UACAP: A Unified Auxiliary Channel Authentication Protocol

Rene Mayrhofer, Jürgen Fuß, Iulia Ion

Abstract— **Authenticating spontaneous interactions between devices and users is challenging for several reasons: the wireless (and therefore invisible) nature of device communication, the heterogeneous nature of devices and lack of appropriate user interfaces in mobile devices, and the requirement for unobtrusive user interaction. The most promising approach that has been proposed in literature involves the exploitation of so-called auxiliary channels for authentication to bridge the gap between usability and security. This concept has spawned the independent development of various authentication methods and research prototypes, that, unfortunately, remain hard to compare and interchange and are rarely available to potential application developers. We present a novel, unified cryptographic authentication protocol framework (UACAP) to unify these approaches on using auxiliary channels and analyze its security properties. This protocol and a selection of auxiliary channels aimed at authentication of mobile devices has been implemented and released in an open source ubiquitous authentication toolkit (OpenUAT). We also present an initial user study evaluating four of these channels.**

Index Terms— **C.2.1 Network Architecture and Design/Wireless Communication, D.4.6.b Security and Protection/Authentication, H.4 Information Systems Applications/Miscellaneous, C.2.8.a Algorithm/protocol design and analysis, C.3.h Ubiquitous computing, C.2.8.d Mobile environments, H.2.b Human-centred computing, J.9 Mobile Applications**

I. Introduction

Security in Ubiquitous Computing is currently a hot topic; as many research projects mature and their core findings start to influence real-world applications, non-functional requirements become increasingly more important. Security is one of the most important among these non-functional requirements and is a prerequisite to wide deployment.[1] Using standard cryptographic approaches, many security requirements – for example confidentiality, integrity, non-repudiability, auditability, or access control – can be fulfilled once all involved parties have been successfully authenticated. *Authentication* is therefore the key requirement to secure any interaction. Within the vision of ubiquitous computing, this is a particularly challenging task mostly due to three main reasons: (1) wireless communication channels are insecure, (2) many devices lack sufficiently capable user interfaces, and (3) user attention does not scale. One commonly cited example is the serendipitous use of available infrastructure such as printers or projectors (which typically do not offer any directly accessible user input method and are under different administrative control) from mobile devices such as mobile phones (which offer some input method that is often difficult, erroneous, or annoying for longer input, but are trusted personal devices), being connected via untrusted wireless links (such as an open guest WLAN).

These problems have, over the past years, spawned the development of different authentication methods for specific application areas (e.g. [3]–[15]). However, most of these efforts are still separate and thus difficult to compare and not interchangeable. Actual implementations are often unavailable or otherwise restricted to specific, prototypical demonstration applications. This hinders both additional research on authentication methods for ubiquitous computing and application developers using those that have already been suggested. Furthermore, usability studies have shown that no one single device pairing method can be satisfactory for all users in all situations. Users prefer different methods in different situations, depending on the data they want to exchange, type of device they connect to, and the person operating the other device [16]. We therefore propose a unified basis for comparable and interchangeable protocols and a library of ready-made implementations aimed at deployment outside of research projects. To this end, we previously suggested to develop an open source toolkit implementing some of these methods with a common structure [17].

We use the following application scenarios as motivating examples that illustrate different potential use-cases for our system. In all these cases, selecting the intended communication partner is a major issue, with potentially tens of different wireless networks and hundreds of unknown devices in these networks. Depending on the lifetime of the keys exchanged, we distinguish *short-lived* associations (ephemeral "one-shot" keys, used just once) and *long-lived* pairings (keys are stored and reused for future communication between the devices).

1) *Exchanging vCards and PGP keys:* Alice and Bob meet at a conference and wish to exchange contact information (vCards) and PGP keys for future remote communication. This exchange is short-lived, as their mobile phones are unlikely to directly communicate again. A specific issue is spontaneous authentication with severely limited user interfaces and highly personal devices that users might not wish to hand over.
2) *Printing a confidential document on a Bluetooth printer:* Alice is in the airport and wishes to use the printer in her waiting lounge to print multiple parts of a confidential report saved on her phone. Keys may be reused for printing separate documents after initial establishment. A specific issue is selecting the "correct" printer in a list of similar ones.
3) *Connecting a mobile phone to a Wi-Fi router:* While visiting his friend's house, Bob wishes to check his email. He

Some material presented in this paper previously appeared in a habilitation thesis [1] and a technical report [2].

[1] Unfortunately, this is often neglected and product manufacturers as well as standardisation bodies often try to retrofit security measures onto otherwise completed projects. The recent track record shows that this procedure is clearly unsuccessful in producing secure systems.

connects his smart phone to his friend's wireless LAN. Keys may be long-lived to be reused on future visits, but authorisation may be limited to the actual visits. A specific issue is that the wireless access point might not be directly accessible.

4) *Temporarily pairing a mobile phone and Bluetooth headset:* Alice wishes to talk on the phone while driving and borrows a Bluetooth headset to make a single call. Keys are short-term, because she returns the headset after use. A specific issues is that a headset typically has no user interface other than a single button (and audio).
5) *Permanently pairing a mobile phone and Bluetooth headset:* Alice wishes to listen to music using her mobile phone and a stereo Bluetooth headset.

In this article, we present a *Unified Auxiliary Channel Authentication Protocol (UACAP)* that can be used to securely pair heterogeneous devices and/or services with limited input/output capabilites (such as mobile phones and infrastructure components) by relying on diverse auxiliary channels. We also release a specific implementation in the form of the *Open source Ubiquitous Authentication Toolkit (OpenUAT)*. The main contributions are:

- We propose a unification of previously suggested, originally separate protocols into a common protocol framework that is configurable towards the specific security properties of the chosen auxiliary channel (Section III). A reference implementation of UACAP uses arbitrary wireless (in-band) channels and an ASCII-based protocol specification (Section V).
- We further identify three main categories of auxiliary channels (*input*, *transfer*, and *verification*) with sub-categories concerning channel capabilities (confidential vs. non-confidential and short vs. long in terms of channel bandwidth) as the basis for classifying auxiliary channels both from a user interaction and a security point of view (Section III). This classification informs the different options for UACAP and may help application designers select appropriate channels according to their user interaction model and security requirements. A few specific auxiliary channels from different categories have been implemented in prototypical form (Section V).
- Based on these auxiliary channel categories and previous work on manual authentication protocols, we provide a security analysis of UACAP and show that it is secure against currently known attack scenarios under standard assumptions on wireless and auxiliary channels (Section IV).

II. Related Work

Over the last few years, different cryptographic protocols for multi-channel authentication have been proposed (e.g. [18]–[25]). Most of them have in common that they assume a main wireless communication channel and a more restricted, so-called *auxiliary* or *out-of-band* channel. While the main channel has – in terms of cryptographic key exchange – practically unlimited bandwidth, the auxiliary channel is often limited to either short messages or slow and/or obtrusive transfer. Consequently, different cryptographic protocols have been developed to exploit these diverse characteristics for the purpose of secure authentication between devices, users, and services (a good summary of the most important protocol proposals can be found in [19]). However, this diversification of protocols means that they are not easily interchangeable and that security analysis need to be done for each of them. In the present paper, for the first time, we contribute a unified protocol that can exploit any combination of security guarantees from arbitrary auxiliary channels. UACAP is a unification of some of the recently proposed protocols, retaining their security properties with a minimal number of messages.

A considerable amount of prior work on using auxiliary channels to establish shared secret keys between two (or multiple) devices has been presented. The "resurrecting duckling" as a pairing model suggested direct electrical contact [26], while "constrained channels" [27] and "location-limited channels" [20] were proposed as more general models of auxiliary channels for authentication purposes. Specific auxiliary channels are *video* by using mobile phone cameras and 2D barcodes [3], blinking patterns [28], face matching [15], or laser channels [11], *audio* by relying on ambient audio [29], comparing spoken sentences [4] or MIDI tunes [5], *ultrasound* [10], *motion* by common movement [6], [7], gestures [12], or synchronised button presses [8], or *radio frequency* by measuring common environment [9] or relying on trusted third parties [14]. Part of this research is slowly moving into products. The Bluetooth Simple Secure Pairing (SSP) [30] and the Wi-Fi Protected Setup (WPS) [31] already use some of the results on pairing protocols, although initial implementations will be limited to standard display and keypad entry methods. Envisaged applications are pairing Bluetooth mice or keyboards, which currently use empty (and therefore insecure) passwords, with laptops or desktop PCs. So far, however, these approaches are completely separate with different cryptographic protocols and implementations. With the toolkit currently being extended, we aim to implement as many auxiliary channels as possible in a common structure so that they are comparable and interchangeable.

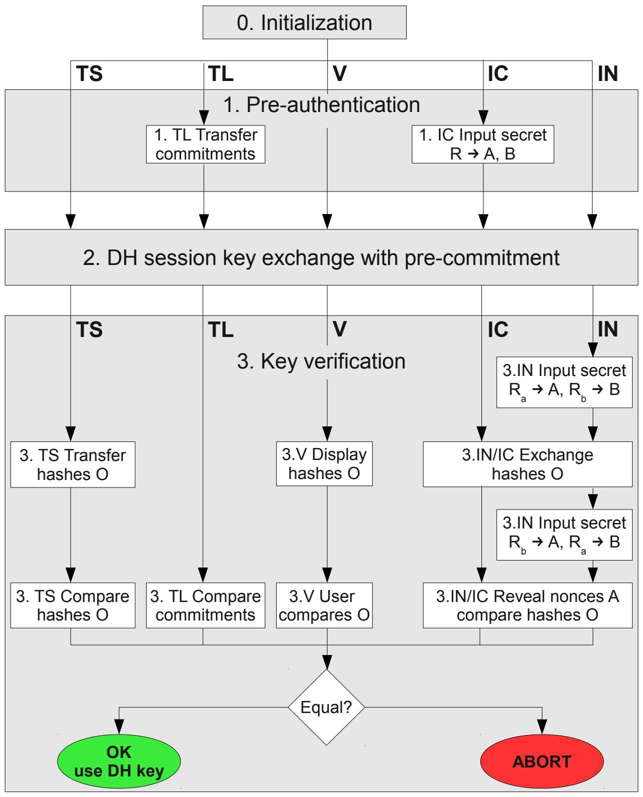
III. Unified Auxiliary Channel Authentication Protocol (UACAP)

The *Unified Auxiliary Channel Authentication Protocol* (UACAP) is based on the recent proposal by Laur and Nyberg called MA-DH [19], but adopts aspects of the MANA III variant described by Wong and Stajano [25] and an option for pre-authentication as suggested, among others, by Balfanz et al. [20]. Its main part relies on the well-known Diffie-Hellman (DH) key agreement [32] with a prior one-way commitment to prevent basic man-in-the-middle (MITM) attacks. The result of this main phase is a secret session key shared between two devices. Then, to ascertain that there is no other device involved (e.g. by accidentally pairing with the wrong device or malicious MITM attack), the obtained key must be authenticated using the properties of an auxiliary channel.

Due to its reliance on DH key agreement, UACAP offers perfect forward secrecy (PFS) concerning attacks on previous (or future) session keys. However, this means that UACAP can not, in its current form, be used on devices that are incapable of running DH during connection establishment, effectively excluding the lowest-end devices such as RFID tags or some sensor nodes with significantly restricted CPU resources. For these classes of devices, protocols relying solely on symmetric cryptography such as CKP [13] are more appropriate — at the cost of reduced resistance against off-line brute-force attacks on weak shared secrets. However, most mobile devices are already capable of using DH key agreement, including Bluetooth headsets, keyboards, and comparable low-level auxiliary devices (as evidenced by their support of Bluetooth SSP).

Depending on the application scenario and properties of the auxiliary channel, UACAP supports different modes of operation from a user point of view:

- **Input** channels allow the user to provide common input to all involved devices, for example by explicit PIN code entry (cf. [18]), synchronous button presses (cf. [8]), or shaking them together (cf. [6], [7]). We need to further distinguish if the user input can be shielded from others or not:
  - **IN:** *Non-confidential* input must happen interactively during the protocol run. A potential use case is application scenario 1 presented earlier, i.e. exchange of vCards between mobile phones; users may be unwilling to hand over their own devices for a direct contact or common movement, but can easily exchange and enter 4-digit PIN codes.
  - **IC:** *Confidential* "pre-authentication" is possible before the main protocol part and with no further interaction on the auxiliary channel. This is the only case in which the auxiliary channel must be confidential; in all other protocol cases its authenticity is sufficient. Application scenarios 4 and 5 lend themselves well to this mode of authentication from a user point of view, e.g. by shaking both the mobile phone and the headset together for a few seconds and therefore providing common (partially confidential) input.
- **Transfer** channels support direct, user-mediated (and ideally human-verifiable) transmission of messages, for example by capturing a 2D barcode displayed on one device with the camera of another (cf. [3]) or audible MIDI sequences (cf. [5]). We further distinguish according to the auxiliary channel bandwidth:
  - **TS:** *Short* transfer (32–64 bits) must happen interactively during the protocol run. Application scenario 2 is a potential candidate for this type of interaction; the printer could display a short string, which the user would then enter on their mobile device.
  - **TL:** *Long* "pre-authentication" ($\geq$160 bits) is possible before the main protocol part and with no further interaction on the auxiliary channel (non-interactive with respect to the auxiliary channel). This has the advantage that, by taking place before any communication on the main wireless channel, required addresses (for example MAC or IP addresses) may also be transmitted in the same pre-authentication message to support easy-to-use device selection methods. This is well suited for many use cases, including all of our suggested application scenarios: a mobile phone could display a 2D barcode that includes its Bluetooth MAC address and its public key part when initiating a vCard transfer, or a static 2D barcode could be printed on the case of a printer, Wi-Fi router, or Bluetooth headset.
- **Verification** channels allow the user to compare data from different devices, for example by reading non-sensical English sentences (cf. [4]), comparing random visual art (cf. [33]), or MIDI tunes (cf. [5]). We denote this option:
  - **V:** Explicit user verification can always use "short" bit strings in the range of 32–64 bits over a public, non-confidential medium. This is suitable for all use cases in which both devices feature compatible output devices, such as application scenarios 1 and 2 or scenarios 4 and 5 if the Bluetooth headset is equipped with a small display or capable of producing text-to-speech output.



0. Initialization
TS
TL
V
IC
IN
1. Pre-authentication
1. TL Transfer commitments
1. IC Input secret R → A, B
2. DH session key exchange with pre-commitment
3. Key verification
3.IN Input secret R_a → A, R_b → B
3. TS Transfer hashes O
3.V Display hashes O
3.IN/IC Exchange hashes O
3.IN Input secret R_b → A, R_a → B
3. TS Compare hashes O
3. TL Compare commitments
3.V User compares O
3.IN/IC Reveal nonces A compare hashes O
Equal?
OK use DH key
ABORT

Fig. 1
UACAP OVERVIEW: DIFFERENT OPTIONS FOR KEY VERIFICATION

These options depend on the choice of authentication mode (transfer, verify, or input) and channel type (short or long, confidential or non-confidential) and may be predetermined by the respective application scenario (e.g. direct input is not possible when interacting with non-accessible infrastructure devices like large, distant screens or projectors). By supporting all options, UACAP does not limit application designers by enforcing a particular user interaction mode, but allows to choose those that best match the application. Central to the protocol is a DH key agreement that is universal among all choices of modes and channels.

In the following, we describe UACAP on three levels: First, the overview in Fig. 1 depicts the logical flow of UACAP, pointing out the decision points and different options at different steps, until reaching the final decision; second, Fig. 2 presents a detailed specification on the level of cryptographic operations and values exchanged in messages; third, we describe the current reference implementation in terms of on-the-wire message transfer and protocol commands in Section V-A.

A. Protocol Specification

For the formal description in Fig. 2, the following notation is used: $H(m)$ describes the hashing of message m with some secure hash function H, and $m\|n$ the concatenation of strings m and n. $\mathrm{SHA_{DBL}}$-256 is used as a secure hash function, it is a double execution of the standard SHA-256 message digest to safeguard against length extension and partial-message collision attacks [34] and is defined as $\mathrm{SHA_{DBL}\text{-}256}\,(m) = \mathrm{SHA\text{-}256}\,((\mathrm{SHA\text{-}256}\,(m))\,\|m)$. $\mathrm{Trunc\text{-}Hash}(x\|k)$ is a truncated hash value over x using key k of particular small size; the proof of security of MANA IV in [19] is based on the use of (≥ 32) bits of $h(x\|k)$ for a particular class of hash functions h (cf. [19, Sec. 5] and a discussion of practical proposals for h such as standard hash functions like $\mathrm{SHA_{DBL}}$-256). Another possible implementation is to select the first few (≥ 32) bits of an HMAC as suggested in Appendix B of [35]: $\mathrm{Trunc\text{-}Hash}(x|k) := \mathrm{HMAC}_k(x) := H\,(k \oplus opad\|H\,(k \oplus ipad\|x))$ with two constants $opad$ and $ipad$ and a suitable hash function H (e.g. SHA-256). $Com()$ denotes a cryptographic commitment. $Open(c, x, p)$ is the opening function of $Com(x, r)$: if $Com(x, r) = (c, p)$, then $Open(c, x, p)$ evaluates to TRUE. A possible implementation is $Com(x, r) := (\mathrm{HMAC}_r(x), r)$ where the hash value is verified in the opening phase after revealing the ephemeral key and a random nonce r. In [19], a CCA2 secure encryption scheme is recommended to be used as a commitment scheme; the use of an HMAC based commitment as well as of a hash commitment based on OEAP padding are discussed as practical alternatives. The consequence of using a simple hash commitment with $Com(x, r) := (\mathrm{SHA_{DBL}\text{-}256}(x), \mathrm{nil})$ is that $M_{3,b}$ is empty and may be omitted. For the Diffie-Hellman parameters we assume that g is selected from a subgroup of prime order q in a suitable group. Possible parameters are mentioned in [36]. Subscripts denote the different sides (a or b for an authentication between A and B). The notation $\widetilde{X}$ is used to point out that a value X has been sent over an insecure channel and may therefore have been modified (by transmission error or malicious behaviour).

In Fig. 2, the different options are indicated in the heading of specific steps and are only executed for the respective channel type. Mandatory parts are indicated with a grey background. Channels over which messages are transmitted are either the main wireless channel (RF), an authentic but "short" (AS), authentic and "long" (AL), or an authentic and confidential but "short" (ACS) auxiliary channel. Note that all auxiliary channels must always provide authenticity, which is the primary reason for their use in authenticating device interaction.

The protocol execution consists of the following steps:

a) Initialization: This phase is common to all protocol options (TL, TS, V, IN, IC) and initializes the ephemeral DH parameters (the domain parameters are assumed to be publicly known) and identities of the involved parties. The (potentially ephemeral) identities of the parties A and B, for example their network addresses, are represented by I_a and I_b, respectively.

b) Pre-authentication: In this phase, pre-authentication data is used before starting the actual key exchange protocol. This has the advantage that the actual protocol run can be non-interactive and thus even more unobtrusive to the user. It also means that the part requiring user involvement (transfer or input) can happen at any time before the associated devices start their wireless interaction. Pre-authentication is only supported in the following two cases:

- **TL**: Commitments are exchanged over the authentic channel. Observe that the commitment of party A is sent in Phase *2. DH session key exchange with pre-commitment*. Thus in this phase only B's commitment has to be sent. This step is depicted as *1.TL Transfer commitments*.
- **IC**: A common, short secret R is input to both devices on a confidential channel (*1.IC Input secret*). R *must* remain confidential until the protocol finishes. Such confidential user input can be sensor data such as the accelerometer time series resulting from shaking devices together, or the same password/PIN entered on all devices (and shielded from others).

c) Diffie-Hellman key exchange with pre-commitment: This part is mandatory and common to all protocol options. Optional parameters for specific protocol instances (for example the number of rounds of an interlock scheme as in the ultrasonic spatial authentication protocol [10]) can be transmitted from the initiator A to the responder B in plain text using the optional value P_a. If not required, it may simply be omitted in the first message (indicated with the notation $[P_a]$). Having the initiator form and send a correct commitment before the responder's ephemeral DH is initialised, denial-of-service attacks become (marginally) harder.

The session key $K_a = K_b$ is not used during the verification phase. Ensuring that the correct values X and Y have been transmitted is sufficient to ensure device authentication. In all options of the protocol, explicit key validation for the session key can be achieved via additional encryption of known data with the session keys to detect errors in the shared key derivation (note that this is not a security measure, but only for error detection purposes).

d) Out-of-Band (OOB) key verification: The first part initializes the components for the key verification while the actual OOB verification depends on the chosen protocol option.

- **TS**: Hashes O_a and O_b are transferred over the "authentic but short" channel and subsequently compared with the local ones. If they match on both devices, the key exchange completes successfully.
- **TL**: Commitments of the DH parameters have been exchanged in the pre-authentication phase. For verification, it suffices to open those commitments.
- **V**: Verification hashes O_a and O_b are displayed to the user for comparison. This can be done e.g. by displaying hexadecimal hash representations, showing MADLib sentences, playing MIDI tunes, etc. The user inputs OK if hashes match, otherwise the protocol fails.
- **IN/IC**: If the (short) user input values R_a and R_b can be guaranteed to remain confidential until the protocol run finishes, they may be provided to both devices at the same time and before even starting the protocol, and R_a and R_b may be equal. If R_a and R_b are only authentic but not confidential (and if they are "short" in terms of brute-force attacks), then they *must* be different and *must not* be made public before Step 3 in any case. At this time, they *must* be provided to the respective other device (that is, swapping the different values entered in Step *1.IC*), but may be made fully public.

  Only in the **IN** case, the user must first input R_a to device A and a different R_b to device B. Next, devices compute and exchange hashes Q_a and Q_b. Nonces J_a and J_b are used to protect against replay attacks. Steps *3.IN/IC* in Figure 2 are common to the input case, both for confidential and non-confidential subcases, and must be executed in order. For

Channel	**A (initiator)**	Message	**B (responder)**
		0. INITIALIZE	
	$I_a \in \{0,1\}^{128}$ $x \in_R \{1 \ldots q-1\}$ $X := g^x$		$I_b \in \{0,1\}^{128}$ $y \in_R \{1 \ldots q-1\}$ $Y := g^y$
		[1.TL: TRANSFER COMMITMENTS]	
AL		$\mathbf{M}_{0,b} := (I_b, C_Y)$ $\longleftarrow$	choose r randomly $(C_Y, p_Y) := Com(Y, r)$
		[1.IC: INPUT SECRET]	
ACS	INPUT $R_a = R$		INPUT $R_b = R$
		2. KEY EXCHANGE (DH) WITH PRE-COMMITMENT	
RF	choose r randomly $(C_X, p_X) := Com(X, r)$	$\mathbf{M}_1 := (I_a, C_X[, P_a])$ $\longrightarrow$	
RF	$K_a := H(\widetilde{Y}^x)$	$\mathbf{M}_2 := (I_b, Y)$ $\longleftarrow$	

RF		$\mathbf{M}_{3,a} := (X, p_X)$
		$\longrightarrow$
			if $Open(\widetilde{C}_X, \widetilde{X}, \widetilde{p_X})$
			then $K_b := H(\widetilde{X}^y)$
			else ABORT

		[3.TL: OPEN COMMITMENTS]	
RF	if not $Open(\widetilde{C}_Y, \widetilde{Y}, \widetilde{p_Y})$ then ABORT else OK	$M_{3,b} := (p_Y)$ $\longleftarrow$	
		3. OOB KEY VERIFICATION	
	$OI_a := (I_a \| \widetilde{I}_b)$ $OK_a := (X \| \widetilde{Y})$		$OI_b := (\widetilde{I}_a \| I_b)$ $OK_b := (\widetilde{X} \| Y)$
		[3.V/TS: COMPUTE HASHES]	
	$O_a :=$ Trunc-Hash$(OI_a \| OK_a)$		$O_b :=$ Trunc-Hash$(OI_b \| OK_b)$
		[3.V: DISPLAY HASHES, USER VERIFY]	
AS AS	OUTPUT O_a INPUT OK iff $O_a = O_b$		OUTPUT O_b INPUT OK iff $O_a = O_b$
		[3.TS: TRANSFER HASHES]	

AS		$\mathbf{M}_{4,a} := (O_a)$
		$\longrightarrow$
			if $\widetilde{O}_a \neq O_b$
			then ABORT
			else OUTPUT OK

AS	INPUT OK iff B did not ABORT (without transfer from B to A)		

AS	if $\widetilde{O}_b \neq O_a$
	then ABORT
	else OUTPUT OK
		$\mathbf{M}_{4,b} := (O_b)$
		$\longleftarrow$	

AS			INPUT OK iff A did not ABORT (without transfer from A to B)
		[3.IN: NON-CONF. INPUT]	
AS	INPUT R_a		INPUT R_b
		[3.IN/IC: EXCHANGE HASHES]	
	$J_a \in_R \{0,1\}^{128}$ $Q_a := \text{HMAC}_{J_a}(OI_a \| OK_a \| R_a)$		$J_b \in_R \{0,1\}^{128}$ $Q_b := \text{HMAC}_{J_b}(OI_b \| OK_b \| R_b)$
RF		$\mathbf{M}_{5,a} := (Q_a)$ $\longrightarrow$	
RF		$\mathbf{M}_{5,b} := (Q_b)$ $\longleftarrow$	
		[3.IN: NON-CONF. INPUT EXCHANGE]	
AS	INPUT R_b		INPUT R_a
		[3.IN/IC: REVEAL NONCES AND COMPARE]	
RF		$\mathbf{M}_{6,a} := (J_a)$ $\longrightarrow$	
RF		$\mathbf{M}_{6,b} := (J_b)$ $\longleftarrow$	
	if $\text{HMAC}_{\widetilde{J}_b}(OI_a \| OK_a \| R_b) \neq \widetilde{Q}_b$ then ABORT else OK		if $\text{HMAC}_{\widetilde{J}_a}(OI_b \| OK_b \| R_a) \neq \widetilde{Q}_a$ then ABORT else OK

Fig. 2
UNIFIED AUXILIARY CHANNEL AUTHENTICATION PROTOCOL (UACAP) SPECIFICATION

the confidential case **IC**, the common secret $R = R_a = R_b$ has been input to both devices during the pre-auhentication phase and no further input is required. Only in the **IN** case, the user now inputs R_b to device A and R_a to device B. Finally, nonces J_a and J_b are made public, and the hashes Q_a and Q_b are verified by A and B.

IV. UACAP Security Analysis

In the following section, we analyse the security properties of UACAP. Being based on previously suggested protocols, UACAP carries over their security properties.

We are assuming a Dolev-Yao attacker on the wireless channel; an adversary attacking the pairing protocol is assumed to have full control of the wireless channel, that is, they can eavesdrop, delay, drop, replay, and modify messages. The OOB channel is assumed to be authentic and stall-free; the adversary cannot replace bits or arbitrarily delay messages, but may typically eavesdrop without restriction unless mentioned otherwise (in the **IC** case, the input channel is also assumed to be confidential).

The main feature that UACAP adapts from the MANA IV family is the commitment prior to Diffie-Hellman key agreement in Phase 2 to prevent *brute-force* attacks on short authentication strings. This one-way commitment to ephemeral public Diffie-Hellman key parts X and Y together with (potentially ephemeral) identifiers I_a and I_b used as session identifiers ensures that any MITM attack must be performed *online*. An adversary is reduced to either attacking DH keys once the commitment and key exchange have been completed (passive attack, which is currently assumed to be infeasible) or a single, one-off chance for fabricating the DH key parts in such a way that the (short) auxiliary messages will still match (active attack). Because of this protocol design element, all auxiliary messages besides the pre-authentication case (i.e. O_a and O_b) may be "short" in terms of brute-force key search. With only 32 bits, an adversary is left with a single 2^{-32} chance to remain undetected during an online attack, which is acceptable for most scenarios.

*a) **V**:* The design of the **V** option is consistent with the MANA IV family of protocols, more precisely the MA-DH protocol in [19], which provide formal security proofs. MA-DH is the optimal variant in terms of number of messages. The proof of security of this protocol in [19] is based on the assumptions that

- the hash functions[2] used are almost universal and perfect hash functions. For practical reasons we adopted their choice of a standard cryptographic hash function (SHA_{DBL}-256).
- the commitment scheme used is non-malleable. Laur and Nyberg in [19] recommend CCA2 secure encryption and discuss the use of hash commitment based on OAEP padding; as a practical alternative they propose an HMAC based hash commitment. Again, for practical reason we used the latter variant for an implementation.

V may be susceptible to values of O_a and O_b that are sufficiently similar for human users to confuse them (but not equal) given a suboptimal encoding (e.g. hexadecimal strings). This is a well-known shortcoming of protocols that require users to perform verification of some representation of bit strings. It is the task of the auxiliary channel to convey O_a and O_b to users in a way that they are hard to confuse even for values that are different by only a few bits.

*b) **TS**:* The **TS** option differs from the **V** option only in the final verification step, where the values O_a and O_b are exchanged over the auxiliary channel. Because the message transfer through the authentic channel is done at the end of the protocol, an attacker Eve would at least have to find a value z and identity I_c, such that $\text{Trunc-Hash}(OI_a \| OK_a) = \text{Trunc-Hash}(OI_c \| OK_c)$, where $OK_c := (X \| Z)$ and $Z = g^z$. This is considered infeasible under the assumption of the unique key property of Diffie-Hellman and the second-pre-image resistance of Trunc-Hash and is therefore reduced to guessing with a success probability of 2^{-l} for l-bit messages.

*c) **TL**:* The protocol starts with pre-authentication of both parties, through the exchange of commitments on an authentic channel. Commitments are then opened in Phase 3. This option is consistent with pre-authentication as in the SiB protocol described in [3, Sec. 4] inheriting its security properties. To impersonate Alice (A), an attacker Eve would have to either find Alice's private key x (i.e. solve a discrete logarithm problem) or find values z and p such that $Open(C_X, g^z, p)$ which is not feasible for a binding commitment function (even when Alice's public key is known).

"Pre-authentication" was previously described for SiB [3] and by Balfanz et al. [20]. Because pre-authentication requires transfer of "long" messages that are effectively secure hashes of public DH key parts, authentication is secure as long as these commitments remain so (that is, no second pre-image can be found online during the protocol run). In practice, this may be realized by pre-fabricated messages M_0, b for transferring both the identity of B (the responder) and the commitment to its public DH part, e.g. in the form of printed QR codes on the respective device case, business cards, etc. or infrared beacons replaying the same message in a loop. In this case, y, Y, and I_b are static. Consequently, either I_a or x and X need to be ephemeral (chosen randomly for each protocol run) to protect against offline brute-force attacks (cf. Section IV-A) and retain the security properties of UACAP.

*d) **IN/IC**:* From the "VIC" protocol proposal that combines MANA with SiB [28], BEDA [8] and Wong and Stajano's MANA III variant [25], we adopt the second round of mutual commitments (3.IN/IC Exchange hashes) for the "short input" case. Again, security arguments presented for these protocols remain valid for UACAP because equivalent messages are used. In principle, the same argument as for the first commitment (described above in the **TL** case) holds:

An attacker would need to guess the short inputs before they are revealed to successfully masquerade as A or B when the long nonces J_a and J_b are revealed for comparison of Q_a and Q_b. By adding these nonces to the exchanged hashes, a brute-force guessing game has the order of 2^{128} and only a one-off chance for guessing R_a and R_b (equal for the IC case) remains when the attacker fabricates own nonces as an active MITM.

A. Wong-Stajano attack prevention

The Wong-Stajano attack on MANA III assumes that an attacker, after running a standard MITM attack on the DH key exchange, can find a collision for the originally proposed verification function $\text{Trunc-Hash}_{K_1}(I_a \| X \| Y \| R)$ respectively $\text{Trunc-Hash}_{K_2}(I_b \| X \| Y \| R)$ so that the random values K_1 and K_2 (used as key for the keyed Trunc-Hash function), which the

[2]The actual function used in the protocol can also be regarded as a two-key MAC.

attacker can choose freely, mask the differences in DH keys and lead to the same verification codes [25]. When those verification functions are implemented using cryptographic hashes, e.g. HMAC keyed by X, then this attack translates to finding an l-Bit collision in the hash function, which can always (even assuming perfect hash functions) be performed as a brute-force search in $O(2^l)$ (which is a much simpler attack than the one-off chance we would like to remain as the only online attack vector). When transmitting the full hash output (as defined in MANA III), this seems infeasible. However this attack does not work on UACAP by inheriting the principal security properties of MA-DH, which have been improved over the original MANA III proposal that allowed for the Wong-Stajano attack. A man-in-the-middle C would, in the general UACAP protocol run – independently of how the auxiliary message is transmitted –, perform the steps listed in Figure 3. The adversary function A must then generate an x' so that the truncated MAC suffers from a collision, i.e. $\text{Trunc-Hash}_{X\|Y'}(I_a\|I_b) = \text{Trunc-Hash}_{X'\|Y}(I_a\|I_b)$. At this time, the adversary has access to the components I_a, I_b, X', Y, and attempts to generate Y' so that the collision occurs. However, X has not yet been made public, and under the assumption that ephemeral DH keys are used and thus X is random, this translates to a guessing game with a one-off chance of 2^{-l} of succeeding. The difference between MANA III (and the subsequently proposed Wong-Stajano variant) and MA-DH (and MANA IV) is the initial commitment message, which prevents this attack. We point out that the above argument is not valid in the case where both a static DH key X and static identities (such as Bluetooth/Ethernet MAC or long-lived IP addresses) are used. In that case, a collision of the truncated MAC may be found in a bruteforce manner after having eavesdropped on an authentication. We therefore recommend to always use ephemeral DH keys (at least on the initiator side A) for each instance of running the protocol, as the identities are more likely to remain static. The only disadvantage is higher computational cost, especially for mobile devices.

An interesting case arises when auxiliary channels are one-way, discussed in more detail in [28], [25], and [11]. Authentication can not be fully mutual in the sense of human verifiability, but when a remote device can be trusted to correctly perform comparison of received auxiliary messages and report results of this comparison, then mutual authentication on the protocol level is possible. From a usability point of view, one-way human verification with implicit mutual authentication is the more likely variant, considering the any human involvement is costly in terms of scalability of human attention.

V. OpenUAT

OpenUAT aims to be an open-source, ready-to-use toolkit for authentication in ubiquitous computing applications [17]. The methods and protocols it implements are selected and designed to be intuitive and usable for the end user, and the overall toolkit aims to be modular and compact for the developer. Most parts are implemented in Java and verified to work on most Java virtual machines (JVMs) including Java 2 Micro Edition (J2ME) as available on many off-the-shelf mobile devices and the newer Android platform. However, protocol implementations use ASCII commands whenever possible to ease interoperability with other platforms (as specified below in Section V-A). A central design pattern is to use asynchronous, background processing and event notification. This has the advantage that applications and their user interfaces are not blocked by potentially lengthy protocol runs and that events provide a general "hooking" mechanism to react to various stages of authentication. All core parts are documented using Javadoc and covered by JUnit tests to ensure API stability during code changes. In the following, we describe the main components, pointing out how the UACAP specification is currently implemented.

A. UACAP Reference Implementation

In the current reference implementation of UACAP in OpenUAT, the following protocol commands are assumed to be executed in order. A client (or *initiator*) A connects to the server (or *responder*) B to start a protocol run. Then, following common Internet protocols, the server starts by sending its greeting:

1) "`HELO OpenUAT Authentication`", sent by B, indicates that the client may start its authentication request, i.e. that the (insecure, in-band) channel the client has connected to is connected to an OpenUAT instance.
2) "`AUTHREQ UACAP-1.0` I_a C_a `[PARAM` P`...]`" (M_1) is sent from A to B to transmit the client identity (e.g. its IP or Bluetooth MAC address or a nonce acting as an ephemeral identifier for a single interaction), its commitment and an optional parameter, which may be free-form and of arbitrary length. Note that the client requests key exchange and verification using a named protocol (currently `UACAP-1.0`) for interoperability with future versions.
3) "`AUTHACK` I_b Y" (M_2) is sent from B to A to respond with the server identity and the (ephemeral) server public DH key part.
4) "`AUTHACK2` X" (M_3) is sent from A to B to finish the DH key agreement with the client public key part.

All values are encoded depending on the chosen (in-band) RF channel. If it supports 8-bit data transmission (such as TCP sockets or Blueooth RFCOMM), the values are transmitted efficiently as byte arrays in standard network byte order (i.e. big-endian) with a prepended single byte specifying the array length. If the channel only supports character (e.g. ASCII) transmission (such as Jabber chat-messages as transport medium), all values are hex-encoded and sent as ASCII strings with space as delimiter.

The following protocol messages are dependent on the chosen authentication option. For the *input* case:

1) "`AUTHINPCOM` $O_{a/b}$" (M_5) provides the second commitment and is symmetrically sent by both A and B. After correctly receiving these messages, both devices should *exactly at this stage* (not earlier and not later) query for the user input that was previously provided to the other side.
2) "`AUTHINPOPEN` $J_{a/b}$" (M_6) is also sent symmetrically by both A and B to open the previous commitment and allow comparison of the user inputs.

For *transfer* and *verification*, other messages are transmitted over auxiliary channels and therefore with different format. The encoding and specific protocol message depends on the respective channel.

B. Auxiliary Channels Implementation

Currently, OpenUAT implements several auxiliary channels:

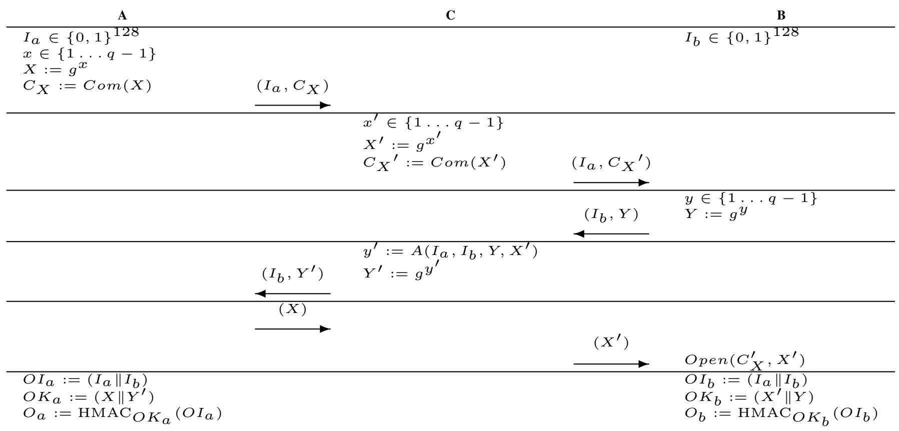


Fig. 3
WONG-STAJANO ATTACK ON MANA III IS PREVENTED IN UACAP

TABLE I
CURRENTLY IMPLEMENTED AUTHENTICATION METHODS

Channel/ Mode	Input	Transfer	Verify
Visual	-	Barcode transfer	Compare sentences Manual string comparison
Audio		Audio transfer	Compare melodies Ultrasound
Motion	Shaking		Shaking
Keypad	BEDA Manual keypad entry		

- **2D barcodes** [3], [28] are used to display the out-of-band messages on any screen and capture them with a mobile phone camera. QR codes are generated with an adapted embedded implementation, while we make use of the Google ZXing library [3] for decoding. The QR code includes at least 7 Bytes of O_a or O_b in hexadecimal notation, but can be scaled to transmit the full values in binary form when sufficient resolution in the "receiver" camera can be assumed for decoding the larger versions of QR codes.
- **Audio** as used in HAPADEP [5] is integrated into the OpenUAT code base. It was initially developed only for desktop systems, i.e. for Java 2 Standard Edition (J2SE) but subsequently ported to J2ME/MIDP and restructured to integrate with UACAP. In transfer mode, O_a or O_b is encoded with a "fast" codec and played as a wave file. The other device records the sound, decodes it, and verifies that $O_a = O_b$. In verify mode, applying the "slow" codec results in a piano-like melody, which should be more pleasant and easy to recognise by users.
- **Ultrasound** is used for a verification of *Spatial References* using the Interlock* protocol both on RF and ultrasonic channels [10].
- **Manual keypad entry** is used for "short" (confidential or non-confidential) input of R_a and R_b.
- **Manual string comparison** uses MADLib to create non-sensical sentences from the short messages O_a or O_b which are then compared by the user [4].
- **Synchronised button presses** are used in an implementation of BEDA [8] as another form of common input.
- **Motion** is used to detect when two devices are shaken together, either in verification mode by exchanging accelerometer time series with an Interlock* protocol or in input mode by creating keys directly out of sensor time series [6], [7]. Accelerometer data acquisition has been implemented for Spark Fun Electronics WiTilt sensors over Bluetooth, some on-mainboard sensors (e.g. in Thinkpad and Macbook laptops), for Symbian S60 (through a Python module which communicates with the Java MIDlet via a TCP socket), Android, and some Windows Mobile phones (through a native C# implementation that communicates over TCP using the same format).

Full source code is available under the terms of the GNU LGPL at `http://openuat.org`.

C. User Study

We used OpenUAT to conduct a first experiment towards comparing the usability of different authentication methods. For this user study, we selected four different authentication methods: using the *audio* and *visual* channels in *transfer* and *verification* modes. As a result, in transfer cases, the devices automatically decide whether authentication was successful, while in the verify mode, this responsibility lies with the user. We call the device initiating the communication C (client) and the device responding S (server).

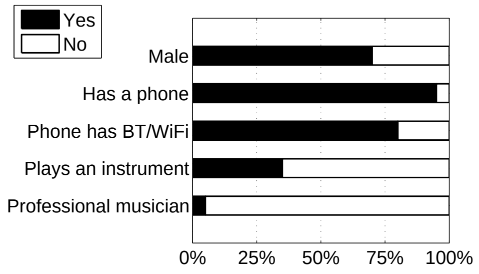
User Study Setup: From the user perspective, the verification step following automatic key agreement (step 4 in UACAP) proceeds as follows for the four different authentication methods. For **Barcode transfer**, S displays a 2D barcode. The user takes a picture of the barcode using C. C confirms whether the key was correctly exchanged. In **Audio transfer**, S plays a (fast) tune. C records the tune and confirms whether the key was correctly exchanged. To **Compare melodies**, C plays a melody. Subsequently, S plays a melody. The user compares the melodies and, if they match, acknowledges that the pairing was successful. Finally, to **Compare sentences**, both devices concurrently display a sentence. The user compares these sentences and, if they match, acknowledges that the pairing was successful.

We recruited 20 participants for our user study, mainly a group of fairly young, well-educated and technology-savvy participants

[3]ZXing http://code.google.com/p/zxing/

(19 university students and researchers and one secretary). The demographics and related background information of the participants are summarised in Table II. Due to the obvious bias concerning educational background and focus on technology in our group of participants, absolute numbers on estimated usability and completion time will likely not be representative of the average population. However, we argue that the more interesting findings, namely the comparison of authentication methods in relation to each other, are still valid and applicable to a larger population.

TABLE II
PARTICIPANT PROFILE



Age	18-24	10 %
	25-29	60 %
	30-34	15 %
	35-40	10 %
	40 +	5 %
Education	Bachelor	30 %
	Masters	65 %
	PhD	5 %

Test Procedure: All experiments were conducted in normal office conditions, namely good lighting conditions and relatively low noise level. Before starting, participants were asked to fill in the background questionnaire, which served to learn about their experience with mobile devices and music related background and is summarised in Table II. Afterwards, participants were given a brief overview of the problem. We motivated the need for secure device pairing and briefly presented the goals of our study. Although most of the users were already familiar with the functionality of the devices used, we gave a brief overview of the basic operations needed to interact with the device (e.g. how to navigate through the application menu, how to select options and to take pictures).

We have chosen Scenario 1 and 2 described in section I, namely exchanging vCards and printing a document, to motivate the participants to perform the tasks. To capture the QR code when pairing two phones (a Nokia N82 and a Nokia N95 8GB), a focus lens was attached to the camera of the phone, compensating for the lack of focus control in J2ME and consequently in the ZXing QR decoder. The printer was simulated by a laptop. For each scenario, participants were asked to run the pairing application with all four authentication methods. No attack was simulated, i.e. verification sequences always matched. To reduce the learning bias on test results, half of the users were first presented with Scenario 1 and the other half with Scenario 2. User actions and their durations were automatically logged. Afterwards, each participant filled in a post-test questionnaire form and was given some minutes of free discussion.

Results: Table III summarises the logged data with average completion times between 13.2 seconds (comparing sentences) and 37.7 s (barcode transfer from a phone display) and average number of tries until successful pairing between 1 (comparing sentences) and 2.3 (barcode transfer). Figure 4 presents user's perceived ease of use and level of security, split further between short-lived and long-lived keys, while Fig. 5 presents preference ratings. Especially our direct comparison of four authentication methods produced some interesting findings:

TABLE III
SUMMARY OF THE LOGGED DATA FOR PAIRING 1) TWO PHONES(PH-PH) AND 2) PHONE AND LAPTOP (PH-LAP)

Method	Average completion time (sec.)		Average number of tries		Percentage of failures	
	Ph-Ph	Ph-Lap	Ph-Ph	Ph-Lap	Ph-Ph	Ph-Lap
Barcode transfer	37.7 *(sd*=14.0)*	34.1 *(sd=15.3)*	2.1	2.3	5%	15%
Audio transfer	30.7*(sd=12.6)*	31.9 *(sd=14.2)*	1.2	1.6	0%	5%
Compare melodies	19.6 *(sd=8.3)*	20.3 *(sd=10.8)*	1.3	1.3	20%	10%
Compare sentences	15.6 *(sd=7.8)*	13.2 *(sd=4.5)*	1.0	1.0	0%	0%

*sd = Estimated standard deviation

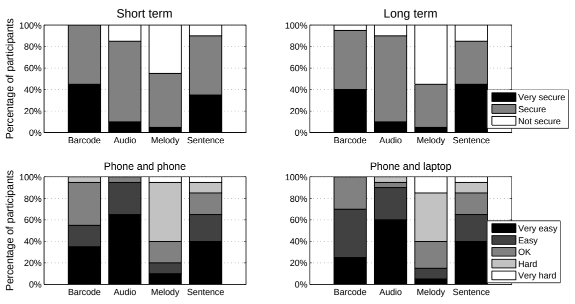


Fig. 4
USABILITY AND SECURITY ESTIMATION BY USERS

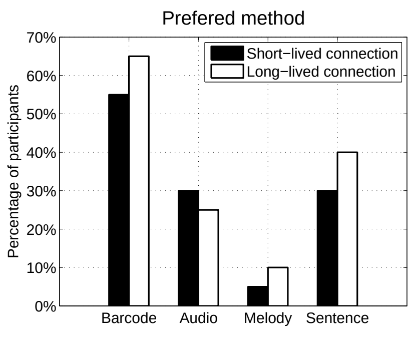


Fig. 5
USER PREFERENCE

Barcode transfer Results show that, even if using the visual channel in transfer mode resulted in the longest completion time, it was by far the most preferred and most highly trusted pairing method. In fact, users showed high acceptance even when decoding the QR code failed and they had to retry. After filling in the post-test questionnaire, participants were presented with the live QR decoder application that is pre-installed on some Nokia devices (but which could not be integrated with OpenUAT due do unavailability of open APIs). Seeing how smooth and fast decoding could be made, this method appealed even more to users. These results are in accordance with the background questionnaire which revealed that taking pictures is the most widely performed task on mobile phones (by over 80% of the participants).

Audio transfer Using the audio channel in transfer mode is the only authentication method in which the user did not have to assist the devices in any way. However, while overall study results acknowledged the method as the easiest to perform (see Figure 4), other factors such as the social context made users prefer the less obtrusive, seemingly more secure visual channel. Several users suggested replacing audio by ultrasound transfer (already implemented in OpenUAT, but inherently requiring additional hardware and visualisation). Another interesting result is the perceived duration time of the pairing process. Even though the method using the audio channel takes less time to complete – in average 15 s less than the visual channel – some users were bothered by the long time needed to decode the audio message (during which they did not have to conduct any task). On average, decoding the audio message took 16 s for phone to phone and 13 s for laptop to phone (due to better sound quality on the laptop).

Comparing melodies This is the method that users found the most difficult and which they least trusted. 35% of the participants have been playing at least one instrument for 3 to 27 years, with an average of 12 years. Unexpectedly, people with advanced music experience did not find the melodies easier to compare than people that do not play any instrument. On the contrary, these people were more sensitive to sound differences between devices (even the same tune will sound different when played by different devices) and trusted this authentication method less. On average, the tunes were replayed 1.3 times. Although there was no attacker, in 10% and 20% of the trials, respectively, participants failed to recognise the tune as being the same. The general impression was that the sequences were too long. A melody lasted 4 to 5 s and played 7 to 9 notes.

Comparing sentences This was generally considered a secure method. It had the fastest completion time (13 and 15 s), but required significant user attention. Some participants were bothered by the lack of semantics of the sentences (e.g. *"DURWARD FOOLHARDILY DISTORT-ed to BRANCH on a COMMITTEE"*). Because sentences are automatically constructed from a cryptographic token, they do not have any meaning.

VI. Conclusions

The problem of authenticating spontaneous interactions has seen significant interest in recent years due to its wide applicability in current and future application scenarios. Many approaches have been suggested independently, and only rarely distributed with an open, reproducible implementation. In this paper, we contribute UACAP as a unified cryptographic protocol for device authentication to use with arbitrary auxiliary channels. We also contribute OpenUAT as an open source, publicly available toolkit for authentication and implement some intuitive authentication methods in a common library based on UACAP. Video, audio, ultrasound, motion, keypad input, and sentence comparison are already available for application developers and are easily comparable and interchangeable. Further auxiliary channel implementations are currently being integrated.

It seems important to provide a vast library of different methods — they should be chosen to best suit the envisaged application, and direct comparability in rapid prototyping will assist application designers in doing so. All of our suggested application scenarios can already be supported by UACAP and the existing implementations in OpenUAT, and we speculate that many future mobile applications will also fit within these categories and therefore immediately benefit from the presented work. By providing OpenUAT as a toolkit for system builders (available at `http://openuat.org`), we hope to both foster future research and to shorten the gap between research prototypes and real-world applications.

References

[1] R. Mayrhofer, “Ubiquitous computing security: Authenticating spontaneous interactions,” September 2008, habilitation thesis, University of Vienna.

[2] R. Mayrhofer and I. Ion, “OpenUAT: The open source ubiquitous authentication toolkit,” ETH Zurich, Tech. Rep. 657, August 2010.

[3] J. M. McCune, A. Perrig, and M. K. Reiter, “Seeing-is-believing: Using camera phones for human-verifiable authentication,” in *Proc. IEEE Symp. on Security and Privacy*. IEEE CS Press, 2005, pp. 110–124.

[4] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun, “Loud and clear: Human verifiable authentication based on audio,” in *Proc. ICDCS 2006*. IEEE CS Press, July 2006, p. 10.

[5] C. Soriente, G. Tsudik, and E. Uzun, “HAPADEP: Human asisted pure audio device pairing,” Cryptology ePrint Archive, Report 2007/093, March 2007.

[6] R. Mayrhofer and H. Gellersen, “Shake well before use: Authentication based on accelerometer data,” in *Proc. Pervasive 2007: 5th International Conference on Pervasive Computing*, ser. LNCS, vol. 4480. Springer-Verlag, May 2007, pp. 144–161.

[7] ——, “Shake well before use: Intuitive and secure pairing of mobile devices,” *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 792–806, June 2009.

[8] C. Soriente, G. Tsudik, and E. Uzun, “BEDA: Button-enabled device pairing,” in *Proc. IWSSI 2007*, September 2007, pp. 443–449.

[9] A. Varshavsky, A. Scannell, A. LaMarca, and E. de Lara, “Amigo: Proximity-based authentication of mobile devices,” in *Proc. UbiComp 2007*. Springer-Verlag, September 2007, pp. 253–270.

[10] R. Mayrhofer, H. Gellersen, and M. Hazas, “Security by spatial reference: Using relative positioning to authenticate devices for spontaneous interaction,” in *Proc. Ubicomp 2007: 9th International Conference on Ubiquitous Computing*, ser. LNCS, vol. 4717. Springer-Verlag, September 2007, pp. 199–216.

[11] R. Mayrhofer and M. Welch, “A human-verifiable authentication protocol using visible laser light,” in *Proc. ARES 2007: 2nd International Conference on Availability, Reliability and Security*. IEEE CS Press, April 2007, pp. 1143–1147.

[12] S. N. Patel, J. S. Pierce, and G. D. Abowd, “A gesture-based authentication scheme for untrusted public terminals,” in *Proc. UIST 2004*. ACM Press, October 2004, pp. 157–160.

[13] R. Mayrhofer, “The candidate key protocol for generating secret shared keys from similar sensor data streams,” in *Proc. ESAS 2007: 4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, ser. LNCS, vol. 4572. Springer-Verlag, July 2007, pp. 1–15.

[14] A. J. Nicholson, I. E. Smith, J. Hughes, and B. D. Noble, “LoKey: Leveraging the SMS network in decentralized, end-to-end trust establishment,” in *Proc. Pervasive 2006*. Springer-Verlag, May 2006, pp. 202–219.

[15] I. Buhan, J. Doumen, P. Hartel, and R. Veldhuis, “Secure ad-hoc pairing with biometric: SAfE,” in *Proc. IWSSI 2007*, September 2007, pp. 450–456.

[16] I. Ion, M. Langheinrich, P. Kumaraguru, and S. Čapkun, “Influence of user perception, security needs, and social factors on device pairing method choices,” in *Proc. SOUPS 2010*. New York, NY, USA: ACM, July 2010, pp. 6:1–6:13. [Online]. Available: http://doi.acm.org/10.1145/1837110.1837118

[17] R. Mayrhofer, “Towards an open source toolkit for ubiquitous device authentication,” in *Workshops Proc. PerCom 2007: 5th IEEE International Conference on Pervasive Computing and Communications*. IEEE CS Press, March 2007, pp. 247–252, track PerSec 2007: 4th IEEE International Workshop on Pervasive Computing and Communication Security.

[18] C. Gehrmann, C. J. Mitchell, and K. Nyberg, “Manual authentication for wireless devices,” *RSA Cryptobytes*, vol. 7, no. 1, pp. 29–37, 2004.

[19] S. Laur and K. Nyberg, "Efficient mutual data authentication using manually authenticated strings," in *Proc. CANS 2006*. Springer-Verlag, December 2006, pp. 90–107.
[20] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *Proc. NDSS'02*. The Internet Society, February 2002.
[21] J.-H. Hoepman, "The emphemeral pairing problem," in *Proc. 8th Int. Conf. Financial Cryptography*. Springer-Verlag, February 2004, pp. 212–226.
[22] S. Vaudenay, "Secure communications over insecure channels based on short authenticated strings," in *Proc. CRYPTO 2005*. Springer-Verlag, August 2005.
[23] S. Creese, M. Goldsmith, R. Harrison, B. Roscoe, P. Whittaker, and I. Zakiuddin, "Exploiting empirical engagement in authenticated protocol design," in *Proc. SPC 2005*. Springer-Verlag, April 2005, pp. 119–133.
[24] M. Čagalj, S. Čapkun, and J.-P. Hubaux, "Key agreement in peer-to-peer wireless networks," *IEEE (Special Issue on Cryptography and Security)*, vol. 94, pp. 467–478, 2006.
[25] F.-L. Wong and F. Stajano, "Multi-channel protocols," in *Proc. Security Protocols Workshop 2005*. Springer-Verlag, 2006.
[26] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Proc. 7th Int. Workshop on Security Protocols*. Springer-Verlag, April 1999, pp. 172–194.
[27] T. Kindberg and K. Zhang, "Context authentication using constrained channels," HP Laboratories, Tech. Rep. HPL-2001-84, April 2001. [Online]. Available: http://www.hpl.hp.com/techreports/2001/HPL-2001-84.pdf
[28] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel," Cryptology ePrint Archive, Report 2006/050, 2006.
[29] S. Sigg and D. Schuermann, "Secure communication based on ambient audio," *IEEE Transactions on Mobile Computing (TMC)*, 2012, accepted for publication.
[30] Bluetooth SIG, "Bluetooth Special Interest Group. Simple Pairing Whitepaper (Revision V10r00)," 2006.
[31] Wi-Fi Aliance, "Wi-Fi protected setup specification v1.0," January 2007.
[32] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.
[33] A. Perrig and D. Song, "Hash visualization: a new technique to improve real-world security," in *Proc. CrypTEC'99*, 1999, pp. 131–138.
[34] N. Ferguson and B. Schneier, *Practical Cryptography*. Wiley Publishing, 2003.
[35] P. J. Bond, A. L. Bement, and W. Mehuron, "Fips pub 198 federal information processing standards publication the keyed-hash message authentication code (hmac)," Tech. Rep., 2002.
[36] P. Gallagher, D. D. Foreword, and C. F. Director, "Fips pub 186-3 federal information processing standards publication digital signature standard (dss)," 2009.

Rene Mayrhofer currently holds a full professorship for Mobile Computing at Upper Austria University of Applied Sciences, Campus Hagenberg, Austria. Previously, he held a guest professorship for Mobile Computing at University of Vienna, Austria, during which he received his venia docendi for Applied Computer Science. His research interests include computer security, ubiquitous computing, and machine learning, which he brings together in his research on intuitive and unobtrusive techniques for securing spontaneous interaction. He received his Dipl.-Ing. (MSc) and Dr. techn. (PhD) degrees from Johannes Kepler University Linz, Austria, and subsequently held a Marie Curie Fellowship at Lancaster University, UK.

Jürgen Fuß is full professor for Applied Mathematics and Cryptography at the Department of Secure Information Systems at Upper Austria University of Applied Sciences, Campus Hagenberg, Austria. His research interests include algebra, cryptography, computer security, and high-performance computing. He studied Mathematics in Linz, Austria, and Grenoble, France, and received his Dipl.-Ing. (MSc) and Dr. techn. (PhD) degrees from Johannes Kepler University Linz, Austria.

Iulia Ion is currently a PhD candidate at ETH Zurich, in the Institute for Pervasive computing. Her research interests lie in creating usable security systems and tools for users to protect their privacy in the cloud. She received her Masters and Bachelors in Computer Science from International University in Germany.