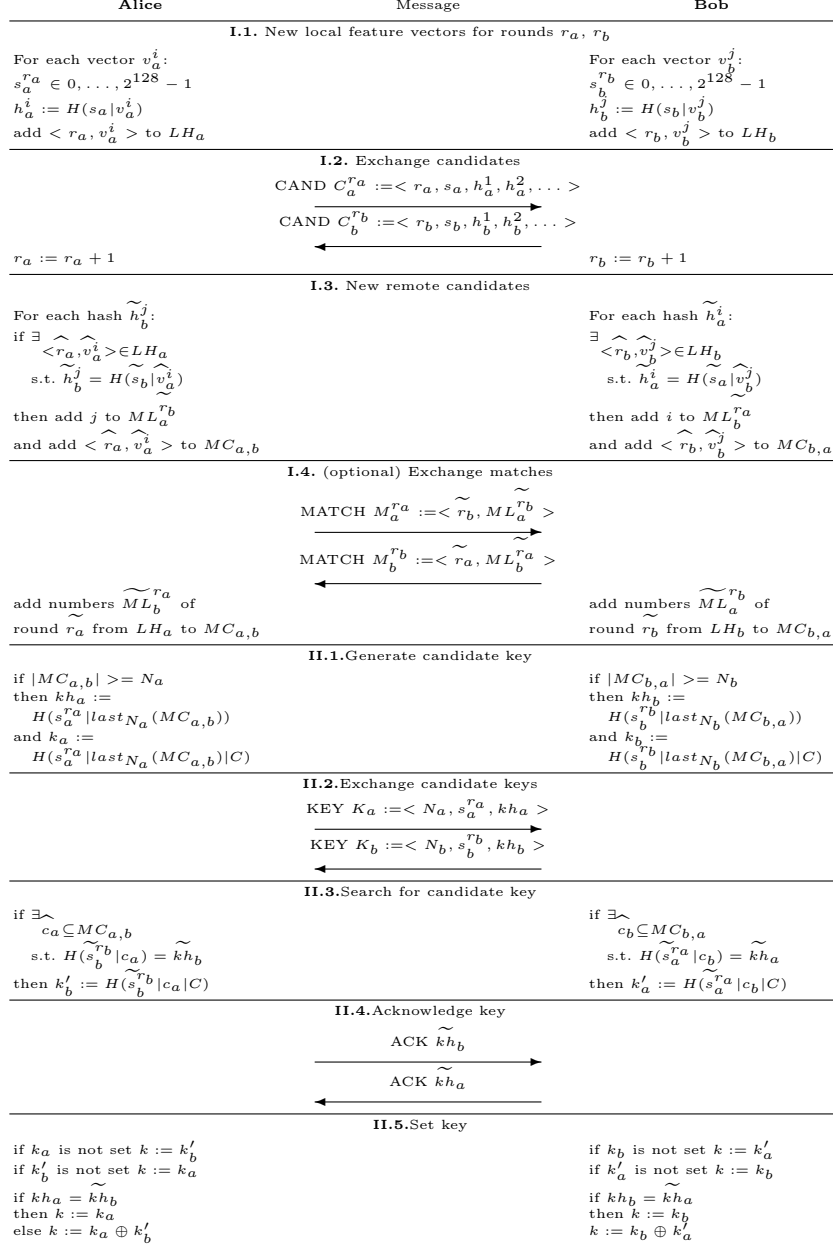
Alice	Message	Bob
I.1. New local feature vectors for rounds r_a, r_b		
For each vector v_a^i: $s_a^{r_a} \in 0, \ldots, 2^{128} - 1$ $h_a^i := H(s_a|v_a^i)$ add $< r_a, v_a^i >$ to LH_a		For each vector v_b^j: $s_b^{r_b} \in 0, \ldots, 2^{128} - 1$ $h_b^j := H(s_b|v_b^j)$ add $< r_b, v_b^j >$ to LH_b
I.2. Exchange candidates		
	CAND $C_a^{r_a} := < r_a, s_a, h_a^1, h_a^2, \ldots >$ $\longrightarrow$	
	CAND $C_b^{r_b} := < r_b, s_b, h_b^1, h_b^2, \ldots >$ $\longleftarrow$	
$r_a := r_a + 1$		$r_b := r_b + 1$
I.3. New remote candidates		
For each hash $\widetilde{h_b^j}$: if $\exists\ \widehat{<r_a, v_a^i>} \in LH_a$ s.t. $\widetilde{h_b^j} = H(\widetilde{s_b}|\widehat{v_a^i})$ then add j to $ML_a^{r_b}$ and add $< \widehat{r_a}, \widehat{v_a^i} >$ to $MC_{a,b}$		$\exists\ \widehat{<r_b, v_b^j>} \in LH_b$ s.t. $\widetilde{h_a^i} = H(\widetilde{s_a}|\widehat{v_b^j})$ then add i to $ML_b^{r_a}$ and add $< \widehat{r_b}, \widehat{v_b^j} >$ to $MC_{b,a}$
I.4. (optional) Exchange matches		
	MATCH $M_a^{r_a} := < \widetilde{r_b}, ML_a^{\widetilde{r_b}} >$ $\longrightarrow$	
	MATCH $M_b^{r_b} := < \widetilde{r_a}, ML_b^{\widetilde{r_a}} >$ $\longleftarrow$	
add numbers $\widetilde{ML_b^{r_a}}$ of round $\widetilde{r_a}$ from LH_a to $MC_{a,b}$		add numbers $\widetilde{ML_a^{r_b}}$ of round $\widetilde{r_b}$ from LH_b to $MC_{b,a}$
II.1. Generate candidate key		
if $|MC_{a,b}| >= N_a$ then $kh_a :=$ $H(s_a^{r_a}|last_{N_a}(MC_{a,b}))$ and $k_a :=$ $H(s_a^{r_a}|last_{N_a}(MC_{a,b})|C)$		if $|MC_{b,a}| >= N_b$ then $kh_b :=$ $H(s_b^{r_b}|last_{N_b}(MC_{b,a}))$ and $k_b :=$ $H(s_b^{r_b}|last_{N_b}(MC_{b,a})|C)$
II.2. Exchange candidate keys		
	KEY $K_a := < N_a, s_a^{r_a}, kh_a >$ $\longrightarrow$	
	KEY $K_b := < N_b, s_b^{r_b}, kh_b >$ $\longleftarrow$	
II.3. Search for candidate key		
if $\exists\ \widehat{c_a} \subseteq MC_{a,b}$ s.t. $H(\widetilde{s_b^{r_b}}|c_a) = \widetilde{kh_b}$ then $k_b' := H(\widetilde{s_b^{r_b}}|c_a|C)$		if $\exists\ \widehat{c_b} \subseteq MC_{b,a}$ s.t. $H(\widetilde{s_a^{r_a}}|c_b) = \widetilde{kh_a}$ then $k_a' := H(\widetilde{s_a^{r_a}}|c_b|C)$
II.4. Acknowledge key		
	ACK $\widetilde{kh_b}$ $\longrightarrow$	
	ACK $\widetilde{kh_a}$ $\longleftarrow$	
II.5. Set key		
if k_a is not set $k := k_b'$ if k_b' is not set $k := k_a$ if $kh_a = \widetilde{kh_b}$ then $k := k_a$ else $k := k_a \oplus k_b'$		if k_b is not set $k := k_a'$ if k_a' is not set $k := k_b$ if $kh_b = \widetilde{kh_a}$ then $k := k_b$ $k := k_b \oplus k_a'$

Fig. 3. Specification of CKP

4.2 Specification

After introducing the concepts of candidate key parts and candidate keys, we now present the detailed specification of CKP. Figure 3 defines the steps of the protocol.

For the formal description, we use the following notation: $H(m)$ describes the hashing of message m with some secure hash algorithm, and $m|n$ describes the concatenation of strings m and n. The symbol $\oplus$ describes bit-wise XOR and $|S|$ the number of elements in a set S. When a message M is transmitted over an insecure channel, we denote the received message $\widetilde{M}$ to point out that it may have been modified in transit, by noise or attack. Subscripts denote the different sides (a or b for an authentication between A and B), while superscripts denote specific vectors in a set of vectors. The syntax $\widehat{x}$ denotes the (open) result of a search for matches in a set. When a hash is computed from a set of vectors, we mean the concatenation of all vectors in some pre-defined order, typically by their round number.

v denotes raw feature vectors without cryptographic key properties, i.e. they do not need to be distributed uniformly. h denotes cryptographic hashes of feature vectors and r denotes round numbers. Each host keeps a set LH as a history of recently added local feature vectors and one set MC for each remote host to store the matching candidates as reported by this host. Any of the SHA family of hashes seems appropriate to implement H, and we currently use SHA-256 as a secure hash.

CKP consists of two phases:

I *Collecting entropy from feature vectors and determining matching candidate key parts*: In step I.1, locally generated feature vectors are stored in a local history for future reference. This history LH may be implemented as a circular buffer, overwriting oldest feature vectors. By computing the secure hash, *candidate key parts* are created from these feature vectors and sent to the remote device in step I.2. Note that each round r uses a unique salt value s^r that is prepended before hashing to make attacks with lookup tables more expensive.

In step I.3, received candidate key parts are compared with feature vectors in the local history LH. All matching vectors are advanced to the status of *matching key parts* by adding them to the set of matching candidates MC, which is specific to each remote host that CKP is run with.

Step I.4 is optional, and should only be used in asymmetrical settings. In an asymmetrical setting, only one host broadcasts candidate key parts. Any host receiving the candidate key parts and recognizing matching key parts acknowledges these matches, which enables the broadcasting host to keep track of matching key parts.

II *Generating the secret shared key*: Each host can check locally if enough matching key parts have been collected, and/or if the associated feature vectors accumulate enough entropy for a secret shared key. When the local criteria are fulfilled, a *candidate key k* and an associated *candidate key*

identifier kh are generated in step II.1 by concatenating the feature vectors that belong to the last N matching key parts and, again, computing secure hashes over the concatenated string with prepended salt values. To decouple the actual key and its identifier, a public padding string C is appended before hashing for the generation of k. The candidate key identifiers are exchanged in step II.2.

In step II.3, the hosts then try to locally generate a key that matches a received candidate key identifier. This may be computationally expensive, depending on the number of matching key parts in MC, the number of matching key parts N used for the generation of kh, and the number of duplicate matches in each round. The reason is that, for example, host B has no knowledge about the exact set of matching key parts chosen by host A to generate its kh_a. Because hosts A and B may be out of sync with their round counters, it is unknown which rounds contributed matching key parts. And because A and B most probably generate candidate key parts in different order even within the same round, it is unknown which of the matches in a specific round was chosen when there were multiple. B therefore needs to try all possible combinations of N_a elements of $MC_{b,a}$, which has potentially a run time complexity of $O(A^H)$ where A is the maximum number of different candidate feature vectors generated in each round, and $H > N$ is the maximum size of the history MC. However, in practice we expect only few duplicates, and the search can be further optimized by starting with the most likely, i.e. the most recent, round numbers recorded in MC. Another possible optimization is to transmit the round and vector numbers with candidate key messages to uniquely identify the set of parts. This trade-off between message size and computational cost depends on application-specific cost models, but does not influence the security level of the protocol.

If a matching key could be generated, it is acknowledged in step II.4. After receipt of a *key acknowledge*, the hosts can start to use the generated key k that matches the acknowledged key identifier kh.

Note that at this stage, there is the possibility that the generated keys at hosts A and B are different. This can happen when hosts A and B independently generate and exchange candidate keys in steps II.1 and II.2 and the respective KEY messages overlap during transmission. Then, in steps II.3 and II.4, both hosts may find and acknowledge the respective remote host's key, again with overlapping ACK messages. That is, when host A generated a key k_1 and B a key k_2 in step II.1, then after step II.4, host A may have found and acknowledged k_2 while B may have found and acknowledged k_1. By concurrently reacting to overlapping messages, A and B have effectively swapped their keys, but are still left with different k_1 and k_2. To solve this synchronization problem locally, the hosts remember the originally generated keys and check if the received key acknowledge is different. If yes, they can simply compute the final secret shared key as the XOR of the two different keys.

In this form, CKP does not assume the communication channel to have any specific properties, because our basic assumption is a MITM with full control over this channel.

5   Implementing CKP with Lossy Channels

In a practical implementation, the communication channel may be lossy. That is, packet delivery is not guaranteed even when no MITM attack is taking place. This is the case for most broadcast radio frequency (RF) channels such as IEEE 802.11 WLAN or IEEE 802.15.4 ZigBee.

Our first implementation of CKP uses UDP as a lossy communication protocol. This has three advantages: a) UDP can be used directly between any hosts connected via an IP based network. b) UDP allows to broad- or multicast packets and can therefore be used for group authentication or spontaneous authentication as described in more detail below. c) UDP offers guarantees comparable to many low-level broadcast RF channels, thus porting our implementation e.g. to TinyOS [21], should be straightforward.

The protocol specification presented in section 4 lends itself to implementation on lossy channels, because it is robust against packet loss. When candidate key parts get lost, there will simply be no matches for the respective round. When candidate keys get lost, they can not be used to generate secret shared keys, but new candidate keys will be generated in subsequent rounds. However, issues arising from asynchronism and overlapping messages need to be dealt with at the implementation level.

There are various possibilities for asynchronism in CKP. Here we concentrate on the case where a remote message arrives for a round before the respective local action has been processed. This includes many special cases like a disparity between the system clocks or delayed processing due to multi-tasking. To cope with such asynchronism, we introduce message buffers to keep a history of recently received messages. Then, when local operations such as adding new feature vectors are processed, this history is replayed to simulate a new arrival of messages using the updated local state. This method allows to cope with asynchronism while considering limited resources in terms of memory and CPU capabilities.

Note that, among others, [7, Theorem 8] states that "perfect synchronization is impossible", i.e. that there are always some cases in which the decisions of Alice and Bob about generating a common key are different. Our implementation of CKP using UDP can only safeguard against Alice and Bob agreeing to a different shared key (i.e. a MITM attack). But, under the assumption of a completely insecure communication channel without any guarantees, it will always be possible for one host to finish the protocol with success, while the other finishes with failure. In this case, further secure communication is not possible, and the hosts can use time-outs to detect it.

6   First Experimental Results

CKP has already been applied to one specific device pairing method: implicit authentication by shaking devices together for a few seconds [22]. This method uses 3D accelerometers as input to two alternative authentication protocols, one of them being CKP. When shaking two devices with integrated accelerometers together, their sensor time series are similar enough to create a secret shared key, but an adversary can not obtain these time series with sufficient accuracy. The lower bound of the entropy rate has been estimated at about 7 bits per second [22], which means that around 20 s of shaking are sufficient to generate over 128 bits of entropy. Experiments on "human man-in-the-middle" attacks, where adversaries try to duplicate the shaking patterns of victims to produce similar sensor time series, show that people are unable to reproduce these patterns even when we allow for cooperation between adversary and victim (which would not be possible during a real attack). It remains to be shown if high-speed cameras could be used to estimate the local acceleration values and thus lower the candidate key parts entropy from an adversary's point of view.

7   Security Analysis and Discussion

When generating cryptographic key material, the most important point is to achieve high entropy with regards to a possible adversary's knowledge. A key can only remain secret if Eve is sufficiently uncertain about it. It is important to note that, principally, CKP can not increase the entropy of a secret key compared to the total entropy of all feature vectors it has been created from. Instead, any public communication between Alice and Bob must reveal something about the key — CKP can only try to make this additional information useless to Eve. Note that feature extraction and estimation of entropy are entirely application specific. We can only assume the locally added feature vectors to carry sufficient entropy, and leave it to the specific implementation to guarantee this.

Hashing the sensor time series to generate candidate key parts and candidate keys serves to reduce an adversary's usable information about them. This is often termed "privacy amplification". When we assume the SHA family of hash functions to be universal as defined in [23] and reproduced in [24], then an upper bound for the information that Eve can obtain about the secret key has been shown in [24, Corollary 5]: if Eve has access to t bits of the (weak) secret W with n bits, then her expected knowledge about a key $K = H(W)$ with a length of $r = n - t - s$ bits for some safety parameter $s < n - t$ is restricted to a maximum of $2^{-s}/\ln 2$. This assumes that W is uniformly distributed. For our application to sensor time series, W is not uniformly distributed, and a significant part of its distribution function may be known to Eve. We can only conjecture that the above corollary may be applicable to those components of the sensor time series that are completely unknown to Eve and thus uniformly distributed from her point of view, but can not currently provide a proof. This conjecture suggests that, if we intend to extract a secret shared key with a size of $r = 128$ bits,

then the difference between the length of the sensor time series W and Eve's information about it, i.e. $n - t$ bits, must be larger then 128. Intuitively, this requirement is trivial. But the theoretical analysis indicates that by hashing the input, all the entropy of the weakly secret sensor time series should be retained in K. This means that transmitting the candidate key parts, which are hashes over the sensor time series, should not reveal more about them than an adversary already knew. It is currently unclear if more information about the final secret key is revealed when MATCH messages are transmitted to acknowledge matching candidate key parts, but we do not expect this to be the case. Nonetheless, the normal mutual authentication mode seems more conservative, because only the candidate key part hashes and the candidate key hashes are transmitted, but no more information about which of the candidate key parts have been used to construct the secret key.

It is important to note that there is a possibility for brute-force attack. The problem arises when parts that are extracted from sensor time series only have a small entropy from Eve's point of view. In this situation, even when reversing the hash function is impossible, she could just generate lookup tables of all possible time series parts and compare their hashes with the CAND messages. This is slightly mitigated by our use of salting, but only makes the attack more computationally expensive, and not less likely to be successful. Eve only needs to keep a small amount of possibly matching key parts in a history to try and create keys that match the transmitted KEY messages, in much the same way that is also used in the legitimate protocol run. For this reason, it is better to use less candidate key parts to construct a key. When the sensor time series parts that can be extracted naturally using domain specific knowledge only have a small entropy, then multiple such parts should be buffered and bundled into one candidate key part. Guessing a candidate key part and verifying that it matches its received hash value has an average complexity of $O\left(2^{e-1}\right)$ when the feature vector has e bits of entropy from Eve's point of view. Thus, two concatenated feature vectors would need $O\left(2^{2e-1}\right)$ steps to guess. This entropy level directly defines the security level of the whole CKP run. It has been shown in [24] that adding random material can in principle increase the length of K that can be extracted from the weak secret W, but we currently do not see a method to apply this to CKP.

Finally, there are two additional advantages of CKP over more traditional authentication protocols, e.g. ones based on public key infrastructures. First, the continuous broadcasting of candidate key parts and, after detecting matches, of candidate keys, allows remote hosts to "tune in to" another host's authentication stream. This allows to easily construct applications with *opportunistic authentication*, where hosts automatically authenticate with each other as soon as they enter a shared context: when a host picks up broadcasts from another and is able to generate matching key parts, they are guaranteed to record similar sensor readings.

Second, CKP can be trivially generalized to group authentication. In the specification in Fig. 3, only steps II.4 and II.5 need to be adapted. All hosts

can continue to generate candidate key parts and candidate keys, and to search for candidate keys as for two-host authentication. However, keys can only be acknowledged and used in steps II.4 and II.5, respectively, after all hosts that should be members of the authenticated group were able to generate matching keys. A possible solution is to split step II.4 into a *tentative acknowledge* and a *group acknowledge* message, where the latter is only sent after the tentative acknowledge has been received from all group members.

8   Conclusions

Our proposed Candidate Key Protocol (CKP) is one approach to solving the problem of device-to-device authentication for spontaneous interactions. Replacing *explicit* means of authentication like manual password input or string verification with an *implicit* authentication based on similar sensor data streams offers significant advantages from a user point of view: wireless communication can be made secure by default instead of relying on a separate authentication step.

Context-based authentication is in fact a classification problem, with the known problems of false positives, which need to be strictly avoided for security reasons, and false negatives, which hinder seamless and unobtrusive user interaction. One main novelty of CKP is that multiple candidate key parts in each step can be used to address the problem of false negatives. Its advantages over other proposed approaches to the same problem and based on Diffie-Hellman key agreement authenticated by short, or weak, shared secrets are threefold: it is less computationally expensive and thus well suited for implementation with limited resources, it provides opportunistic authentication, and it is trivially extensible to group authentication. The major disadvantage is that the generated secret shared key is only as secure as the entropy of the candidate key parts and that it does not provide forward secrecy. Newer results on information theoretically secure key agreement are very promising for authentication based on sensor data streams, but have not yet been implemented in practice. Relying on conventional secure hashes allows us to implement and test CKP in real-world settings like the authentication method based on shaking devices together.

Complete source code of our current Java implementation is available at `http://www.openuat.org`, as part of the open source ubiquitous authentication toolkit.

Acknowledgments

References

1. Weiser, M.: The computer of the twenty-first century. Scientific American **1496** (September 1991) 94–100
2. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. on Information Theory **IT-22**(6) (1976) 644–654
3. Maurer, U.M.: Perfect cryptographic security from partially independent channels. In: Proc. STOC '91: 23rd ACM Symp. on Theory of Computing, ACM Press (May 1991) 561–571
4. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: Proc. 6th ACM Conf. on Computer and Communications Security, ACM Press (1999) 28–36
5. Juels, A., Sudan, M.: A fuzzy vault scheme. Cryptology ePrint Archive, Report 2002/093 (July 2002)
6. Dodis, Y., Smith, A.: Correcting errors without leaking partial information. In: Proc. STOC '05: 37th ACM Symp. on Theory of Computing, ACM Press (May 2005) 654–663
7. Maurer, U., Wolf, S.: Secret-key agreement over unauthenticated public channels — part i: Definitions and a completeness result. IEEE Trans. on Information Theory **49**(4) (2003) 822–831
8. Maurer, U., Wolf, S.: Secret-key agreement over unauthenticated public channels — part ii: The simulatability condition. IEEE Trans. on Information Theory **49**(4) (2003) 832–838
9. Maurer, U., Wolf, S.: Secret-key agreement over unauthenticated public channels — part iii: Privacy amplification. IEEE Trans. on Information Theory **49**(4) (2003) 839–851
10. Renner, R., Wolf, S.: Unconditional authenticity and privacy from an arbitrarily weak secret. In: Proc. CRYPTO 2003, Springer-Verlag (2003) 78–95
11. Boyko, V., M., P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. Cryptology ePrint Archive, Report 2000/044 (2000)
12. Rivest, R.L., Shamir, A.: How to expose an eavesdropper. Commununications of ACM **27**(4) (1984) 393–394
13. Gehrmann, C., Mitchell, C.J., Nyberg, K.: Manual authentication for wireless devices. RSA Cryptobytes **7**(1) (2004) 29–37
14. Pasini, S., Vaudenay, S.: An optimal non-interactive message authentication protocol. In: Topics in Cryptology - Proc. CT-RSA 2006, The Cryptographers' Track at the RSA Conf. 2006, Springer-Verlag (2006) 280–294
15. Stajano, F., Anderson, R.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Proc. 7th Int. Workshop on Security Protocols, Springer-Verlag (April 1999) 172–194
16. Hoepman, J.H.: The emphemeral pairing problem. In: Proc. 8th Int. Conf. Financial Cryptography, Springer-Verlag (February 2004) 212–226
17. Hoepman, J.H.: Ephemeral pairing on anonymous networks. In: Proc. SPC 2005: 2nd Int. Conf. on Security in Pervasive Computing, Springer-Verlag (April 2005) 101–116
18. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Proc. CRYPTO 2005, Springer-Verlag (August 2005)
19. Creese, S., Goldsmith, M., Harrison, R., Roscoe, B., Whittaker, P., Zakiuddin, I.: Exploiting empirical engagement in authenticated protocol design. In: Proc. SPC 2005: 2nd Int. Conf. on Security in Pervasive Computing, Springer-Verlag (April 2005) 119–133
20. Čagalj, M., Čapkun, S., Hubaux, J.P.: Key agreement in peer-to-peer wireless networks. IEEE (Special Issue on Cryptography and Security) **94** (2006) 467–478
21. TinyOS Alliance: TinyOS web page. `http://www.tinyos.net` (2006)
22. Mayrhofer, R., Gellersen, H.: Shake well before use: Authentication based on accelerometer data. In: Proc. Pervasive 2007: 5th International Conference on Pervasive Computing, Springer (May 2007) *to appear*.
23. L.Carter, M.Wegman: Universal classes of hash functions. Journal of Computer and System Science **18** (1979) 143–154
24. Bennett, C.H., Brassard, G., Crépeau, C., Maurer, U.: Generalized privacy amplification. IEEE Transaction on Information Theory **41**(6) (1995) 1915–1923