Security Associations for Personal Devices

N. Asokan[1,2] and Kaisa Nyberg[1,2]

[1] Nokia Research Center
[2] Helsinki University of Technology
{N.Asokan, Kaisa.Nyberg}@nokia.com

Abstract. Introducing a new device to a network or to another device is one of the most security critical phases of communication in personal networks. It is particularly challenging to make this process of *associating* devices easy-to-use, secure and inexpensive at the same time. A cornerstone of this process is key establishment. There has been a number of research proposals for key establishment in personal networks. Some of them have been adapted by emerging standard specifications. In this paper, we first present a taxonomy of protocols for key establishment in personal networks as well as describe and analyze specific protocols. We then use this taxonomy in surveying and comparing association models proposed in several emerging standards from security, usability and implementability perspectives.

Keywords: Personal networks, security association, key agreement, standards.

1 Introduction

Short-range communication standards have brought a large number of new services to the reach of common users. For instance, standards for personal networking technologies such as Bluetooth[3], Wi-Fi[4], Wireless Universal Serial Bus (WUSB)[5], and HomePlugAV[6] enable users to easily introduce, access, and control services and devices both in home and mobile environments.

The initial process of introducing a new device to another device or to a network is called an *association*. Association consists of the participating devices finding each other, and possibly setting up a *security association*, such as establishing a shared secret key, between them.

The part of the association procedure that is visible to the user is called an *association model*. Association models in today's personal networks such as those based on Wi-Fi or Bluetooth, typically consist of the user scanning the neighborhood from one device, selecting the other device or network to associate with, and then typing in a shared passkey. These current association procedures

[3] http://bluetooth.org
[4] http://wi-fi.org
[5] http://usb.org/wusb
[6] http://homeplug.org

have several usability and security drawbacks arising primarily from the fact that they are used by ordinary non-expert users. First, when there are many devices or networks in the scanned neighborhood, users find it difficult to choose the correct one from a, possibly long, list of choices. Second, the security of the association protocol depends on the strength of the shared passkey. Making the passkey long and hard-to-guess impacts usability. Using a short or memorable passkey leaves the protocol vulnerable to dictionary attacks, even by passive eavesdroppers. Also, over the last few years several other cryptographic weaknesses have also been discovered in the association protocols used in Wi-Fi and Bluetooth.

To address these concerns, various new ideas have been proposed with the intent of providing a secure yet usable association model. For instance, there have been proposals for key establishment schemes utilizing short passwords/checksums [48, 18, 11, 19, 42, 44] or various types of out-of-band channels [37, 1, 24, 34, 36]. In reality, it is impractical to mandate a single association model for all kinds of devices because different devices have different hardware capabilities. Also, different users and application contexts have different usability and security requirements. Because of this, forthcoming standards are adopting multiple association models. Although low-end devices like headsets and wireless access points may be limited to one association model, richer devices like mobile phones and personal computers will naturally support several. The security of individual association models has been studied widely. But new kinds of threats may emerge when several models are supported in personal devices and several standards, both new and old, are in use simultaneously.

In this paper, we present and analyze various protocols for key establishment in personal networks and present a taxonomy for classifying them. We then make a comparative analysis of association models proposed in different standards from a practical point of view. The surveyed standards are Bluetooth Secure Simple Pairing [35], Wi-Fi Protected Setup [46], Wireless USB Association Models [47], and HomePlugAV security modes [28]. We show the similarities between the protocols in different standard specifications by relating them to our taxonomy. We point out other similarities as well: All of the them can address the problem of finding the right peer device usually by supporting some variation of the notion of *user-conditioning*: a device participates in the association only when it is in a special association mode; typically a device enters the association mode in response to an explicit user action, such as pressing a button. All of the surveyed standards are targeted for personal networks and support multiple association models.

The rest of this chapter is organized as follows. In Section 2 we provide a systematic taxonomy of different protocols for key establishment and describe some basic protocols. In Section 3 we discuss formal analysis of the security of some of these protocols. In Section 4 we look at how different types of secure channels and physical interfaces can be used to implement the protocols discussed in Section 2. In Section 5 we explain how and which key establishment protocols and related association models are used in the surveyed standards. In Section 6 we evaluate and analyse the various association models described in

these standards. Finally, in Section 7 we summarize the chapter and contemplate possible future developments in this area.

2 Key Establishment Protocols

2.1 Classification of Key Establishment Methods

All of the association models we will survey in Section 5 are based on one or more protocols for human-mediated establishment of a shared key between two devices. The shared key is typically used to protect subsequent communication over the otherwise insecure communication channel and, possibly, in authentication for other access control decisions. We show that the same basic protocols are used in different standard specifications, even though the exact instantiations naturally differ.

As a prelude to identifying and comparing these different instantiations, we present a systematic classification of human-mediated key establishment protocols that can be used in personal networks. Figure 1 provides an overview of this classification.

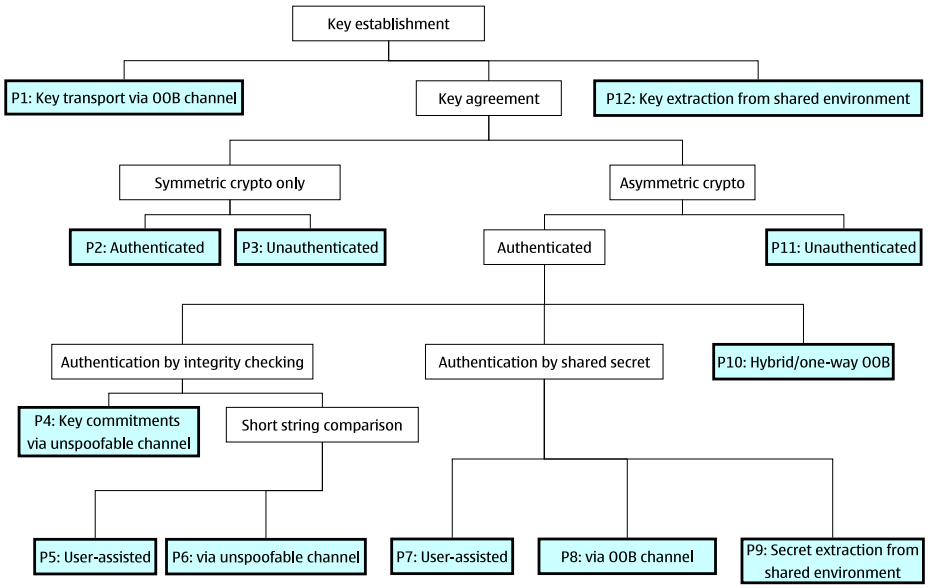

Fig. 1. Classification of Key Establishment Methods for Personal Networks

The attacker model for key establishment is as follows. The two devices involved in key establishment are capable of communicating over an insecure communication channel. The devices themselves are assumed to be secure and trustworthy. The attacker has the standard Dolev-Yao capabilities [8] over the

insecure channel: he can insert, delete, modify or delay messages sent over the insecure channel. The security objective of the participating devices is to establish a common key shared only between the two devices, which they can use to protect subsequent communication between them. The goal of the attacker is to intervene in this process so that either it can read subsquent communication between the participating devices, or act as an active man-in-the-middle. In the latter case, the attacker can generate or modify messages and fool one or both of the devices into accepting these messages as originating from the peer device.

At a high level, key establishment may be a simple *key transport* or involve running a *key agreement* protocol. In the context of personal networks where the devices are likely to be in close proximity, an additional key establishment method is *key extraction* from the common shared environment.

Key transport: In key transport, one device chooses the key and transmits it directly to the second device using an out-of-band secure communication channel (**P1**). Typical out-of-band channels used for key transport include a direct USB cable connection or the use of removable memory, like flash drives. The security of key transport depends on the out-of-band channel being secret and unspoofable: a man-in-the-middle must not be able to modify the data transmitted between the devices.

Key extraction: Devices in personal networks are in close proximity to one another and thus share a common ambient environment. This gives rise to an interesting third possibility for key establishment: measurements of certain environmental parameters, such as the signal strengths of radio beacons in the vicinity [41] or ambient noise, may be similar in devices that are close to each other but hard to predict from devices that are not in the same place at the same time. By measuring such parameters, and using them in a key agreement protocol, the devices may be able to *extract an authenticated shared secret* (**P12**).

Key Agreement: Key agreement protocols may be based purely on symmetric key cryptography, or may be based on asymmetric key cryptography as well. In the latter case, the typical protocol is Diffie-Hellman key exchange [7].

Key agreement may be *unauthenticated* or *authenticated*. Unauthenticated symmetric key agreement (**P3**) is vulnerable even to passive eavesdroppers. Unauthenticated asymmetric key agreement (**P11**) is secure against passive eavesdroppers but is vulnerable to active man-in-the-middle.

2.2 Authentication Methods

There are a number of ways to authenticate key agreement. Key agreement based on symmetric key cryptography is authenticated by using a sufficiently long *pre-shared secret* (**P2**). The security of such protocols depend on the length of the pre-shared secret. Authentication of asymmetric key agreement can be performed using some form of *integrity checking*, or by using a pre-shared secret or using a combination of these two. Authentication by integrity-checking can be done either by exchanging and comparing commitments to public keys, or by exchanging and comparing short integrity checksums. Now we take a closer look at the protocols involved in each case.

Authentication by exchanging key commitments: A simple folklore protocol to authenticate the public keys of two devices is to use an auxiliary channel to exchange commitments to the public keys (**P4**) [1]. The auxiliary channel is unspoofable in that it is difficult for an attacker to insert, modify or delete messages in the channel without being detected. When the devices exchange public keys via the in-band channel, they can validate the authenticity of these keys by using the information exchanged via the auxiliary channel.

The security of the protocols depends on the auxiliary channel being unspoofable. There are two ways to realize such auxiliary channel. The first is to use a separate, out-of-band, physical channel which is resistant to spoofing. Several such out-of-band channels have been proposed in the literature including audio [15], visual [24, 34], infrared [1] and Near-Field Communication (NFC). Both devices involved in the association are assumed to support the same type of physical hardware interfaces. The second way is to use the *I-Codes* [43] technique which uses the anti-blocking property inherent in some otherwise insecure in-band channels to construct a logical auxiliary channel which is difficult to spoof.

The security also depends on the commitments of public keys being strong enough (e.g., a cryptographic hash function with at least 80 bits of output) to resist the attacker finding a second pre-image to the commitment.

Authentication by short integrity checksum: The idea of using short checksums to authenticate a key agreement was originally proposed by Zimmermann in PGPfone [48]. Subsequently several researchers have proposed variations and enhancements [42, 44, 19, 30]. In these protocols, each device computes a short checksum from the messages exchanged during the key agreement protocol. As we shall see in the example protocol below, the messages are structured such that if the two checksums are the same, the exchange is authenticated. This is sometimes referred to as "short authenticated string" (SAS) protocols. A basic three round mutual authentication protocol from [19] is depicted, in a simplified form, in Figure 2. Devices D_1 and D_2 first exchange their public keys PK_1 and PK_2. The protocol is used to mutually authenticate public keys. The notations are as follows: in practice, h is a cryptographic hash function like SHA-256; f is also a hash function, but with a short output mapped to a human-readable string of digits. The security requirements on the cryptographic primitives h and f will be discussed in Section 3. The hat '$\hat{}$' symbol is used to denote the receiver's view of a value sent in protocol message over the insecure in-band channel.

The check in the last step can be done in many different ways. One way is to ask the user to do the comparison (**P5**): Each device "shows" its own string to the user and ask whether it is the same as what the other device is showing. "Showing" can use any applicable user interface: displaying the string on a screen, or having a voice synthesizer read out the characters in the string. If the checksum strings are identical, the user indicates this to both devices and both devices conclude that the authentication is successful. Otherwise, the user indicates a mismatch to both devices and both conclude that the authentication did not succeed. An alternative way is to do the check using an auxiliary un-

1. D_1 generates a long random value R_1, computes commitment $h = h(R_1)$ and sends it to D_2
   $\mathsf{D}_1 \rightarrow \mathsf{D}_2$: h
2. D_2 generates a long random value R_2 and sends it to D_1
   $\mathsf{D}_1 \leftarrow \mathsf{D}_2$: R_2
3. D_1 sends R_1 to D_2
   $\mathsf{D}_1 \rightarrow \mathsf{D}_2$: R_1
4. D_2 checks if $\hat{h} \stackrel{?}{=} h(\hat{R}_1)$. If equality holds, D_2 computes $V_2 = f(P\hat{K}_1, PK_2, \hat{R}_1, R_2)$, otherwise it aborts.
   D_1 computes $V_1 = f(PK_1, P\hat{K}_2, R_1, \hat{R}_2)$.
5. Both devices check if V_1 equals V_2.

Fig. 2. Authentication by Short Integrity Checksum

spoofable channel (**P6**). As before, the unspoofable channel can be a physical out-of-band channel (as in [34, 36]) or an I-Codes channel (as in [43]).

To break this protocol, a man-in-the-middle has to choose random mumbers R'_1, R'_2 and public keys PK'_1, PK'_2 so that $f(PK'_1, PK_2, R'_1, R_2)$ equals $f(PK_1, PK'_2, R_1, R'_2)$. The security of the protocol depends on the quality of the functions h and g. If h is collision-resistant, attacker has to choose R'_1 without knowing anything about R_2. If h is one-way, attacker has to choose R'_2 without knowing about R_1. If the output of f is a uniformly distributed ℓ-bit value, then the chance of a man-in-the-middle succeeding is $2^{-\ell}$ because the attacker cannot influence the outcome of g. This success probability does not depend on any additional assumptions about the computational capabilities of the attacker beyond that he cannot break h in real time. In Section 3, we explain the cryptographic assumptions on h and f and give an overview of the formal proof presented in [20]. In Section 6 practical instantiations of this protocol in standards are evaluated.

Authentication by (short) shared secret: Key exchange can also be authenticated using a short pre-shared secret passkey. A number of different methods have been proposed for password-authenticated key exchange since Bellovin and Merrit introduced the idea in [4]. In Figure 3 we describe a variant of the MANA III protocol [11] originally described in [18]. It uses a one-time passkey P to authenticate PK_1 and PK_2. P is split into k pieces, labelled $P_1 \ldots P_k$. The steps in the protocol are repeated k times. The figure shows the exchanges in the i^{th} round.

In each round, each party demonstrates its knowledge of P_i. A man-in-the-middle can easily learn P_1 by sending garbage in message 2, and figuring out P_1 by exhaustive search once D_1 reveals R_1 in message 3. However, without knowing $P_i, i = 2 \ldots k$, the attacker cannot successfully complete the protocol run (recall that P is a *one-time* passkey). With ℓ-bit passkey and k rounds the probability for a successful man-in-the-middle attack is $2^{-(\ell - \frac{\ell}{k})}$. As in the case of short authentication string, the man-in-the-middle success probabilities

1. D_1 generates a long random value R_{i1}, computes commitment $h_{i1} = h(1, PK_1, P\hat{K}_2, P_i, R_{i1})$ and sends it to D_2
   $\mathsf{D}_1 \rightarrow \mathsf{D}_2$: h_{i1}
2. D_2 generates a long random value R_2, computes commitment $h_{i2} = h(2, PK_2, P\hat{K}_1, P_i, R_{i2})$ and sends it to D_1
   $\mathsf{D}_1 \leftarrow \mathsf{D}_2$: h_{i2}
3. D_1 responds by sending R_{i1} to D_2
   $\mathsf{D}_1 \rightarrow \mathsf{D}_2$: R_{i1}
   D_2 checks if $\hat{h}_{i1} \stackrel{?}{=} h(1, P\hat{K}_1, PK_2, P_i, \hat{R}_{i1})$ and aborts if it does not hold.
4. D_2 responds by sending R_{i2} to D_1
   $\mathsf{D}_1 \leftarrow \mathsf{D}_2$: R_{i2}
   D_1 checks if $\hat{h}_{i2} \stackrel{?}{=} h(2, PK_1, P\hat{K}_2, P_i, \hat{R}_{i2})$ and aborts if it does not hold.

Fig. 3. Round i of Authentication by (Short) Shared Secret

do not depend on additional assumptions about the attacker's computational capabilities.

There are three different ways for arranging for both devices to know the same P. One way is to have the user as the intermediary (**P7**): one device may show a value for P which the user is asked to enter into the second device, or the user may choose P and enter it into both devices. Alternatively, P may be transported from one device to another using a out-of-band channel providing communication secrecy (**P8**). A third possibility is to extract P from the shared environment (**P9**). In the latter two methods, there is no need for a human to transfer P between the devices. Consequently P can be longer, thus making probability for a successful attack smaller. Note that P is still used only to authenticate the key agreement, rather than as the long term secret.

Hybrid authentication: Hybrid authentication protocols are used to achieve mutual authentication when only a one-way out-band-channel is available (**P10**). The one-way channel is used to transmit the shared secret value and a hash of the public key from the first device to the second. The second device authenticates the first based on the public key hash. The first device authenticates the second based on its knowledge of the shared secret. A basic protocol is depicted in Figure 4. The function $c(M, K)$ is a message authentication code (MAC) on message M using a key K.

The security of the protocol depends on the out-of-band communication being both secret and unspoofable, as well as on strength of the hash function h and the message authentication code function c.

3 Data Authentication in the Two-Channel Model

3.1 The Two-Channel Model

Protocols for data authentication can be used as a building block for authenticating key agreement. In this section, we analyse the security principles of data

1. D_1 picks two long random values R_1 and K, computes commitment h to public key PK_1 as $h = h(PK_1, R_1)$ and sends h and K using OOB channel
   $\mathsf{D}_1 \Rightarrow \mathsf{D}_2$: S, C_1 (sent via the OOB channel)
2. D_1 sends its public key and random value using in-band channel.
   $\mathsf{D}_1 \rightarrow \mathsf{D}_2$: PK_1, R_1
3. D_2 checks if $h \stackrel{?}{=} h(\hat{PK}_1, \hat{R}_1)$ and aborts if it does not hold. Otherwise, D_2 picks its own long random value R_2, computes $C = c(\hat{PK}_1|PK_2|\hat{R}_1|R_2, K)$ and sends the result to D_1 with its own public key and random value.
   $\mathsf{D}_1 \leftarrow \mathsf{D}_2$: PK_2, R_2, C
4. D_1 checks if $\hat{C} \stackrel{?}{=} c(PK_1|\hat{PK}_2|R_1|\hat{R}_2, K)$ and aborts if it does not hold.

Fig. 4. Hybrid Authentication Protocol

authentication protocols that are deployed in the new association models for wireless standards and how they make use of secure short strings in data authentication protocols. In particular, we will investigate what is the minimum number of secure bits needed for the purposes of the data authentication protocol. As the number of bits sent over the secure channel is small, even doubling it would make a significant difference.

We start by presenting the basic concepts of data authentication protocols. A *unilateral data authentication protocol* has two parties, a sender with identity I and a receiver, whose identity is typically not specified. The purpose of the protocol is to allow the receiver to corroborate the sender's identity. The sender generates a message M. The input to the protocol is (I, M). At the end of the protocol, the receiver outputs the result, which is either (I, M) or no result. In the latter case the receiver has not been able to corroborate the sender's identity. In a *data cross-authentication protocol* two unilateral data authentication protocols, where the sender in one protocol is the receiver in the second one, and vice versa, have been integrated into one protocol. A cross-authentication protocol is called a *mutual data authentication protocol* if the sender's and receiver's data are equal.

An adversary of a data authentication protocol has one goal: forgery of the data. The adversary succeeds if for the sender's input (I, M) to the protocol the receiver outputs (I, M'), where $M' \neq M$. The adversary is allowed to try all possible strategies to achieve its goal. In particular, the adversary is allowed to control the synchronization of the protocol messages and run separate conversations with the sender and the receiver. However, the legitimate parties are supposed to act according to the protocol. In particular, the order of the messages sent by each entity is fixed, and each entity will wait until they receive a message from the previous round before going to the next round.

On the other hand, in the absence of an adversary, the authentication protocol must work correctly with overwhelming probability. In other words, the protocol must be complete.

Cryptographic protocols make use of two types of channels: a secure channel and an insecure channel. The secure channel is used for key management or some other type of protected communication. Based on this communication the data sent over the insecure channel can be secured by cryptographic means. The secure channel is subject to limitations that are typically measured in terms of the length of the secret key. Within the context of this chapter the limitations are particularly severe, as the number of bits that can be transmitted over the secure channel is much less than the typical length, say 128 or 256, of a cryptographic key.

The protocols we are now investigating try to achive the property that the short string transmitted over the secure channel is unknown to the adversary at the time when she must submit her forged message to the protocol. On the other hand, guessing these bits correctly is sufficient to succeed in the protocol. Hence the lower bound to the adversary's probability of success is $2^{-\ell}$ where ℓ is the number of secure bits.

To investigate the adversary's success probability we distinguish between unconditionally secure protocols and computationally secure protocols. We will see that the best protocols in the computational security model achieve the least achievable forgery probability, while in the unconditional security model twice as many bits must be used to achieve the same security level. Moreover, the recent research results show that going below the forgery probability ϵ with less than $2\log_2 \frac{1}{\epsilon}$ secure bits is possible if and only if one-way functions exist.

We start by investigating the security of data authentication protocols in the case where the secure channel provides secrecy.

3.2 Secure Channel with Secrecy

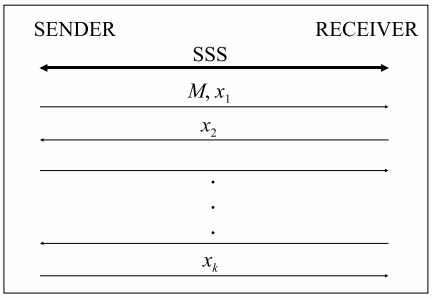

Fig. 5. SSS Authentication Protocol

Figure 5 depicts the model of a data authentication protocol that uses a short secret string (SSS). In the simplest protocol, only the first message (M, x_1) is sent. In this noninteractive case only unilateral authentication of the data from

the sender to the receiver can be achieved. If more messages are sent, then mutual or cross authentication of the data can be achieved.

The security of such a protocol can be considered in either unconditional or computational security model depending on the cryptographic primitives that are used by the protocol to compute the messages x_i.

Unconditionally Secure Data Authentication Using SSS. Unconditional security for data authentication protocols is measured in terms of the success probability a computationally unbounded attacker has when trying to forge a message. We say that a protocol using a shared secret of length ℓ is an unconditionally secure (n, ℓ, k, ϵ)-authentication protocol, if data of length upto n bits can be authentiacted by running k rounds of the protocol such that an attacker's probability of forging the data is at most ϵ.

Universal hash function families introduced by Carter and Wegman in [45] are the most important primitive in unconditionally secure authentication. They are also used in computationally secure protocols, see Section 3.3.

Definition 1. *An ϵ-almost universal (ϵ-AU) family of hash functions is a collection of functions $\{H_K\}$ from a message space $\mathcal{M}$ to strings of fixed length ℓ. The functions depend on a parameter K such that for any $x, y \in \mathcal{M}$, $x \neq y$ the probability that $H_K(x) = H_K(y)$ is at most ϵ.*

Hidden in this definition is the fact that ϵ is always at least equal to $2^{-\ell}$ and moreover that ϵ typically depends on the size of the message space, that is, on the length n of the message. A concrete example of ϵ-AU hash family is given next using the Reed-Solomon codes.

Suppose that the bit-length of the message M to be authenticated is n and that it is given as a string of ℓ-bit blocks. We denote $L = \lceil n/\ell \rceil$ and let M_i be the ith message block of M, $i = 1, 2, \ldots, L$. Then given a key K of length ℓ bits, an authentication tag of length ℓ is computed by evaluating a polynomial with coefficients M_i over the field $GF(2^\ell)$ at the point $K \in GF(2^\ell)$, that is,

$$H_K(M) = M_1 + M_2 K + \ldots + M_L K^{L-1}.$$

Here all operations are computed in the finite field $GF(2^\ell)$. Then it can be shown that $\{H_K\}$ is an ϵ-AU family of hash functions with $\epsilon = (L-1) \times 2^{-\ell}$.

If the constant term of the polynomial is not used, or equivalently, instead of $\{H_K\}$ we consider the family $\{\overline{H}_K\}$, where $\overline{H}_K = KH_K$, for messages of length ℓL. Then we have what is called an ϵ-almost xor-universal (ϵ-AXU) family of hash functions. It has the property that for any $x, y \in \mathcal{M}$ and for any δ of length ℓ bits, the probability that $\overline{H}_K(x) \oplus H_K(y) = \delta$ is less than or equal to ϵ.

Given an ϵ-AXU family of hash functions $\{\overline{H}_K\}$, and two keys K_1 and K_2, the Carter-Wegman MAC of message M is computed as $\overline{H}_{K_1}(M) \oplus K_2$. In the case when Reed-Solomon codes are used in the construction of $\{\overline{H}_K\}$ as explained above, this MAC is also known as Galois MAC or GMAC [25].

Message authentication codes are subject to two types of forgeries: impersonation attacks and substitution attacks. An impersonation attack succeeds if

an adversary finds a message and a valid MAC for it without using any previously computed messages and valid MACs. In substitution attacks the adversary exploits existing valid message-MAC pairs to forge a new MAC. A MAC construction based on an AXU hash family gives protection against both types of attacks. However, the hash family being just AU is sufficient to thwart impersonation attacks.

Given a key of length 2ℓ a message of length $L\ell$ can be authenticated using a Carter-Wegman MAC of length ℓ such that the forgery probability is less than or equal to $L2^{-\ell}$. In other words, a noninteractive protocol in which the sender appends such a MAC to a message to be authenticated is a $(L\ell, \ell, 1, L2^{-\ell})$-authentication protocol.

The MANA I protocol described in [11] is based on this principle, but used in such a way that also the MAC is sent over the secure channel. Hence, using MANA I the SSS consists of two parts of length ℓ each, a key and a MAC. As the MAC is sent over the secret channel it suffices to use a MAC construction based on an AU hash family. On the other hand, this means that the SSS in MANA I depends on the message. This can be avoided by transmitting 2ℓ randomly generated bits of SSS over the secret channel and use it to compute a Carter-Wegman MAC of the message to be authenticated.

Long Messages. The problem with unconditionally secure message authentication codes is that the length of the shared secret increases with the length of the message if the forgery probability should remain the same. In 1993, Gemmel and Naor [14] stated that the minimum length ℓ of the shared secret for any unconditionally secure and noninteractive message authentication protocol is

$$\log_2 n + \log_2 \frac{1}{\epsilon} \leq \ell \leq \log_2 n + 2\log_2 \frac{1}{\epsilon}.$$

Further, they showed that by making the protocol interactive and increasing the number of rounds as the length of the message grows, the upper bound to the forgery probability can be kept the same without essentially increasing the length of the SSS. In particular they showed that there exists an $(n, \log_2^{(k)}(n) + 2\log_2 \epsilon, k, \epsilon)$-authentication protocol, where $\log_2^{(k)}$ denotes the composition of k base-2 logarithm functions. The original protocol presented in [14] is secure only in the synchronous communication model as was pointed out by Gehrmann in [10], after which the authors presented an improved version which is secure also for asynchronous communication. This version is available form the authors [13].

The lower bound for the length of the SSS in unconditionally secure message authentication protocols remained an open question until 2006 when it was shown by Naor, et al., that to achieve forgery probability at most ϵ any unconditionally secure message authentication protocol requires at least $2\log\frac{1}{\epsilon}$ secret bits, and that going below this bound is possible only using one-way functions, that is, relying on computational security [27].

Computationally Secure Data Authentication Using SSS. Considering computational security means introducing time t as one of the parameters of the

protocol. As above n is the length of the message, k is the number of rounds, and ℓ is the number of bits in the SSS. Then a protocol is said to be a computationally secure $(n, \ell, k, \epsilon, t)$-authentication protocol in the SSS-model if for any polynomial time adversary and for any $c > 0$ the forgery probability is at most $\epsilon + t^{-c}$ for sufficiently large t. Here the parameters n, k and ℓ may also depend on the parameter t.

Without going into the details of computationally secure MAC codes we mention that, in the ideal case, and for large values of ℓ, they can be used to construct a noninteractive authentication protocol with ϵ close to the maximum of the values $2^{-\ell}$ and 2^{-m} where m is the length of the MAC. The MAC is appended to the messages and sent over the insecure channel. However, for small values of ℓ the forgery probability is equal to 1 as by using 2^{ℓ} time, given a valid pair of a message and a MAC, the attacker can find the value of SSS by doing exhaustive search.

If instead, the MAC computation is randomized, no attack is known which could compute the SSS given the MAC and the message until the randomizer is also given to the attacker. In this manner, a computationally secure MAC is often used as a practical instantiatiation of a hiding commitment scheme, which we will explain next.

A commitment scheme consists of three algorithms: SETUP, COMMIT and OPEN. The SETUP algorithm generates a public random parameter. The known techniques for provable security assume that the public parameter is a public key in some public key cryptosystem, for which there is a private key which nobody can use. In practise, the public random parameter is typically a description of a hash function. The COMMIT algorithm is non-deterministic. It takes two strings as input, a message M and a random string R. It produces a commit value C and a decommit value D. The third algorithm is the algorithm OPEN and it is deterministic. It takes C and D as inputs and yields M or error signal as output. The OPEN algorithm shall be complete in the sense that whenever $\text{COMMIT}(M, R) = (C, D)$, then $\text{OPEN}(C, D) = M$.

Computational security of a cryptographic scheme is considered with respect to the resources (typically time) an adversary has available to attack an scheme. Let τ be the time-bound of an adversary. The most important security property of a commitment scheme is that it is binding. As our purpose is to explain informally the security principles and assumptions used in the security proofs, we use the following illustrative definition, see also [42].

Definition 2. *A commitment scheme is (τ, ϵ)-binding, if any τ-bounded user, who after producing a message M and a commitment C on M, is given a randomly selected R, cannot produce a decommit value D such that* $\text{COMMIT}(M, R) = (C, D)$ *with a probability larger than $2^{-q} + \epsilon$, where q is the length of R in bits.*

In Section 3.3 we will see that sometimes a stronger binding property, non-malleability, is needed. It means that the above type of forgery is not possible even if the adversary has some control of the value of R, and can prepare for it when generating the message and the commitment. The commitments used in SSS protocols must also be hiding, which we will define next.

Definition 3. *A commitment scheme is (τ, ϵ)-hiding, if no τ-bounded adversary, given M and a commit value C computed using a randomly selected R of length q bits, cannot find R with a probability larger than $2^{-q} + \epsilon$ even it is allowed to choose the messages M for which the commitment C is computed.*

The randomizer R that was used explicitly in the above definitions is usually omitted from the notation of commitment schemes. In the sequel, we will write $\text{COMMIT}(M) = (C, D)$ and $\text{OPEN}(C, D) = M$.

In 1984 Rivest and Shamir were puzzled by the man-in-the-middle problem in a situation where two parties send messages to each other encrypted with the intended receiver's public key [33]. The messages are assumed to contain some redundancy, secret to the man-in-the-middle, using which the parties can recognize legitimate messages. Their solution was to "interlock" the encrypted messages by splitting the messages into two parts and send them interleaved. They also suggest an alternative approache to commit to the ciphertext before revealing it, by sending a "cryptographic checksum" of the ciphertext first. The solution requires synchronous communication, otherwise it is insecure as shown by Bellovin and Merrit in [3]. They also discuss the problem when the interlock protocol is used for encryption of passkeys and mention that D. Davies suggested to split the passkey into two parts. This is essentially the idea used in MANA III, see Section 2. But now, instead of protecting secrecy of passkeys in public key encryption against an wiretapper, the task is to authenticate public keys using a short passkey.

MANA III makes use of a commitment scheme based on a hash function. The parameter R is a long random number that typically acts also as a randomizer to the commitment scheme. The COMMIT algorithm is non-deterministic to prevent the exhaustive search for possible short M described above. Let h be a hash function that takes two inputs. To compute a commitment to message M we first generate a long random number R and set $C = h(M, R)$ and $D = (M, R)$. When opening the commitment (C, D) the value $h(M, R)$ is computed first. If this value is equal to C, then OPEN outputs M, and an error signal otherwise.

To analyze the security of MANA III, let us describe it again using the abstract notion of commitment scheme. Clearly, the use of MANA III is not limited to authentication of public keys but can be used for any data M known to the parties. As before, let k denote the number of rounds, and P_i, $i = 1, 2, \ldots, k$, the SSS split into k parts. We denote the parties by A and B. Then we denote by $M_{ij} = (i, M, P_i)$ the message at round i, where $j = 1$ if the sender is A, and $j = 2$, if the sender is B. The ith round of the generic MANA III is given in Figure 6. For simplicity, unlike in Section 2 we do not mark the received copies of the data items using the '^' symbol.

It is immediate to see that it is necessary that this commitment scheme is ϵ-hiding and ϵ-binding with a small ϵ, since otherwise the adversary has a significant advantage in breaking the MANA III interlock protocol. In particular, if the commitment scheme is not hiding, the adversary may find the used randomizer and the corresponding part of the SSS. On the other hand, in practical applications it is used as a computationally secure $(n, \ell, k, 2^{-(1-\frac{1}{k})\ell} + \epsilon', t)$-authentication

1. A generates a commitment $(C_{i1}, D_{i1}) = \text{COMMIT}(M_{i1})$ and sends C_{i1} to B.
   $A \rightarrow B$: C_{i1}
2. B generates a commitment $(C_{i2}, D_{i2}) = \text{COMMIT}(M_{i2})$ and sends C_{i2} to A.
   $B \rightarrow A$: C_{i2}
3. A responds by sending D_{i1} to B
   $A \rightarrow B$: D_{i1}
   B opens the commitment by computing $\text{OPEN}(C_{i1}, D_{i1})$ and aborts if the commitment does not open.
4. B responds by sending D_{i2} to A
   $B \rightarrow A$: D_{i2}
   A opens the commitment by computing $\text{OPEN}(C_{i2}, D_{i2})$ and aborts if the commitment does not open.

Fig. 6. Round i of an Interlock Protocol Using SSS

protocol with a small ϵ' that depends on ϵ under the assumption that the hash-based commitment scheme in MANA III is ϵ-hiding and ϵ-binding. Currently, it is generally believed that given the computational capabilities of a realistic adversary, this holds for the hash function SHA-256 with $\epsilon' \ll 2^{-\ell}$.

Breaking the hiding and binding properties in practise means either exhaustive search for a key or pre-image. In both cases, the time needed to succeed is approximately $N\epsilon$ where N is the set over which the search is performed. As the SSS can always be guessed with probability $2^{-\ell}$ we can assume that in (t, ϵ)-hiding and binding commitment schemes schemes $\epsilon = 2^{-\ell}$. The size N of the set is determined by the parameters of the commitment scheme. As an example, let us consider the hash function based commitment scheme $(C, D) = (h(M, R), (M, R))$, where the length of the hash code is a bits and the size of the randomizer R is q bits. Then the parameter sizes are in balance if $a = q$ and $2^a 2^{-\ell} \approx t$, where t is the upper bound to the computing time any imaginable real world adversary might have. For example, if $a = q = 128$ and $\ell = 20$, then $t = 2^{108}$.

However, independently of the computational resources of the attacker, the security of MANA III is reduced if the SSS P is used more than once. If P is used t times and k is the number of interlock rounds, then at each time, the adversary gets $\frac{t}{k}\ell$ bits of P. Hence, after the same SSS has been used more than k times an active man-in-the-middle can act as any of the legitimate parties in the interlock protocol. The MANA III interlock protocol achieves cross authentication of the data. But even if it is used for unilateral authentication, it is not sufficient that only the receiver verifies the protocol messages including the opening of commitments. Also the sender must verify the commitments and stop sending any further messages if the verification fails.

Other SSS Association Protocols. The hybrid authentication protocol presented in Section 2 is a variant of an SSS-based authenticated Diffie-Hellman protocol presented in the Appendix of [21] as Mechanism A.3. Bluetooth Secure

Simple Pairing employs this protocol in the context of an out-of-band channel where the number of secret bits is not strictly limited. The protocol in [21] was developed in the IST-SHAMAN project [31]. It takes 2ℓ secret bits to achieve cross-authentication of Diffie-Hellman public keys with forgery probability less than about $2^{-\ell}$. Hence it provides about the same security level as MANA III does with 2 rounds (about 6 protocol messages), but achieves it only with 2 protocol messages over the insecure channel. The protocol is depicted in Figure 7. The SSS is 2ℓ bits and comprises a string K of length ℓ and an authenticator $H_K(h(PK_A))$ of the hash-value of the public key of A. Here it is sufficient to assume that H_K is an ϵ-AU family of hash functions. In the first message over the insecure channel party A sends its public key to party B. In other words, unilateral MANA I is used to authenticate the public key of party A. Then B computes the shared Diffie-Hellman key K_{DH}, uses it to compute a computationally secure MAC the first part of the SSS, which is now used as a one-time password to authenticate B to A. Hence, the idea is very similar to the commonly used approach for mutual authentication in the Web, where server authentication using TLS is used in conjunction with client authentication using HTTP Digest Authentication. In addition to taking only two messages over the insecure channel this protocol has another significant advantage over MANA III. Even if used repeatedly with different K_{DH} the SSS is not revealed to an active attacker. We use the same notation $c(M, K)$ as in Fig. 4 for a computionally secure MAC function.

1. A sends its Diffie-Hellman public key PK_A to B.
   $A \rightarrow B$: PK_A
2. B completes MANA I protocol when receiving PK_A. Then it takes its Diffie-Hellman key PK_B, computes a shared secret Diffie-Hellman key K_{DH} and computes a computationally secure MAC using the key K_{DH} on the first part K of the SSS and sends it to A.
   $B \rightarrow A$: $PK_B, c(K, K_{DH})$
   When receiving $PK_B, c(K, K_{DH})$ party A computes K_{DH} and verifies $c(K, K_{DH})$.

Fig. 7. Hybrid Authenticated Diffie-Hellman Protocol Using SSS

3.3 Authenticated Channel without Secrecy

Protocols using short shared secrets, passwords and passkeys, for securing communication have a long history, whereas only recently it was observed that it is not necessary to keep this short string secret if it is excahnged at the end of the protocol, see Figure 8. In [11] this protocol was called MANA II. Serge Vaudenay was the first to develop a formal model for such a protocol in [42], where he presented also a four-round protocol using SAS and proved its security in the computational security model.

Similarly as in the SSS case distinction between unconditional and computational security is made.

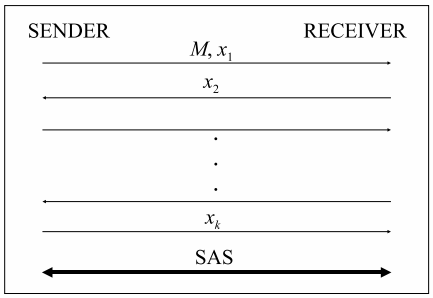

Fig. 8. SAS Authentication Protocol

Unconditionally Secure Data Authentication Using SAS. Unconditional security for data authentication protocols is measured in terms of the success probability an computationally unbounded attacker has when trying to forge a message. We say that a protocol that uses a short authenticated string of length ℓ is an unconditionally secure (n, ℓ, k, ϵ)-authentication protocol if data of length n bits can be authenticated by running k rounds of the protocol such that an adversary's probability of forging the data is bounded from above by ϵ.

The MANA II protocol that uses the Reed-Solomon AU hash family presented above is a $(n, \ell, 1, \epsilon)$-authentication protocol, for $n = (2^\ell \epsilon + 1)\ell$, that is, for data of limited length. For example, if $\epsilon = 2^{-\ell}$, then at most 2ℓ bits can be authenticated. To avoid degradation of security, the standard [21] instructs to hash the data first by using a computationally secure hash function. A second possibility is to use an unconditionally secure data authentication protocol with more rounds. Indeed, similarly as in the SSS-model, it was shown in [27] that there exists a $(n, \log_2^{(k)}(n) + 2\log_2 \frac{1}{\epsilon}, \epsilon)$-authentication protocol in the SAS-model. Further, it was shown that for any unconditionally secure data authentication protocol using SAS the length of the SAS is lower-bounded by $2\log_2 \frac{1}{\epsilon}$ to have the forgery probability bounded from above by ϵ. Breaking this bound is possible only by using one-way functions.

Computationally Secure Data Authentication Using SAS. Given the forgery probability ϵ the minimum SSS length is $\log_2 \frac{1}{\epsilon}$ in the computaional security model. The SSS-based protocols used in the association models of wireless standards do not achieve this optimum length, even if they make use of conjectured one-way functions. In the SAS-model such protocols are known. Next we will describe one such protocol, MANA IV, which is the cryptographic primitive

for the *numeric comparison* association model of Bluetooth Simple Pairing, see Figure 2.

As above n denotes the bit-length of the message, k the number of rounds, and ℓ the number of bits in the SAS. Then a protocol is said to be a computationally secure $(n, \ell, k, \epsilon, t)$-authentication protocol in the SAS-model, if for any polynomial time adversary and for any $c > 0$ the forgery probability is at most $\epsilon + t^{-c}$ for sufficiently large t. Here the parameters n, k and ℓ may also depend on the parameter t.

A and B have exchanged data M.

1. A generates K_1 and computes commitment $(C, D) = \text{COMMIT}(K_1)$ and sends C to B.
   $A \rightarrow B$: C
2. B generates a random number K_2 and sends K_2 to A.
   $B \rightarrow A$: K_2
   A computes $V_1 = h(M; K_1, K_2)$
3. A responds by sending D to B
   $A \rightarrow B$: D
   B opens the commitment by computing $\text{OPEN}(C, D)$ and aborts if the commitment does not open.
   B computes $V_2 = h(M; K_1, K_2)$
4. Both devices check if V_1 equals V_2 over the SAS-channel.

Fig. 9. The provably secure SAS protocol MANA IV

MANA IV is depicted in Figure 9. A formal proof of security is given in [20]. The proof identifies all possible forged interactions that a man-in-the-middle possibly can have with the legitimate parties. For all these attacks a suitable set of countermeasures is extracted. The main task of a security proof is to formalize the extracted countermeasures as a set of security assumptions on the cryptographic primitives used in the protocol.

The security proof of MANA IV given in [20] comprises five lemmata. The first two of them cover the attack strategies where the attacker does not change the synchronisation of the messages. The remaining three lemmata cover the abnormal execution paths, where the man-in-the-middle generates and sends a protocol message to one party before seeing the corresponding message sent by the other party. In what follows we concentrate on explaing Lemma 1 and 2 as they capture the essential security requirements.

Recall that in the presence of a man-in-the-middle the data M_1 held by A is different from the data M_2 held by B in the beginning of the data authentication protocol. We investigate the adversary's possible strategies in two disjoint cases: either the adversary changes the value of the commitment sent in Message 1 or not. We use a prime ' ' to denote data items that are deliberately forged by the man-in-the-middle.

Attack 1: Adversary does not change the commitment. If the man-in-the-middle does not do any changes in the protocol massages at all, then he succeeds if

$$h(M_1; K_1, K_2) = h(M_2; K_1, K_2).$$

This is prevented by assuming that the hash family defined by $H_K(M) = h(M; K)$, where $K = (K_1, K_2)$, is ϵ_u-AU.

Assume now that the man-in-the-middle changes K_2 but not K_1. In this case the changed value K_2' must be such that the SAS verification is successful, that is,

$$h(M_1; K_1, K_2') = h(M_2; K_1, K_2). \tag{1}$$

This would be a computationally easy task if, when selecting K_2', the adversary would know the value of K_1. Even partial knowledge of K_1 may give the adversary some non-negligible advantage. It is therefore required that the commitment scheme that A uses to compute C sent in Message 1 is ϵ_h-hiding.

A second strategy for determining a correct value of K_2' is to guess the value of $h(M_2; K_1, K_2)$ that B will compute. Then the adversary searches for a suitable value K_2' such that Equation (1) holds. The weakest strategy is to use preimage search in the random oracle model. Then the success probability is at most $2^{-\ell}$. We want that approximately this upper bound of the forgery probability also holds for the concrete cryptographic primitive used for h. This is formalized by the following definition.

Definition 4. *A function $h : \mathcal{M} \times \mathcal{K}_1 \times \mathcal{K}_2 \to \mathcal{H}$ is called ϵ-almost universal with respect to the set $\mathcal{K}_1$ if for any $M_1, M_2 \in \mathcal{M}$ and $K_2, K_2' \in \mathcal{K}_2$, $K_2 \neq K_2'$, we have that* $\Pr[h(M_1; K_1, K_2') = h(M_2; K_1, K_2)] < \epsilon$ for a randomly selected $K_1 \in \mathcal{K}_1$.

Finally, let us assume that the adversary changes K_1. This attack strategy is thwarted by the assumtion that the commitment scheme is ϵ_b-binding. Then opening the commitment, originally computed with K_1, to another value K_1', will not succeed for a computationally bounded adversary except for probability at most $\epsilon_b + 2^{-q}$.

By collecting the security requirements extracted above from the different attack strategies, we get the following result. Suppose that the commitment scheme is ϵ_h-hiding and ϵ_b-binding, and that the hash family used to compute the SAS value is ϵ_u-AU and ϵ_u-almost universal with respect to the set $\mathcal{K}_1$. Then the success probability of Attack 1 is bounded from above by $\epsilon_u + \epsilon_h + \epsilon_b$. This result is formally stated in Lemma 1 of [20]. When trying to make the final attack probability as close to the minimum upper bound $2^{-\ell}$, the value of ϵ_u is dominating and must be taken as close to $2^{-\ell}$ as possible. Significantly smaller vales of ϵ_h and ϵ_b for the commitment scheme.

Attack 2: Adversary changes the commitment. The man-in-the-middle changes the commitment after seeing a commitment C sent by A in Message 1.

The man-in-the-middle knows that there exists D such that $(C, D) = \text{COMMIT}(K_1)$. The first strategy is to generate K_1' and compute $(C', D') = \text{COMMIT}(K_1')$. The adversary succeeds if it can find an opening D'' such that $\text{OPEN}(C', D'') = K_2'$ such that

$$h(M_1; K_1, K_2') = h(M_2; K_1', K_2).$$

The requirements to thwart this strategy are very similar to those for Attack 1: the commitments must be hiding and binding, and we will not investigate them here in detail. However, essentially stronger requirements are imposed on the commitment scheme to prevent that the adversary cannot succeed in an attack strategy to be explained next.

The attack exploits some regularities that the family of hash functions and the commitment scheme might have. Let us describe the attack using a small example. The sole purpose of this artificial example is to demonstrate that the hiding and binding properties are not sufficient to prevent exploiting some regularities and relationships between the commitment scheme and the hash family. Let us assume that for the given messages M_1 and M_2 the following holds with a probability over random choices of u, v and w that is significantly larger than on the average:

1. $h(M_1; u, v + 1) = h(M_2; u + 1, v)$, and
2. if $(C, D) = \text{COMMIT}(w)$ then $(C + 1, D + 1) = \text{COMMIT}(w + 1)$.

The adversary can exploit this property in the following strategy. When seeing C in Message 1, the adversary replaces it by $C' = C + 1$, and sends it further to B. Then when seeing K_2 in Message 2, the adversary changes it to $K_2' = K_2 + 1$ and sends it to A. In Message 3, adversary sees D such that $\text{OPEN}(C, D) = K_1$. Then it uses property (2) above and replaces D by $D' = D + 1$ and K_1 by $K_1' = K_1 + 1$. Then based on the assumed properties the adversary achieves some advantage that the commitment C' opens to K_1' and that $h(M_1; K_1, K_2') = h(M_2; K_1', K_2)$.

The precise definition of non-malleable commitment scheme is given in [20]. As the attack has a number of steps described above, also the formal definition of security against this attack is given using three adversial algorithms and their interplay. The less formal and less general definition is given below to capture the attack described above. Note that in MANA IV the messages for the commitment scheme are the random numbers K_1.

Definition 5. *Consider the following adversarial strategy: the adversary selects a function f on the message space of the commitment scheme; then given $C \in \mathcal{C}$ such that $(C, D) = \text{COMMIT}(M)$, she produces C' such that after seeing D she can produce a decommit value D' such that $\text{COMMIT}(f(M), R) = (C', D')$ holds for some R. A commitment scheme is (τ, ϵ)-non-malleabe if for all functions f on the message space of the commitment scheme the success probability of the above described strategy is less than $2^{-q} + \epsilon$, where q is the length of R in bits.*

Within this attack it must be required that the hash-family is ϵ-regular, that is, even if one key, either K_1 or K_2 is fixed, the resulting hash family is ϵ-AU

whatever the fixed key value is. Assuming this and that the commitment scheme is ϵ_m-non-malleable we have that the probability that Attack 2 succeeds for a computationally bounded adversary is less than $\epsilon + \epsilon_m$.

After having explained some aspects of the security proof of the MANA IV-protocol let us finally state the theorem for which the complete proof is given in [20].

Theorem 1. *For any t, there exists $\tau = t + \mathcal{O}(1)$ such that if the commitment scheme is (τ, ϵ_b)-binding, (τ, ϵ_h)-hinding, and (τ, ϵ_m)-non-malleable and h is ϵ-regular and ϵ-almost universal with respect to $\mathcal{K}_1$, then MANA IV is a $(n, \ell, 3, \epsilon + 2\epsilon_b + 2\epsilon_h + \epsilon_m, t)$-data authentication protocol.*

With a suitable choice of the hash family the value of ϵ is very close to $2^{-\ell}$. The other ϵ-values can be made arbitrarily small by choosing a sufficiently strong commitment scheme. Similarly as MANA III, the practical instantiations of MANA IV make use of hash function based commitment schemes. Also the almost universal hash families are implemented in practice using a computationally secure hash function which is truncated to produce a short output. It would be feasible to experimentally analyze how well, that is, for how small ϵ such a construction satisfies the requirements of ϵ-regularity and ϵ-uniformity with respect to $\mathcal{K}_1$. To our knowledge such analysis has not been performed.

MANA IV is deployed as the cryptographic solution to the numeric association model of Bluetooth Secure Simple Pairing. A variant of it that uses the Diffie-Hellman public keys as randomizers to the commitment has been adapted for the Wireless USB association models. The security proof of MANA IV explained above, can be extended to cover this WUSB variant of the protocol. However, as shown in [20], it requires a stronger universality property of the hash function. An example of a hash function which does not satisfy this property is a function h defined as $h(M; K_1, K_2) = h_0(M; K_1) \oplus K_2$ for any hash function h_0, where the output of h_0 is short. The hash function used to compute the checksums for the numeric comparison in WUSB association is not known to have any such weakness.

4 Secure channels and physical interfaces

In this section, we survey various types of secure channels and physical interfaces and how they can be used for key establishment in the various methods we looked at in Section 2.

4.1 Out-of-band Secure Channels

Out-of-band channels are communication channels distinct from the insecure channel over which the devices normally communicate. Using out-of-band channels to aid in association and key establishment can greatly improve usability by minimizing user actions. Therefore, from very early on [37] researchers have looked for ways of using out-of-band channels in key establishment.

Various types of out-of-band channels have been considered in the literature including physical contact [37], infrared [1], audio channels [36], visual channels [24, 34], very short-range wireless communication channels like Near Field Communications (NFC)[7]. Different types of channels have different characteristics which affect their applicability to the different methods we saw in Section 2. The characteristics that are relevant for key agreement are the following:

- **Channel security**: All useful types out-of-band channels are assumed to provide *integrity*: an attacker is assumed incapable of modifying, inserting or deleting messages sent via the channel. Some types are assumed to provide *secrecy* as well: an attacker is assumed incapable of reading the information sent via the channel. Usually physical connections and NFC channels are assumed to provide secrecy; however the validity of these assumptions have been questioned [16].
- **Directionality**: Depending on the hardware available on the devices, the out-of-band channel may be unidirectional or bidirectional.
- **Bandwidth**: Bandwidth of a channel is the rate at which it can transfer data. The bandwidth of an out-of-band channel is relevant in key establishment because it influences the time it takes to complete the association process.

Table 1 lists the protocols from Section 2 that can be implemented using out-of-band channels. Each row also lists papers which describe how different types of out-of-band channels are used with that protocol.

Method	Integrity	Secrecy	Directionality	data size	Refs.
P1: Key transport		√	1-way	128-256 bits	[37]
P4: Exchange of key commitments	√		2-way	128-256 bits	[1, 24, 36]
P6: Short string comparison	√		1-way[1]	12-20 bits	[34]
P8: Transfer of (short) secret		√	1-way	12-20 bits	
P10: Transfer of key commitment and secret	√	√	1-way	256-512 bits	

Table 1. Applicability of out-of-band channels

[1] For mutual authentication, the method relies on the user as the return channel.

Although the promise of better usability is the motivation for using out-of-band channels in key establishment, the downside is the need to have the necessary hardware interfaces on both devices. There is no universal out-of-band

[7] `http://www.nfc-forum.org`

channel guaranteed to be available on all devices. The vast majority of personal devices are low-cost commodity devices. Therefore adding a new hardware interface simply for the purpose of easing the association process is usually not an economically viable option. Researchers have therefore investigated ways to establish associations while maximizing security, usability and cost. One approach is to design the association procedures taking the resource asymmetry between the devices involved in the association. Typically one device, like a laptop or phone, has greater capabilities, while the other, like an access point or headset, is extremely resource constrained and cost-sensitive. Saxena, et al., [34] describe setting up a security association using a visual channel: one device is assumed to have a video camera while the other device needs to have only a single light source (such as a light-emitting diode) and mechanisms for user confirmation (like buttons for indicating yes and no).

In the next two sections we will look at two radically different approaches for key establishment to the same end of balancing usability, security and cost.

4.2 Integrity-protected Logical Channels

Recall that in Section 2, we assumed an attacker with standard Dolev-Yao capabilities over the insecure channel. Čagalj, et al., [44] observed that this assumption is too strong in the short-range wireless channels typically used for proximity communication in personal devices. They argued that although an attacker can insert signals, it is much more difficult for him to *erase* a signal emitted by one of the devices without the disruption being detected. To be able to erase a signal the attacker has to know the exact characteristics of the signal so that he can match its phase, amplitude and frequency. The legitimate sender can further frustrate the attacker by choosing an encoding scheme where these parameters can be randomized by the sender.

Čagalj, et al., design a mechanism called *I-codes*[44] as follows. Communication on the channel uses a simple on-off coding scheme: a 1-bit is represented by the presence of a signal (of arbitrary phase, amplitude and frequency) during a certain time period; a 0-bit is represented by the absence of any signal. The recipient only measures the average power level during the time slot to decide if the slot corresponds to a 1-bit or 0-bit. As a result, while an attacker can turn a 0-bit into a 1-bit, he cannot do the reverse. The actual message itself is encoded in such a way that the representation of either message bit has at least one 1-bit in the encoded message. An example of such an encoding is Manchester coding which represents a 1-bit as "10" and a 0-bit as "01". Figure 10 illustrates this scheme using an example. A recipient who decodes the bits received over the channel and extracts the message can be certain that the message is exactly what the sender intended. I-codes therefore consitutes logical integrity-protected channel realized over the insecure channel. Consequently, key agreement protocols utilizing authentication over unspoofable channels (Protocols **P4** and **P6** from Figure 1) can be realized using I-codes.

The advantages of this approach is that it minimizes user interaction and does not impose any requirements on additional hardware interfaces. The dis-

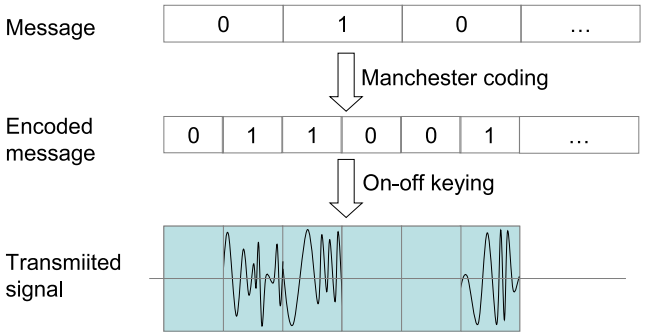

Fig. 10. Integrity Coding example using Manchester codes and on-off keying [44]

advantage is that I-codes need to be implemented at a sufficiently low level in the protocol stack and there should not be any other legitimate traffic in the vicinity. This limits the applicability of the method.

4.3 Key Extraction from Shared Environment

An alternative approach for key establishment is to exploit the fact that the devices involved are in close proximity and thus share a common environment. The first proposal for enviromental key extraction was due to Denning and MacDoran who described shared key establishment based on geo-location [6]. Devices equipped with Global Positioning System (GPS) receivers can map the raw data received from the GPS satellites into a shared secret.

Environmental key extraction requires that the devices have the ability to sense some form of ambient data such that (a) if two devices A and B are in close proximity at the same instant of time, then the "distance" between the sensor readings of v_A and v_B of A and B respectively is small and can be computed efficiently given v_A and v_B and (b) sensor readings taken at different times in the same location, or at different locations at the same time cannot be used to estimate v_A or v_B.

Mayrhofer and Gellersen [23] propose accelerometer readings from two devices shaken together as the source for such ambient data. Varshavsky, et al., [41] propose using received signal strength measurements from radio transmitters in the vicinity for the same purpose. The extracted key can be used directly (**P12** in Figure 1) or it can be used to authenticate asymmetric key agreement (**P9** Figure 1).

Mayrhofer [22] described protocols for the first case: extracting a long shared symmetric key from sensing the environment The protocol is essentially as follows.

1. Both devices read sensor data and extract certain features according to a set of pre-defined methods. Each feature vector is called a *candidate key part.*

2. Devices exchange randomized hashes of each candidate key part along with the randomizing value. Now each device can calculate which candidate key parts match.
3. Steps 1 and 2 are repeated until each device gathers enough matching candidate key parts. Each device computes a candidate key from the candidate key parts and exchange hashes of the candidate keys as key confirmation.

To achieve resistance against dictionary attacks, each candidate key part must be sufficiently long. Therefore using fewer key parts is recommended.

Mayrhofer and Gellersen [23] as well as Varshavksy, et al., [41] describe protocols for the second case: extracting a shared secret and using it to authenticate key agreement. This is useful when environmental key extraction will only yield a short secret. Since the extracted keys are not likely to be identical, they are not used in the same way as a shared password in password-authenticated key agreement protocols. Instead, devices exchange the feature vectors extracted from the environment and check whether they match. To thwart an active man-in-the-middle, both proposals use the Rivest-Shamir interlock protocol [33]. For example, the protocol in [23] is as follows:

1. Devices perform anonymous Diffie-Hellman key agreement.
2. Devices read sensor data and extract certain features according to a set of pre-defined methods. Each device encrypts the feature vectors using the Diffie-Hellman key and split the encryption into two equal parts.
3. Devices exchange the respective first halves of the encryption.
4. Devices exchange the respective second halves of the encryption.
5. Devices reassemble the received halves, decrypt the result and check if the plaintext matches the locally measured feature vectors.

Although neither paper explicitly says it, Rivest-Shamir interlock protocol is not secure in an asynchronous communication model, as we saw in Section 3. An active man-in-the-middle, which has run separate instances of the Diffie-Hellman protocol with each of the devices, can succeed in completing the interlock protocol with the second device. It stops the messages in step 3 from reaching the intended receivers and sends a bogus message to the first device, and then receives the message in 4 from the first device. Now the man-in-the-middle reassembles the two halves of the encryption received from the first device and decrypts it to get the sensor data sent by the first device. Then it can run the interlock protocol successfully with the second device.

Therefore, it is necessary to clearly separate steps 3 and 4 into different synchronous rounds. To do this, one must resort either to synchronized clocks and pre-defined timeout values, or to relying on user assistance, as done in the MANA III protocol in [11].

Castelluccia and Mutaf [5] proposed a different approach for extracting a key from shared environment. They made the assumption that when two devices broadcast over a radio interface while shaken together, it would be difficult for an attacker observing their transmissions to tell which transmission emanates from which device. However, each device can easily distinguish its own transmissions

from its peer's transmissions. This lends to a simple and elegant scheme for constructing a shared key: each device picks a key string; for each key bit in its key string, the device transmits an empty packet and puts either its own address or the receiving peer's address as the sender address depending on if the key bit is 1 (true) or 0 (false), respectively. The peer can reconstruct the key bit by checking if the address is correct or not. The security of the scheme relies on the assumption that an attacker cannot tell who is transmitting. But techniques for radio fingerprinting may render this assumption invalid [9, 32].

5 Association Models in Standards for Personal Networks

In this section, we survey the association models proposed in four emerging standards [35, 46, 47, 28]. We then compare them by referring to the classification presented in Section 2.

5.1 Bluetooth Secure Simple Pairing

Bluetooth Secure Simple Pairing (SSP) [35] is a standard developed by Bluetooth Special Interest Group. It is intended to provide better usability and security than the original Bluetooth pairing mechanism, and is expected to replace it. Simple pairing consists of three phases. In the first phase, the devices find each other and exchange information about their user input/output capabilities and their elliptic curve Diffie-Hellman public keys for the FIPS P-192 curve [29]. In the second phase, the public keys are authenticated and the Diffie-Hellman key is calculated. The exact authentication protocol, and hence the association model, is determined based on the device user-I/O capabilities. In the third phase, the agreed key is confirmed (in one association model, the authentication spans both the second and third phase).

SSP supports four different association models: Numeric Comparison, Passkey entry, 'Just Works' and Out-of-band models. Now we will examine each of these models and the protocols they use for authentication in phase 2.

Numeric comparison model is where the user manually compares and confirms whether the short integrity checksum displayed by both devices are identical (Figure 1: **P5**). The compared checksum is 6 digits long. The phase 2 protocol is an instantiation of the protocol in Figure 2.

Passkey entry model is targeted primarily for the case where only one device has a display but the other device has a keypad. The first device displays the 6-digit secret passkey, and the user is required to type it into the second device. The passkey is used to authenticate the Diffie-Hellman key agreement (Figure 1: **P7**). The protocol is based on user-assisted authentication by shared secret in Figure 3 with 20 rounds ($k = 20$). Devices prove knowledge of one bit of the passkey in each round.

'Just works' model is targeted for cases where at least one of the devices has neither a display nor a keypad. Therefore, unauthenticated Diffie-Hellman

key agreement is used (Figure 1: **P11**) to protect against passive eavesdroppers but not against MitM attacks.

Out-of-band model is intended to be used with different out-of-band channels, in particular with Near Field Communication technology. Device D_A uses the out-of-band channel to send a 128-bit secret r_a and a commitment C_a to its public key PK_a. Similarly, D_B uses the out-of-band channel to send r_b and C_b. If out-of-band communication is bidirectional, mutual authentication is achieved by each party verifying that the peer's public key matches the commitment received via the out-of-band channel. (Figure 1: **P4**).
If the out-of-band channel is only one way, the party receiving the out-of-band message can authenticate the public key of its peer. However, the party sending the out-of-band message must wait until the third, key confirmation, phase of SSP which we now describe.

In phase 3, the same key confirmation protocol is executed in all association models to confirm successful key exchange by exchanging message authentication codes using the newly computed Diffie-Hellman key. Each device includes the random value r received from the peer in the calculation of its message authentication code. In the one-way out-of-band case, the message authentication code serves as a proof-of-knowledge of the shared secret r received out-of-band. This is the hybrid authentication protocol **P10** (Figure 4).

Peer discovery: In current Bluetooth pairing, peer discovery is left to the user: the user initiates pairing from one device which constructs a list of all other Bluetooth devices in the neighborhood that are publicly discoverable and asks the user to choose the right one to pair with. In the out-of-band association model, device addresses are sent via the out-of-band channel. This makes it possible to uniquely identify the peer to pair with, without requiring user selection. SSP does not contain any new mechanisms to make peer discovery easier in the other association models. Individual implementations could use existing Bluetooth modes, like the "limited discoverable mode" and "pairable mode" to support user-conditioning on the peer device. However, since such user-conditioning is not mandated by the specification, it is quite possible that the implementations of SSP may still need to resort to asking the user to choose the right peer device from a list.

Model selection: The association model to be used is uniquely selected during the initialization of the session. If the association process is initiated by out-of-band interaction, and security-information is sent through the out-of-band channel, then the out-of-band model is chosen automatically. Otherwise, in phase 1, the devices exchange their input-output capabilities. The SSP specification describes how these capabilities should be used to select the association model.

5.2 Wi-Fi Protected Setup

Wi-Fi Protected Setup (WPS) is Wi-Fi Alliance's specification for secure association of wireless LAN devices. Microsoft's Windows Connect Now (WCN) includes a subset of association models described in WPS. The objective of WPS

is to mutually authenticate the enrolling device with the Wi-Fi network and to deliver network access keys to the enrolling device. This is done by having the enrolling device interact with a device known as the "registrar", responsible for controlling the Wi-Fi network. The registrar may be, but does not have to be, located in the Wi-Fi access point itself. WPS supports three configuration methods: In-band, out-of-band, and push-button configurations.

In-band configuration enables associations based on a shared secret passkey (Figure 1: **P7**). The user is required to enter a passkey of enrollee to the registrar. This passkey may be temporary (and displayed by the enrollee) or static (and printed on a label). 8-digit passkeys are recommended but 4-digit passkeys are allowed. The passkey is used to authenticate the Diffie-Hellman key agreement between the enrollee and the registrar. The protocol used is a variation of the modified MANA III protocol in Figure 3 with two rounds ($k = 2$).

As in MANA III (Figure 3), once a passkey is used in a protocol run, an attacker can recover the passkey by dictionary attack (although in this instantiation, the attacker needs to be active since the computation of the used commitments includes a key derived from the Diffie-Hellman key).

Out-of-band configuration is intended to be used with channels like USB-flash drives, NFC-tokens or two-way NFC interfaces. There are three different scenarios:

1. Exchange of public key commitments (Figure 1: **P4**), typically intended for two-way NFC interfaces, where the entire Diffie-Hellman exchange and the delivery of access keys takes place over the out-of-band channel.
2. Unencrypted key transfer (Figure 1: **P1**). An access key is transmitted from a registrar to enrollees in unecrypted form, either using USB-flash drives or NFC-tokens.
3. Encrypted key transfer. This is similar to the previous case, except that the key is encrypted using a key derived from the (unauthenticated) Diffie-Hellman key agreed in-band. From a security perspective, this is essentially out-of-band key transfer (Figure 1: **P1**).

Push button configuration is an optional method that provides an unauthenticated key exchange (Figure 1: **P11**). The user initiates the Push button configuration by conditioning the enrollee (e.g., by pushing a button), and then, within 120 seconds the user has to condition the registrar as well. The enrollee will start sending out probe requests to all visible access points inquiring if they are enabled for push button configuration. Access points are supposed to respond affirmatively only when their registrar has been conditioned by the user for this configuration. If a device or registrar sees multiple peers ready to start push button method, it is required to abort the process and inform the user.

Peer discovery: Enrollees start association in response to explicit user conditioning. They scan the neighborhood for available access points and send Probe Request messages. The Probe Response message has a "SelectedRegistrar" flag

to indicate if the user has recently conditioned a registrar of that access point to accept registrations. This is mandatory for push button configuration but is optional for other models. Thus it is possible that user may have to be asked to select the correct Wi-Fi network from a list of available networks.
Model selection: The model is explicitly negotiated at the beginning.

5.3 Wireless USB Association Models

Wireless USB (WUSB) is a short-range wireless communication technology for high speed data transmission. WUSB Association Models Supplement 1.0 specification [47] supports two association models for creating trust relationships between WUSB hosts and devices:

Cable model uses out-of-band key transfer (Figure 1: **P1**) and utilizes wired USB connection to associate devices. Connecting two WUSB devices together is considered as an implicit decision and, hence, the standard does not require users to perform additional actions like accept user prompts.

Numeric model relies on the users to authenticate the Diffie-Hellman key agreement by comparing short integrity checksum values (Figure 1: **P5**). The protocol is an instantiation of the protocol in Figure 2. First D_A and D_B negotiate the length of the checksum to be used. The specification requires that WUSB hosts must support 4-digit checksums whereas WUSB devices must support either 2 or 4-digit checksums.

Peer discovery: The association is initialized by implicit or explicit user conditioning. Attaching a USB-cable is interpreted as an implicit conditioning. The user pressing a button is an example of explicit user conditioning. In the numeric model the user sets a USB device to search for hosts and a USB host to accept connections. The host advertise its willingness to accept a new association in the control messages it transmits on the WUSB control channel.
Model selection: The choice of the association model is based on the type of user conditioning done. In case a cable is plugged, the devices exchange information on whether they support cable association. If so, they use cable model. If conditioning is explicit, they use numeric model.

5.4 HomePlugAV Protection Modes

HomePlugAV is a power-line communication standard for broadband data transmission inside home and building networks. In addition to protecting deliberate attacks, association mechanisms are used to create logically separate subnetworks by distributing an 128-bit AES network encryption key (NEK) for devices in each subnetwork. As with WPS, each HomePlugAV network has a controller device. HomePlugAV supports the following association models [28]:

Simple connect mode uses unauthenticated symmetric crypto based key agreement to agree on a shared key (Figure 1: **P3**). This network membership key

(NMK), is used to transport NEK to the new device. The key agreement process is as follows. To admit a new device, the user is required to first condition the controller device, and then condition the new device, e.g., by turning on its power. The devices find each other and exchange nonces. A temporary encryption key (TEK) is formed by hashing the two nonces together. The controller encrypts the NMK using the TEK and sends it to the new device.

Secure mode allows new devices to have a secret passkey, of at least 12 alphanumeric characters long, typically printed on a label. The user is required to type in this passkey to the controller device. The controller device uses it to construct an encryption of NMK and send it to the new device. The keys for devices joining in secure mode is different from the keys for devices joining in simple connect mode. This is an example of authenticated symmetric crypto key agreement (Figure 1: **P2**).

Optional modes enable alternative use of alternative models for distributing NMKs or NEKs between devices. These include "manufacturer keying" where a group of devices have a factory installed shared secret, and external keying, where trust is bootstrapped from other methods.

MitM attacks are prevented in simple connect mode by utilizing characteristics of powerline medium. Before two nodes can communicate, they must negotiate tone maps, which enable devices to compensate disturbances caused by powerline channel. This negotiation is done in a reliable, narrow-band broadcast channel. Thus a MitM trying to negotiate tone maps with the legitimate endpoints will be detected.

Passive eavesdropping in the point-to-point channel is difficult since an attacker, even with the knowledge of the tone maps used between the legitimate endpoints, will not be able to extract the signal from the channel because the signal-to-noise ratio will be too poor at different locations, particularly, when the attacker is outside a building and the legitimate end points are inside. Also, licensees of HomePlugAV technology do not provide devices that can extract signal without negotiating tone maps. Hence, attackers must be able to build expensive devices for eavesdropping.

Peer discovery: In simple connect mode the peer discovery is performed by the user conditioning the devices into a suitable modes, and the new device scanning the network to find a controller that is willing to accept new devices.

Model Selection: The model is selected by user conditioning. There is no automatic negotiation.

6 Evaluation and Analysis of Proposed Association Models

In this section, we analyze the association models described in Section 5 from different perspectives and point out some problematic areas.

6.1 Comparison of Security levels

First we summarize and compare the security levels provided by the different association models discussed in Section 5. A comparative summary of models' security characteristics are presented in Table 2.

Association Model	**Offline Attacks**		**Online Active Attacks**		
	Protection	Work[1]	Protection	Success Probability	Work[2]
Bluetooth Secure Simple Pairing					
Numeric Comparison	DH	2^{80}	6 digit checksum	2^{-20}	2^{148}
Just Works	DH	2^{80}	-	1	0
Passkey Entry	DH	2^{80}	6 digit passkey	2^{-19}	2^{147}
Out-of-band	DH	2^{80}	OOB security	-	2^{128}
Wi-Fi Protected Setup					
In-band	DH	2^{90}	8 digit passkey	$2^{-13.2}$	$2^{141.2}$
In-band + OOB [3]	DH	2^{90}	OOB security	2^{-128}	2^{196}
Out-of-band	OOB	2^{90}	OOB security	-	-
PushButton	DH	2^{90}	-	1	0
WUSB Association Models					
Numeric Model	DH	2^{128}	2/4 digit checksum	$2^{-6.6}$ or $2^{-13.2}$	$2^{262.6}$ or $2^{269.2}$
Cable Model	OOB	2^{128}	OOB	-	-
HomePlugAV Protection Modes					
Simple Connect	SNR	High	traffic monitoring	Low	High
Secure Mode	AES	2^{72}	passkey	2^{-72}	2^{72}

Table 2. Comparison of security characteristics of association models

[1] Rough work effort estimates based on [2, Table 2] and [17, section 8]
[2] Work effort to break commitments exchanged, with probability 1
[3] OOB passkey + checksum

Offline Attacks The out-of-band association models rely on the secrecy of out-of-band communication to protect against passive attacks against key agreement. The in-band and hybrid models in all of the standards except HomePlugAV use Diffie-Hellman key agreement to protect against passive attacks. The level of protection depends on the strength of the algorithms and the length of the keys used. In the "Work" subcolumn under the "Offline Attacks" column of Table 2, we use some recent sources [17, 2] to estimate the amount of work an attacker has to do in order to be successful. The figures correspond to approximate lower bounds, and should be treated as rough ballbark estimates only. Offline attack protection in HomePlugAV relies on the characteristics of the power-line communications: namely the signal-to-noise ratio make it difficult for an attacker

to eavesdrop. The HomePlugAV secure mode uses symmetric key encryption as protection.

Online Active Attacks Mounting an online active attack as a man-in-the-middle against key agreement is significantly more difficult than passive eavesdropping. Several of the models ('Just Works', 'Push Button', and 'Simple Connect') trade off protection against man-in-the-middle attacks, in return for increased ease-of-use.

Other in-band association models rely on authentication as the means to protect against online active attacks. The probability of success for an online active attack depends on the length of the key as well as the protocol. The Bluetooth SSP numeric comparison model uses 6-digit checksums leading to a success probability of $\frac{1}{1000000}$. The WUSB numeric model allows a success probability of $\frac{1}{100}$ when two digit checksum is used, and $\frac{1}{10000}$ when four digit checksum is used. These probabilities do not rely on any assumptions about the computational capabilities of the man-in-the-middle.

Association models based on numeric comparison use cryptographic hash functions as the commitment function. In principle, a man-in-the-middle who can break the hiding property of the hash commitment function *during* the key agreement process can also succeed by figuring out the nonce used in the commitment. We show this in Table 2, in the "Work" subcolumn under the "Online Active Attacks" column by indicating the amount of *on-line* work the attacker has to perform in order to succeed with probability 1. In this case, assuming that the hash function is strong, and requires exhaustive search to find the correct pre-image, the work factor depends on the size of the nonce and the size of the checksum. Bluetooth SSP uses 128-bit nonces and 20-bit checksum; therefore we use the figure 2^{148}. WUSB numeric model uses the Diffie-Hellman public value as the hidden nonce, which is based on a 256-bit long private value. It uses 2- or 4-digit checksums. Hence, we use a work factor figure of $2^{262.6}$ or $2^{269.2}$. These figures correspond to the amount of on-line work required for the attacker to succeed with probability 1.

Association models based on passkeys also use cryptographic hash functions as the commitment function. An attacker who can break the hiding property of the hash function can figure out the nonce and the passkey component used in a given round. The work factor depends on the size of the nonce plus the size of the passkey component. For Bluetooth SSP the work factor is 2^{147} (128-bit nonce and 19-bit passkey component), whereas for WPS in-band model the work factor is $2^{141.2}$ (128-bit nonce and 4-digit passkey component). Alternatively, an attacker who can break the binding property of the hash function can send a a randomly chosen value as h_{i2} in Step 2 of the protocol (Figure 3), learn the passkey after receiving message 3 and then calculate a suitable R_{i2} that matches the alleged commitment sent earlier in Step 2. The work factor depends on the size of the commitment. Bluetooth SSP uses 128-bit commitments, leading to a work factor of 2^{128}. WPS uses 256-bit commitments, but the size of the random input is only 128-bit. Thus, although 2^{128} amount of work is sufficient to break

the binding property, the attacker cannot always succeed, since he may have used a value in Step 2 for which there is no 128-bit pre-image. Therefore, we stick with the $2^{141.2}$ work factor discussed above.

Recall from Section 2 that with n bit passkeys and k rounds the success probability for an online active attack against the passkey protocols is $2^{-(n-\frac{n}{k})}$. Bluetooth SSP passkey entry model uses 6-digit ($n \approx 20$) one-time passwords in $k = 20$ rounds. This leads to approximately $\frac{1}{1000000}$ success probability. WPS network uses essentially the same protocol, but in two rounds only. This leads to success probabilities of $\frac{1}{100}$ when 4-digit passkeys are used, and $\frac{1}{10000}$ when 8-digit passkeys are used. In both cases, the passkey must be single-use. If the passkey is re-used, the success probability of man-in-the-middle rises dramatically, reaching 1 after the k^{th} re-use, where k is the number of rounds in the original protocol. In other words, if the same fixed passkey in WPS network model is re-used even *once*, the man-in-the-middle can succeed in the next attempt with certainty. As before, we can estimate the on-line work effort the attacker has to do to break the hash commitments. HomePlugAV secure mode uses a 12 character passkey which is used to generate a key for AES encryption, leading to a probability of 2^{-72} and the amount of on-line work effort is 2^{72}.

The hybrid models using a one-directional out-of-band channel, the random secret transferred using the out-of-band channel is 128 bits long leading to a computational security of 2^{-128}.

Wi-Fi and Bluetooth have legacy association models. If a device supports both the improved and the legacy association models, it is vulnerable to a bidding down attack, which is difficult to detect without relying on the user.

Associations with Wrong Peers Unauthenticated association models face the risk of a device being associated with a wrong peer. For instance, in WPS push button model, the user may condition first the enrollee to search for registrars before conditioning the registrar. If the attacker sets a bogus registrar to accept connections before the users does it with the legitimate registrar, the enrollee associates with the attacker's registrar. Only in the case when both registrars, the bogus and the legitimate one, are simultaneously accepting connections, is the procedure aborted.

In HomePlugAV Simple Connect mode, the user sets the control device to accept connections before starting the joining device up. This could be used to reduce the probability for an attacker to successfully masquerading as a bogus control device because since, if the new device sees multiple control points, it can abort association. However, the mode is potentially vulnerable for fatal errors where the user is slow to switch power to the new device. In this case an attacker may connect to user's control point and get the network encryption key.

6.2 Usability Evaluations

Although the standards discussed in Section 5 specify various association models, they do not mandate specific user interactions. For example, for the numeric

comparison model (for protocol **P5** in Figure 1) used by Bluetooth SSP as well as WUSB Association Models, there are three obvious possibilities for the interaction methods:

1. **Compare-and-Confirm**: Each device shows its checksum on its display. The user is then prompted to compare the displayed strings and indicate, on each device, whether the two strings are the same or not.
2. **Select-and-Confirm**: During standardization discussions, there was some concern that the *Compare-and-Confirm* method might be too easy for the users leading to their answering the prompt without actually doing the comparison. A comparison method that forces the user to pay more attention might be preferable. In the *Select-and-Confirm method*, one device shows the checksum on its display. The other device shows a set of values including its own checksum, as well as some other randomly chosen strings. On the second device, the user is asked to select the entry that matches the string shown on the first device, or indicate a failure if there is no matching value. If the entry chosen by the user matches its own checksum, the second device indicates success. Otherwise it indicates a mismatch. On the first device, the user is prompted whether the second device indicated success or not.
3. **Copy-and-Confirm**: Not all devices have displays. A typical pairing scenario is between a phone/computer and a keyboard. The *Copy-and-Confirm* method is intended to be used in such scenarios. The device with the display shows its checksum and asks the user to type this value into the second device. The second device compares the entered value with its own checksums and indicates success if the values are the same. On the first device the user is prompted whether the second device indicated success or not.

Similarly, for the passkey association model (for protocol **P7** in Figure 1) used by Bluetooth SSP as well as WiFi WPS, there are two possible user interaction methods:

4. **Copy**: One device chooses a passkey and displays it to the user and the user is asked to type the displayed value into the second device. The devices automatically run shared secret authentication protocol which succeeds or fails depending on the user's ability to copy the passkey correctly into the second device and the presence of an active attacker. Unlike in the *Compare-and-Confirm method*, no further user interaction is needed here.
5. **Choose-and-Enter**: The user is asked to choose a random passkey and enter it into both devices. Then the devices automatically run shared secret authentication protocol which succeeds or fails depending on the user's ability to enter identical values into both devices and the presence of an active attacker.

Given these different possibilities, it is natural to ask which ones are preferable in terms of usability and whether the choice of user interaction has any impact on the security of the model. This section is based on the work by Uzun, al., [39] which is a first step towards answering some of these questions.

User errors are grouped into two categories. A *fatal error* results in the violation of a security goal. All other errors are *safe errors*. For **P5** a fatal error occurs when the checksums computed by each device are different, but user input causes one or both devices to conclude that the checksums match. Fatal errors are possible in all three interaction methods of **P5**.

For **P7** a fatal error occurs if the user chooses an easy-to-guess passkey in the *Choose-and-Enter* method. There is no possibility of a fatal error in the *Copy* method.

Uzun, et al., ran two rounds of user testing where each round consisted of 40 subjects with similar characteristics (mostly young, mostly male, and well-educated). In the first round, they used straight-forward user interactions. For example, *Compare-and-Confirm* is implemented by presenting randomly generated 4-digit numbers as "checksums". In half the cases, chosen randomly, the checksums shown on the two devices were the same. In the other half, they were different. The user prompt was the question "Check if both devices display the same value". Users were given two button choices labeled as YES and NO to give their answers. The default key was mapped to YES. Table 3 summarizes the results of the first round.

Table 3. Summary of first round usability tests

Method	Variant	Avg. Comp. Time (sec.)	Fatal Error Rate	Total User Error Rate
Compare-and-Confirm		15.6	20%	20%
Select-and-Confirm		22.5	12.5%	20%
Copy-and-Confirm		27.6	10%	20%
Copy	4-digits	20.8	N/A	7.5%
	8-digits	31.7	N/A	5%
Choose-and-Enter		32.7	>42.5%	45%

In addition to the quantitative measurements, test subjects were also asked for subjective feedback about the different methods. *Copy-and-Confirm* as well as *Copy* were perceived to be hard to use but more secure. *Compare-and-Confirm* and *Select-and-Confirm* were perceived to be easy to use but less secure. As can be seen, all interaction methods that were susceptible to fatal errors did exhibit them.

In the second round of testing, Uzun, et al., decided to focus on the methods *Compare-and-Confirm*, *Select-and-Confirm* and *Copy*. Based on the first round experience, several changes were made with the intention of improving usability and security, as described below. All methods are tested with 6-digit numbers, used either as checksum or passcode. This value was chosen because it is the longest value mentioned in the standards. Although WPS allows 8 digit passcodes, it was ruled out based on the results of the first round, as well as the established cognitive fact that the maximum number of chunks of information that can be kept in working memory is 7 [26]. In the user interface, the numeric code was consistently referred to as a PIN, regardless of whether it was used as a passkey or checksum because it was a familiar concept to users. In *Compare-and-Confirm*, the wording of the question was changed to "Compare

the PIN numbers shown on both devices, are they DIFFERENT?" and user was given two choices of SAME and DIFFERENT. The default response key was assigned to the option DIFFERENT, so that accidental or careless user error will no longer be a fatal error (Note also that the default label used exactly the same word as in the question).

The data collected in this round is summarized in Table 4. As can be seen, the fatal error rate of *Compare-and-Confirm* improved to zero down from 20%. The subjective opinions of the users remained similar to the first round: *Copy* was considered hard to use but more secure, while *Compare-and-Confirm* and *Select-and-Confirm* were considered easy to use but less secure.

Table 4. Summary of second round usability tests

Method	Variant	Avg. Comp. Time(sec.)		Fatal Error Rate	Total Error Rate
		Match	No match		
Compare-and-Confirm	6-digit & new GUI	16.4	13	0%	2.5%
Select-and-Confirm	6-digit & new GUI	16.4	26.4	5%	7.5%
Copy	6-digit	13	N/A	N/A	2.5%

The differences in the test set-up between the rounds and the relatively small number of participants imply that further user testing is needed to arrive at definitive conclusions. Still, some general trends could be inferred from the results above.

- Default user action (e.g., default button) must correspond to the safest choice.
- User actions must be labeled using words that are specific to the task expected from the user. Generic (and familiar) labels like YES/NO, CANCEL/CONTINUE should be avoided. Especially those labels that have direct negative and positive associated meaning should be avoided.
- Multi-step interactions, such as those involved in *Select-and-Confirm* or *Copy-and-Confirm* where users can inadvertently and easily change the prescribed order of interactions should be avoided. If such interactions are unavoidable, the UI should make sure that it is difficult to change the prescribed order.

It is also worth noting that user perceptions did not always agree with reality. Although all the methods tested have roughly the same level of security, users felt that the variants involving a passkey are more secure than the variants involving checksums. Further, if average time of completion can be considered an objective measure of ease-of-use, then *Copy* is comparable to *Compare-and-Confirm*, even though users felt otherwise.

To summarize, we conclude the following. *Copy* is inherently resistant to fatal errors. Fatal errors in *Compare-and-Confirm* can potentially be avoided by careful design of the UI but further usability testing is needed to validate this conclusion.

6.3 Further challenges in Implementing Multiple Association Models

Above, we saw how naive implementations of user interaction could increase the likelihood of fatal errors. In this section, we look at further similar challenges in implementation arising out of the fact that the standards invariably support multiple association models simultaneously.

Consider specifications that support an unauthenticated association model as well as user-assisted comparison of integrity checksums. An example is a Bluetooth device that supports the numeric association model and the 'Just Works' model. Figure 11 illustrates a MitM attacker who can intercept messages exchanged during an association. The first associated device has a display and the second may or may not have a display. The attacker changes device capability information so that the first device will be using the numeric comparison model and that the second device will be using 'Just Works' model. This leads to a situation where the first device shows a 6-digit checksum and the second device, using 'Just Works' model, does not display a checksum, even if it would have a display. The user may have been educated to detect a mismatch in checksums. But now, when only one device displays a checksum, the user is likely to be confused and may just go ahead and accept the association.

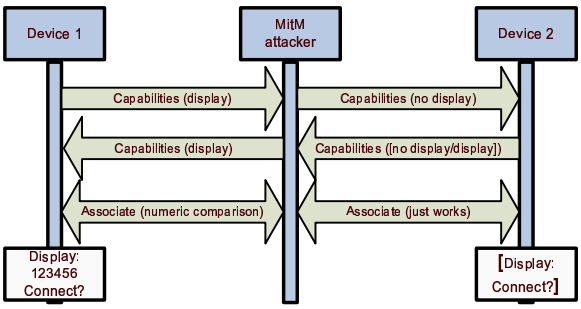

Fig. 11. Man-in-the-middle between Different Association Models

To get an idea about whether such user confusion is likely, Valkonen et al [38, 40] included the situation depicted in Figure 11 as a test scenario in one round of an on-going series of usability testing. Out of 40 test users, 6 accepted the pairing on both devices, 11 noticed the problem and rejected the pairing on both devices, and the rest rejected pairing on Device 1 but accepted it on Device 2.

This attack has two implications. Firstly, when the second device has a display, it is a bidding down attack against this device. The second device will know that the association is unauthenticated. However, the user may still allow the association to happen. Secondly, it is a bidding up attack against the first device since it believes that the association is made using a secure protocol resistant to

MitM attacks. Consequently, the first device may choose to trust this security association more than it would trust a 'Just Works' security association. For instance, it may have a policy rule, which allows more trustworthy devices to initiate connections without user confirmation.

A scenario related to the attack on Figure 11 arises with devices that are willing to participate in setting up a security association without immediate user conditioning. Public printers and access points are examples of devices that may be permanently conditioned for association. Suppose a user starts associating Device 1 with Device 2 using an association model that does not require any user dialog (e.g., WUSB cable model, or HomePlugAV Simple Connect mode) and that Device 2 is permanently conditioned to accept incoming association requests. If an attacker now initiates association with Device 2, say using Bluetooth SSP numeric comparison, a user dialog will pop up on Device 2. Since the user is in the middle of associating Device 1 and Device 2, he might answer the dialog thinking that it is a query about Device 1. Depending on the nature of the dialog, the attacker may end up gaining unintended privileges on Device 2.

Strengthening Devices Now we discuss some implementation guidelines that can help address the kind of attacks identified above. When a security association is stored persistently, information about its level of security should be stored as well. HomePlugAV already does this indirectly by using different keys with different association models. Furthermore, this security-level information should be used in deciding the level of trust granted to the peer device. For instance, devices associated using Bluetooth SSP 'Just Works' or HomePlugAV Simple Connect models should not be allowed to install or configure software, at least, without explicit authorization from the user. This precaution would help to prevent bidding down attacks. The man-in-the-middle attack between numeric comparison and unauthenticated protocols (Figure 11) could be addressed with two alternative strategies:

1. Bidding down the second device from using numeric comparison to the 'Just Works' model could be addressed by requiring that devices believing to be in 'Just Works' association would anyway show the checksum if they are able to do so. However, this solution does not prevent the bidding up attack against the first device.
2. Bidding down and bidding up attacks can both be countered by querying the user appropriately to confirm the I/O capabilities of the peer device. For instance, if the capability negotiation messages indicate that the peer device has no display, a device could ask the user if the peer device does indeed have a display. If the user gives answers affirmatively, it is an indication of a man-in-the-middle. However, such an additional dialogue is likely to impair usability.

7 Conclusions

The problem of designing ways to set up security associations in personal networks is a challenging one because it calls for balancing usability, security and cost. A number of innovative solutions have been proposed in recent research literature. Some of these have been incorporated into new standards for associating devices in personal networks. The objective of the new standards is to make the association process more user-friendly while improving the security at the same time without incurring significant cost penalties.

We surveyed various protocols in the research literature and association models used in different standards specifications. We presented a systematic classification of protocols for human-mediated establishment of session keys and provided formal analyses of some of them. We showed how the different protocols in standard specifications are related by using our classification.

The flexibility of the new proposals also introduce potential for some new attacks. We described some such threats. Careful design of user dialogs may reduce the likelihood of these attacks However, how exactly to design the user dialogs to preserve security without harming usability remains an open issue.

Devices implementing the new standards are beginning to be deployed. All of them provide better security than the old procedures they replace. However, how well they are accepted by users remains to be seen. Unauthenticated key agreement (as in the 'Just works' model of Bluetooth Secure Simple Pairing and the 'Pushbutton' model of WiFi Protected Setup) incur virtually no additional cost and optimal in usability. Therefore it may turn out to be more preferred and more widely deployed than authenticated key agreement. However, unauthenticated key agreement will not be sufficient for certain scenarios. One example is associating input devices (like keyboards and mice) with a computing device – a malicious input device can cause significant damage to the computing device. Another example is associating personal medical devices, or other similar contexts that may be subject to privacy regulation. Thus, the need for extremely inexpensive (and yet secure and usable) solutions for this problem remains. The type of approaches discussed in Section 4 such as in-band integrity channels and extracting secrets from the shared environments using existing sensors seem to be promising avenues to conduct further research.

Acknowledgments

This paper builds on three earlier papers we co-authored with various students. We thank the students who worked with us on those papers: Sven Laur [19, 20], Kristiina Karvonen and Ersin Uzun [39], Jani Suomalainen and Jukka Valkonen [38].

References

1. Dirk Balfanz et al. Talking to strangers: authentication in ad-hoc wireless networks. In *Proceedings of the Network and Distributed System Security Symposium*, 2002.

2. Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management - part 1: General (revised), 2006. `http://csrc.nist.gov/CryptoToolkit/kms/SP800-57Part1_6-30-06.pdf`.
3. Steven M. Bellovin and Michael Merrit. An attack on the interlock protocol when used for authentication. *IEEE Transaction on Information Theory*, 40(1):273–275, January 1994.
4. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, 1992.
5. Claude Castelluccia and Pars Mutaf. Shake Them Up! A movement-based pairing protocol for CPU-constrained devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 51–64, Seattle, Washington, 2005.
6. Dorothy E. Denning and Peter F. MacDoran. Location-Based Authentication: Grounding Cyberspace for Better Security. *Computer Fraud & Security*, 1996(2):12–16, February 1996. `http://www.cs.georgetown.edu/~denning/infosec/Grounding.txt`.
7. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.
8. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 1983.
9. Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proceedings of the 15th Usenix Security Symposium*, pages 167–178, 2006. `http://www.usenix.org/events/sec06/tech/franklin.html`.
10. Christian Gehrmann. Cryptanalysis of the Gemmel and Naor Multiround Authentication Protocol. In *Advances in Cryptology - CRYPTO '94*, number 839 in Lecture Notes in Computer Science. Springer-Verlag, 1994.
11. Christian Gehrmann, Chris Mitcell, and Kaisa Nyberg. Manual authentication for wireless devices. *RSA CryptoBytes*, 2004.
12. Christian Gehrmann and Kaisa Nyberg. Enhancements to Bluetooth baseband security. In *Proceedings of Nordsec 2001*, 2001.
13. Pete Gemmel. `www.cs.unm.edu/~gemmell/`.
14. Pete Gemmel and Moni Naor. Codes for interactive authentication. In *Advances in Cryptology - CRYPTO '93*, number 773 in Lecture Notes in Computer Science. Springer-Verlag, 1994.
15. Michael T. Goodrich, Michael Sirivianos, John Solis, Gene Tsudik, and Ersin Uzun. Loud and clear: Human-verifiable authentication based on audio. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, 2006.
16. Thomas S. Heydt-Benjamin, Dan V. Bailey, Kevin Fu, Ari Juels, and Tom OHare. Vulnerabilities in first-generation rfid-enabled credit cards. In *Proceedings of Eleventh International Conference on Financial Cryptography and Data Security*, Lowlands, Scarborough, Trinidad/Tobago, February 2007. `http://prisms.cs.umass.edu/~kevinfu/papers/RFID-CC-manuscript.pdf`.
17. Tero Kivinen and Markku Kojo. RFC3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE), May 2003. `http://www.ietf.org/rfc/rfc3526.txt`.
18. Jan-Ove Larsson. Higher layer key exchange techniques for Bluetooth security . Open Group Conference, Amsterdam October 24 , 2001.

19. Sven Laur, N. Asokan, and Kaisa Nyberg. Efficient Mutual Data Authentication Using Manually Authenticated Strings. Cryptology ePrint Archive, Report 2005/424, 2005.
20. Sven Laur and Kaisa Nyberg. Efficient mutual data authentication using manually authenticated strings. In D. Pointcheval, editor, *The 5th International Conference on Cryptology and Network Security, CANS 2006*, volume 4301 of *Lecture Notes in Computer Science*, pages 90–107, Suzhou, China, December 2006. Springer.
21. Information technology Security techniques Entity authentication Part 6: Mechanisms using manual data transfer . INTERNATIONAL STANDARD ISO/IEC 9798-6 , 2005.
22. Rene Mayrhofer. The candidate key protocol for generating secret shared keys from similar sensor data streams. In *Proceedings of the European Workshop on Security and Privacy in Adhoc and Sensor Networks (ESAS)*, pages –. Springer, 2007.
23. Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In *Proceedings of Pervasive 2007*, number 4480 in Lecture Notes in Computer Science, pages 144–161. Springer-Verlag, 2007.
24. Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, 2005.
25. David McGrew and John Viega. The use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH. IETF Request for Comments, RFC 4543, May 2006.
26. G.A. Miller. The magical number seven, plus or minus two: Some limtis on our capacity for processing information. *Psycological Review*, 63:81–97, 1956.
27. Moni Naor, Gil Segev, and Adam Smith. Tight bounds for unconditional authentication protocols in the manual channel and shared key models. In *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 214–231, 2006.
28. Richard Newman, Sherman Gavette, Larry Yonge, and Ross Anderson. Protecting domestic power-line communications. In *Proceedings of The Second Symposium on Usable Privacy and Security*, 2006.
29. NIST: National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. U.S. Department of Commerce, January 2000.
30. Sylvain Pasini and Serge Vaudenay. Sas-based authenticated key agreement. In *Proceedings of The 9th International Workshop on Theory and Practice in Public Key Cryptography*, 2006.
31. IST-SHAMAN Project. Final technical report - specification of a security architecture for distributed terminals. `http://www.isrc.rhul.ac.uk/shaman/docs/d13a2v1.pdf`, November 2002.
32. Kasper Bonne Rasmussen and Srdjan Čapkun. Implications of radio fingerprinting on the security of sensor networks. In *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks*, pages –. IEEE, 2007.
33. Ron Rivest and Adi Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–395, April 1984.
34. Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Secure device pairing based on a visual channel (short paper). In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 306–313, 2006.

35. Simple Pairing Whitepaper. Bluetooth Special Interest Group. http://www.bluetooth.com/Bluetooth/Apply/Technology/Research/Simple_Pairing.htm, 2006. Subsequently incorporated into Bluetooth 2.1 specifications http://www.bluetooth.com/Bluetooth/Learn/Technology/Core_Specification_v21__EDR.htm.
36. Claudio Soriente, Gene Tsudik, and Ersin Uzun. HAPADEP: Human Asisted Pure Audio Device Pairing. Technical report, Cryptology ePrint Archive, Report 2007/039, 2007.
37. Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, 1999.
38. Jani Suomalainen, Jukka Valkonen, and N. Asokan. Security associations in personal networks: A comparative analysis. In *Security and Privacy in Ad-hoc and Sensor Networks, 4th European Workshop, ESAS 2007, Cambridge, UK, July 2-3, 2007, Proceedings*, volume 4572 of *Lecture Notes in Computer Science*, pages 43–57. Springer, 2007.
39. Ersin Uzun, Kristiina Karvonen, and N. Asokan. Usability analysis of secure pairing methods. Technical Report NRC-TR-2007-002, Nokia Research Center, 2007.
40. Jukka Valkonen, Aleksi Toivonen, and Kristiina Karvonen. Usability Testing for Secure Device Pairing in Home Networks. In Anne Bajart, Henrik Muller, and Thomas Strang, editors, *UbiComp 2007 Workshop Proceedings, September 2007, Innsbruck, Austria*, 2007.
41. Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal de Lara. Amigo: Proximity-based authentication of mobile devices. In *Proc. Ninth International Conference on Ubiquitous Computing (UbiComp 2007)*, pages –, September 2007.
42. Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptology - CRYPTO 2005*, 2005.
43. Mario Čagalj, Jean-Pierre Hubaux, Srdjan Capkun, Ramkumar Rengaswamy, Ilias Tsigkogiannis, and Mani Srivastava. Integirty (I Codes: Message Integrity Protection and Authentication Over Insecure Channels. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 280–294, 2006.
44. Mario Čagalj, Srdjan Čapkun, and Jean-Pierre Hubaux. Key agreement in peer-to-peer wireless networks. In *Proceedings of the IEEE (Special Issue on Cryptography and Security)*, 2006.
45. M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.
46. Wi-Fi Alliance. Wi-Fi Protected Setup Specification. Wi-Fi Alliance Document available at http://www.wi-fi.org/wifi-protected-setup/, January 2007.
47. Wireless USB Specification. Association Models Supplement. Revision 1.0. USB Implementers Forum. Http://www.usb.org/developers/wusb/, 2006.
48. Philip R. Zimmermann. Pgpfone: Pretty good privacy phone owner's manual, version 1.0 beta 5, appendix c. http://web.mit.edu/network/pgpfone/manual/#PGP000057, January 1996.