Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey

L.H. Nguyen* and A.W. Roscoe
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
E-mail addresses: {`Long.Nguyen, Bill.Roscoe@comlab.ox.ac.uk`}

Abstract

One of the main challenges in pervasive computing is how we can establish secure communication over an untrusted high-bandwidth network without any initial knowledge or a Public Key Infrastructure. An approach studied by a number of researchers is building security though human work creating a low-bandwidth empirical (or authentication) channel where the transmitted information is authentic and cannot be faked or modified. In this paper, we give an analytical survey of authentication protocols of this type. We start with non-interactive authentication schemes, and then move on to analyse a number of strategies used to build interactive pair-wise and group protocols that minimise the human work relative to the amount of security obtained as well as optimising the computation processing. In studying these protocols, we will discover that their security is underlined by the idea of *commitment before knowledge*, which is refined by two protocol design principles introduced in this survey.

Keywords: authentic empirical channel, commitment schemes and digest functions.

1 Introduction

In this paper, we give a survey of authentication protocols which involve manual transfers of short authentication strings (SASs) over an assumed *empirical* or *authentication* channel as might be created by one or more human users of the systems being considered. The careful use of low-bandwidth unspoofable channels offers an interesting alternative solution for the problem of authentication, as opposed to making use of PKI and/or trusted third parties (TTP).

There have been rapid developments in this field in the last few years, resulting in publications, international standards and patent applications relating to a variety of such protocols. In the first few years these protocols were frequently introduced by groups working independently of each other. For example, Stajano and Anderson [47] were the first to attempt to form a secure network for a two-party scenario. The new approach was then studied and refined further by many researchers, most notably Balfanz et al. [3], Creese et al. [9, 10, 11, 12], Gehrmann et al. [13, 14, 15], Hoepman [17, 18], Vaudenay [53], Čagalj et al. [6], Wong and Stajano [56, 57] and Roscoe and Nguyen [32, 33, 37, 41, 42, 43, 44] who introduced both pairwise and group authentication protocols using less human interactions. Creese, Roscoe et al. [9, 10, 11, 12, 41] refer to these human exchanges

*Corresponding author: Telephone: 0044 (0)7821540971, Fax: 0044 (0)1865 273839, and alternative E-mail address: hn2503@gmail.com as empirical channels since the recipient has some empirically based knowledge about the origin of the message, as opposed to cryptographically based knowledge (e.g. via a PKI) of its origin. We therefore use this term throughout this survey.

Given the potential importance of this work, we feel that this survey is timely. We consider one-way protocols, non-interactive in the sense that all communication is one way, and interactive protocols that work both for pairwise interaction and group formation. We also classify and analyse these schemes in terms of information binding strategies and their computational efficiency. We note that there have been several other surveys written by Suomalainen et al. [50], Mashatan and Stinson [27], and Pasini and Vaudenay [39, 53]. However these authors only consider either pairwise protocols (bootstrapping security from human interaction or secret shared passwords) or one-way authentication schemes (both interactive and non-interactive). As we will see, they significantly differ from ours with respect to the scope and depth of analysis (in terms of security, design and efficiency). Due to the range of potential implementation technologies in this paper we largely abstract away the details that are not immediately important to security. We also have imagined there is a preliminary and insecure group/pairwise set-up protocol (implementation dependent) that is run either before or simultaneously with the first messages of the secure protocol to agree on, for example, the number and identities of protocol participants, since the information will significantly reduce the waiting time in a protocol session.

The development of this novel sort of authentication has arisen from many daily life applications. For example, in the authentication technology, for parties to agree on the same payment records in financial transactions, healthcare information in telemedicine, or cryptographic public keys, their portable devices exchange the data over (insecure) WiFi and then display a short and *non-secret* digest of the protocol's run that the devices' human owners *verbally* or *visually* compare to ensure they agree on the (public) data, i.e. the latter uses human interactions to prevent fraud such as identity theft. It is thus easier to implement the solutions, because they do not rely on the needs for PIN numbers, passwords or trusted third parties (e.g. the government or security infrastructures distribute ID certificates or private keys to users) which may be too complex and expensive to use on portable devices.

We explain the notation used in describing the protocols as well as a number of cryptographic primitives such as a commitment scheme, short/long-hash functions, and digest functions in Section 2. A simple model for the computation cost of these cryptographic primitives and an attack model are provided to assess the complexity and security of these protocols as we move along.

Although the authentication protocols considered here have been independently introduced by a number of research groups using different notation, this survey will demonstrate that their security is derived from the idea of *commitment before knowledge*, formally defined in Section 2.2.1. What it does is to force protocol participants to be (jointly) committed to some value before knowing what it is until they reveal their respective shares of the decommitment in a later stage of a protocol run. This committed value will, in turn, always be *instrumental* in the computation of the SASs[1] compared by humans, and therefore the parties' state of knowledge of the SASs is a uniform distribution, i.e. this is the key to defeating any causal influence such as birthday attacks as well as ensuring that search and multiple-shot attacks do not gain any advantage over one-shot and guess attacks. We will see that there are two different approaches of achieving this goal depending on whether the authenticated information is *directly* or *indirectly* bound to SASs, as studied in sections 4.2 and 4.3 respectively. In particular, the *direct* information binding strategy will be refined by two protocol design principles, termed **P1** and **P2**, in sections 4.3 and 5.1.

[1]The committed value could be either used directly as the SAS or inputted as a private key of a digest function, a universal hash function or a MAC function.

We start with a number of non-interactive one-way authentication schemes that use empirical channels in different ways, for example: MANA I proposed by Gehrmann, Mitchell and Nyberg [13, 14, 15]. We then see that the scheme neither optimises human effort nor offers as much security as had previously been believed. We offer an improved version that provides more security for half the empirical work, using a more general empirical channel.

In Section 4, we look at a variety of pairwise interactive authentication protocols. We see in order to optimise (i.e. minimise) the amount of human/empirical work done in a protocol, it is better to handle a single SAS rather than the several used by some protocols. Once the human work has been optimised, we turn our attention to minimising the computing power required for the protocols. This is likely to be important in practice because of potential applications in low-power pervasive computing devices.

While there has been much recent literature on pairwise protocols, we find it strange that apart from the authors' group [9, 32, 33, 41] and Valkonen et al. [52] there has been little on group protocols, although there appear to be many potential applications of these. We will discuss group protocols in Section 5 as well as presenting a number of newly invented and modified versions aiming to further improve the processing cost.

In Section 6.1, the computation cost of every protocol will be gathered into three tables that clearly demonstrate two things:

- Interactive schemes (pairwise or group authentication) can be much more efficient in human work than non-interactive ones (one-way authentication);

- Although the security of the majority of protocols rely on the *commitment before knowledge* idea, the use of *direct binding* to achieve this goal has a clear advantage in efficiency. This arises from the potential to use a digest function designed to produce only the small number of bits required for empirical comparison as opposed to a conventional cryptographic hash function used in *indirect binding.*

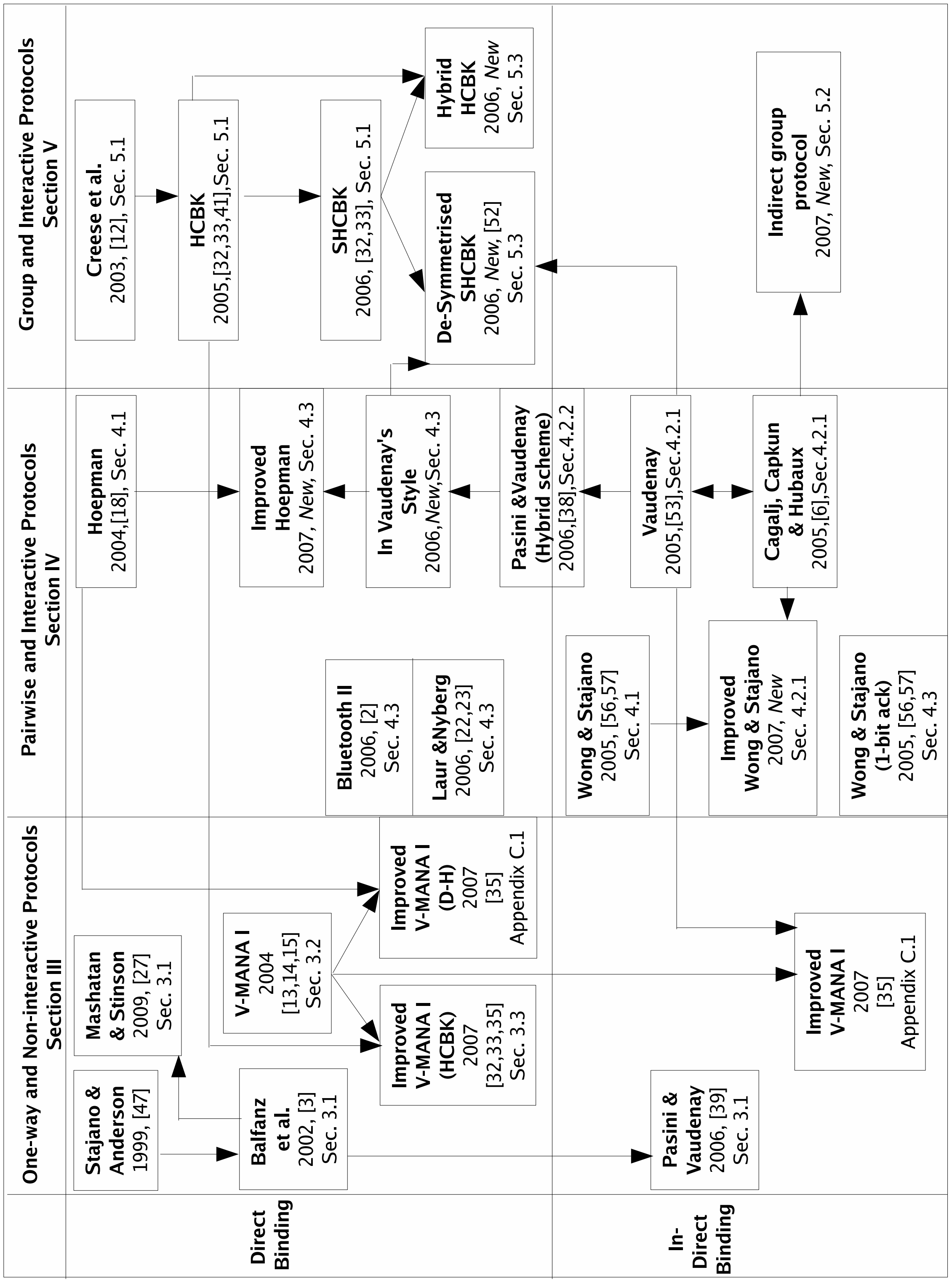
Many of the protocols described in this paper are taken from earlier literature. However we have shown how to make minor improvements to some and major improvements to others. We will use the following notation in protocol descriptions:

- Protocols equivalent to ones from previous literature, though perhaps in different notation, are just cited: [].

- Protocols that have been modified in minor ways, often by replacing one or more cryptographic primitive or other data operation, are cited []*.

- Protocols that are either major modifications to existing ones or just new are marked *New.*

The protocols in this paper are organised according to their aims (e.g. one-way, pair and group) and structure (direct versus indirect binding). This does not always make it easy to see the way the whole topic has been developed in recent years, frequently by several independent groups. Figure 1 shows all protocols both by year of publication, section number where the protocol is described in this paper and dependence on other work (by citation and directed arrows). For example, an arrow from protocol A to B indicates that the design of protocol B is influenced by A.



One-way and Non-interactive Protocols Section III
Pairwise and Interactive Protocols Section IV
Group and Interactive Protocols Section V
Direct Binding
In-Direct Binding
Stajano & Anderson 1999, [47]
Mashatan & Stinson 2009, [27] Sec. 3.1
Balfanz et al. 2002, [3] Sec. 3.1
V-MANA I 2004 [13,14,15] Sec. 3.2
Improved V-MANA I (D-H) 2007 [35] Appendix C.1
Improved V-MANA I (HCBK) 2007 [32,33,35] Sec. 3.3
Pasini & Vaudenay 2006, [39] Sec. 3.1
Improved V-MANA I 2007 [35] Appendix C.1
Hoepman 2004,[18], Sec. 4.1
Improved Hoepman 2007, New, Sec. 4.3
In Vaudenay's Style 2006,New,Sec. 4.3
Pasini &Vaudenay (Hybrid scheme) 2006,[38],Sec.4.2.2
Vaudenay 2005,[53],Sec.4.2.1
Cagalj, Capkun & Hubaux 2005,[6],Sec.4.2.1
Bluetooth II 2006, [2] Sec. 4.3
Laur &Nyberg 2006, [22,23] Sec. 4.3
Wong & Stajano 2005, [56,57] Sec. 4.1
Improved Wong & Stajano 2007, New Sec. 4.2.1
Wong & Stajano (1-bit ack) 2005, [56,57] Sec. 4.3
Creese et al. 2003, [12], Sec. 5.1
HCBK 2005,[32,33,41],Sec. 5.1
SHCBK 2006, [32,33], Sec. 5.1
Hybrid HCBK 2006, New Sec. 5.3
De-Symmetrised SHCBK 2006, New, [52] Sec. 5.3
Indirect group protocol 2007, New, Sec. 5.2

Figure 1: Summary of all protocols.

This paper, particularly as it covers the differing approaches of several research groups and since it sits at the boundary of cryptography and protocol design, requires a lot of (new) notation and definitions. The reader might either choose to study all of this notation in advance, or can read it to the end of Section 2.1 and then refer back to sections 2.2, 2.3 and 2.4 as needed, i.e. the reader might go straight to (V-)MANA I protocols and its improved versions in Section 3.2 which give a concrete goal for the best protocol to achieve.

2 Notation and basic definitions

Parties are identified using capital letters, e.g. A, B, L, S. The intruder or adversary is denoted I, which in most cases is assumed to try to impersonate a node, such as A, to talk to others or to intercept messages sent to A. $\forall A$ (or A') means that a message is sent or received by all parties in a group $\mathbf{G}$ attempting to bootstrap a secure communication between them. In common with much of the literature we are citing, the combination of two pieces of data will frequently be written $x \parallel y$. This will be synonymous with the ordered pair (x, y).

We will assume each node A in a group $\mathbf{G}$ of N parties has some information $INFO_A$ of length K bits ($K/w = K/32 = M$ words, here we assume that a word consists of $w = 32$ bits) that it wants to have authenticated to other members of the group; this might include:

1. Name and addressing information;
2. Its uncertificated public key or Diffie-Hellman token g^{x_A}, this might be a long-term object or generated freshly for the present protocol run;
3. Contextual information to help identify it, such as its location or human owner, or the owner's photograph, video and audio;
4. Information (perhaps certificated) relating to its functionality.

Nothing in this information should be secret since all the protocols we consider will make it public. $INFO_A$ might be attached to A permanently or for the long term; alternately some of it might be relevant to this particular run only. The goals of the protocols will always consist of authenticating pairs $(A, INFO_A)$ as members of the network. We make the identity A explicit here, and in the protocols using it, since the identity is vital to an understanding of who is in the group. In practice, as indicated above, A will normally just be embedded in $INFO_A$. In particular, we assume in these calculations that the name appears in the K bits referred to here. In addition, we refer to $INFOS$ as the concatenation in alphabet order of all the distinct pairs parties want to authenticate: NM words if they are all size M.

If $INFOS$ contains N photographs or similar, it may well be of significant size.

2.1 Communication links

In some cases, the protocols we quote from other papers are changed in appearance because we seek to use a consistent nomenclature and notation: we do not want a single piece of notation to have inconsistent meanings. One respect in which previous work varies is in the assumptions made about the empirical or authentic channels. In this paper, we use different notation for communications over a normal Dolev-Yao (insecure) channel and those over four types of empirical channels, which are presented in a descending order of generality.

These empirical channels provide authenticity and data integrity, but not confidentiality: it is assumed that there is enough direct familiarity or physical presence to ensure that the responsibility of a person for a short message can be immediately ascertained.

- $\longrightarrow_N$ represents the normal Dolev-Yao network, where all messages transmitted between devices using such a channel can be overheard, deleted or modified by the intruder. Examples of this channel are the Internet, WiFi, or a local network.

- $\longrightarrow_{WE}$, the *weak empirical* channel, cannot be forged, but it can be blocked, overheard, delayed or replayed. This is the weaker of the two forms of empirical channel described in [39, 53]. Typical examples of such channels include telephone conversation, voice mail or messages, where messages can be delayed, blocked or replayed, but cannot be forged by the intruder.

- $\longrightarrow_{BE}^t$ is similar to $\longrightarrow_{WE}$ except that messages cannot take more than time t to arrive and cannot be replayed. In other words, no empirical message can be accepted more than t time units after it was sent. Such a channel might be implemented over a reliable medium with a known bound on transmission, or over an unreliable one with the addition of some sort of time-stamp. The latter might make sense if the empirical message is sent by some video means, but otherwise would add significantly to the communication burden. We will call this a *bounded delay empirical channel*.

- $\longrightarrow_E$ is the empirical channel assumed in [32, 33]. Messages transmitted over such a channel cannot be mistaken or delayed from one to another session. To some extends, this implies that $\longrightarrow_E$ is a special case of $\longrightarrow_{BE}^t$ where t is chosen to be some lower bound on the length of time between one session and a later one. This is the type we use most often. Sometimes the two-way arrow $\longleftrightarrow_E$ is used to indicate (possibly) the same message is transmitted in both directions.

  An example of such a channel is provided by manual data transfers [13, 14, 15], i.e. human users manually copy data from one to another device via their in/output interfaces such as screen, monitor and keyboard. In this case, empirical messages cannot be mistaken or delayed from one to another session because: (1) the humans involved are not away at any time during a protocol run, and (2) each device normally only has one in/output interface for displaying or reading data.

- $\longrightarrow_{SE}$ is similar to a normal empirical channel, but it also provides stall-free transmission. As a result, a message transmitted over the channel cannot be delayed, removed or blocked by the intruder. This implies that $\longrightarrow_{SE}$ is the same as $\longrightarrow_{BE}^t$ where $t \approx 0$. We term this a *strong empirical channel* (or a strong authentication channel). This was also defined in [39, 53]. Face to face human conversation is an example of this channel.

2.2 Cryptographic primitives

We will be using a variety of cryptographic primitives which are related to hashing. Since bit-lengths of hash functions vary widely, for clarity, they will be classified relative to three common security properties of a cryptographic hash function, which are inversion-resistance, collision-resistance and second-preimage-resistance. Sometimes they will be subscripted with the number of output bits. For example:

- *longhash*(X) and $hash_B(X)$ refer to standard cryptographic hash functions [30] which are assumed to be inversion-resistant, collision-resistant and second-preimage resistant. In common with the literature, we will generally assume that they have at least $B = 160$ bits.

- *hash*(X) and $hash_{B/2}(X)$ refer to hash functions which are only assumed to be inversion-resistant and second-preimage resistant. Note that *hash*(X) is not required to resist collision attacks since its output length is $B/2$ or around 80 bits.

- *shorthash*(X) or $hash_b(X)$ will be functions with sufficiently many bits to offer weak or short-term versions of these security properties of *hash*(X) or $hash_{B/2}(X)$. Here b which is the bit-length of *shorthash*() is in the range of [16,32].

- *digest*(k, X) will be a short-output universal hash function or a digest function of X keyed by k – the specification and purpose of this function will be discussed at length in Section 2.2.2 as well as an additional property often required of the short output hash functions.

- $h_k(X)$ will be a universal hash function of X keyed by k.

Usually implemented using hash functions, we will also use a commitment scheme, whose definition is given below.

2.2.1 Commitment scheme and Commitment before knowledge

Definition 1 A probabilistic commitment scheme, following Vaudenay [53], consists of two mappings:

- *commit*: $\{0,1\}^K \times \{0,1\}^b \rightarrow \{0,1\}^B \times \{0,1\}^B$
  This mapping takes a public K-bit data $INFO$, a private b-bit random nonce R (e.g. $b = 16$), and then internally generates a $(B-b)$-bit random nonce and produces two B-bit strings (e.g. $B = 160$): a commit value c and a decommit value d. This algorithm is nondeterministic since it uses an internally generated random element of $B - b$ bits.

- *open*: $\{0,1\}^K \times \{0,1\}^B \times \{0,1\}^B \rightarrow \{0,1\} \times \{0,1\}^b$
  This mapping takes a public K-bit data $INFO$, a commit value c and a decommit value d, and produces an error or success signal together with a b-bit random nonce R. This algorithm is deterministic, and has the property that whenever there exists an R such that (c, d) is a possible output for *commit*$(INFO, R)$, then *open*$(INFO, c, d)$ yields R.

The following provides more information about commitment schemes as well as the idea of *commitment before knowledge* and its restricted version called *joint* commitment before knowledge, which underlie the security of nearly every protocol discussed in this paper.

A commitment scheme has the following two properties:

- (ϵ_h, T_h)-*hiding* [53]: no algorithm or adversary I bounded by a time complexity T_h can win the following game by interacting with a challenger C with a probability higher than ϵ_h.

  1. I selects a message $INFO$ and sends $INFO$ to C.
  2. C picks a b-bit random nonce R, runs commit on $(INFO, R)$, gets (c, d), and sends c to I.
  3. I yields R' and wins if $R' = R$.

When $\epsilon_h = 2^{-b}$ and $T_h = +\infty$ we say that the scheme is perfectly hiding, and this is what we assume in all of the uses of a commitment scheme in this paper.

- (ϵ_b, T_b)-*binding*:[2] no algorithm or adversary I bounded by a time complexity T_b can win the following game with a probability higher than ϵ_b.

  1. I selects a message $INFO$ and a b-bit random nonce R.
  2. I then runs commit on $(INFO, R)$ and gets (c, d).
  3. I later computes $(INFO', d')$, where $INFO' \neq INFO$ but d' can be either equal to or different from d, and wins if $(INFO', c, d')$ opens to R.

  When $\epsilon_b = 2^{-b}$ and $T_b = +\infty$ we say that the scheme is perfectly binding, and this is what we assume in all of the uses of a commitment scheme in this paper.

The bit-length of R will be short, e.g. $b \in [16, 20]$ or even $b = 0$, in all the protocols considered. When the random nonce R is null as in the one-way and non-interactive scheme of Pasini-Vaudenay of Section 3.1, we drop R in the notation for a commitment scheme, i.e. we write $commit(INFO)$.

To prevent brute-force search and birthday attacks, the commitment scheme (i.e. $commit()$) needs to extend R by a randomly chosen secret nonce R' of $B - b$ bits so that the combination $R \parallel R'$ has the same bit-length as the output of a cryptographic hash function. This implies that a commitment scheme is designed to be as secure as a standard cryptographic hash function $hash_{160}()$ or $longhash()$. In practice, a commitment scheme is usually built from a pseudorandom function such as a hash function, and therefore they have the same computational complexity. This issue will be discussed in more detail in Section 2.4.2.

The above security and efficiency properties of a commitment scheme can be demonstrated by the following construction which was introduced by Pass [40].

- **Committing**: to bind some public information $INFO$ and a short random secret key R of b bits together, the algorithm picks another secret random nonce $R' \in_{random} \{0,1\}^{B-b}$. It then sets $d = R \parallel R'$ and $c = longhash(INFO \parallel d)$, i.e. $commit(INFO, R) = c \parallel d$. The committer then publishes the commitment c.

- **Decommitting**: to decommit or to use $open(INFO, c, d) = R$, the committer first publishes the decommitment d. Anyone can extract R from d (i.e. the first b bits), verify whether $c = longhash(INFO \parallel d)$, and then output success or failure.

Instead of using a cryptographic hash function $longhash()$ as seen in the above construction, a universal hash function as mentioned in Section 2.2 can be used, i.e. $c = longhash(INFO \parallel d)$ is replaced with $c = h_d(INFO)$. In such a case a similar change needs to be made to decommitting.

The conventional understanding of a commitment scheme has been that the committer knows the value to which he or she is committed. In this paper however we will see several uses of the commitment idea in which the protocol participants, which can be either the committer or other parties, do not know the committed value. We will see this can be done by distinguishing carefully between when nodes are committed without knowledge to a value and when they know it.

The first use of this idea is called *commitment before knowledge*, which directly influences the design of the HCBK protocol of Section 5.1, and some of the one-way and non-interactive protocols in Section 3.

[2] We note that there is a lack of explicitness in the binding property of the commitment scheme defined by Vaudenay [53], since the security specification there fails to bind it to the input message, as was obviously intended. The definition of $commit(INFO, R)$ there is satisfied by defining the commitment $c = longhash(N, r)$, where N is a long random nonce internally generated by the committer, and the decommitment $d = (N, r)$.

> **Commitment before knowledge**:[3] Suppose that, in a partly complete protocol session, a participant A has sent or received parameters, such that in all successful completions of this session some term d has the same value, such as for instance $d = digest(k, INFOS)$. Then A is committed to a value of d at this point in the session.
>
> A knows the value of some term d at a point in a partly complete protocol session if, from values A has received, A can compute d without having to invert hash functions or decrypt messages to which A does not hold the decryption key.
>
> A is committed to the value of d before knowing it if the earliest point at which A is committed to a value of d properly precedes the earliest point at which it knows the value of d.

The same properties also apply to a more restrictive use of the idea, called *joint* commitment before knowledge which influences the design of nearly every interactive protocol, i.e. pairwise and group authentication schemes in sections 4 and 5.

> **Joint commitment before knowledge**: A protocol ensures *joint commitment before knowledge* for a set of participants if in every sufficiently long partial execution of the protocol, there is a point at which each of those participants is committed to a value for a term d, but does not yet know the value of d, and moreover in every successful completion of this partial execution, the participants are committed to the same value for d.

One simple way to achieve *joint commitment before knowledge* is as follows: each node is committed to some value by publishing its commitment, for example, by using the commitment scheme described above or a cryptographic hash function. The jointly committed value (i.e. SAS) is then the output of some function, such as summation, exclusive-or, Diffie-Hellman key agreement or *digest* functions, applying to all of the values to which every node has been committed.

We will see later that there are two different strategies for achieving (joint) commitment before knowledge, i.e. *direct* and *indirect* information binding approaches which are formally introduced in Section 4. In addition, the combination of commitment before knowledge and the *direct binding* approach will be refined by the principle **P2** to design several group authentication protocols in Section 5.1.

2.2.2 Short-output hash functions and digest functions

A normal cryptographic hash function is chosen so that it has enough bits to be essentially immune to searching such as the birthday attacks. In this paper, we will see various uses of new functions that, like cryptographic hash functions, are intended to randomise and convey no useful information about the preimage. However, their outputs are significantly shorter, since they will always produce values of short authentication strings (SASs) transmitted over the empirical channels which can be very limited in bandwidth.

We will see two variants on this idea: the simpler is what we call a shorthash function: $shorthash()$. This has a single argument, and is intended to be uniformly or near-uniformly distributed over its b-bit range as its argument varies. Since the hash output is too short, it is impossible to have properties such as collision and inversion resistances like in cryptographic hash functions. What is required instead is the *strict avalanche criterion*, which states that any number of bit-changes in the input has an equal (but non-negligible) influence on every bit of the output [55].

[3] We are grateful to an anonymous referee for wording of this and the idea of *joint commitment before knowledge*, which improves the versions in earlier drafts of this paper.

In some of the protocols we consider, we need to construct a b-bit digest of $INFOS$ and some key k. Like $shorthash()$, digest functions cannot have the usually specified properties of a cryptographic hash function, namely non-invertability and collision-freeness. On the other hand, we do require that a high degree of randomness arises from the use of the key k, as set out below and in [32, 33].

Definition 2 [32, 33] A b-bit *digest* function: $digest : K \times M \to Y$ where K, M and $Y = \{0...(2^b - 1)\}$ are the set of all keys, input messages and digest outputs, and moreover:

- for every $m \in M$ and $y \in Y$, $\Pr_{\{k \in K\}}[digest(k, m) = y] = 2^{-b}$
- for every $m, m' \in M$ $(m \neq m')$ and $\theta \in R$: $\Pr_{\{k \in K\}}[digest(k, m) = digest(k \oplus \theta, m')] \leq 2^{-b}$

The rationale for these two specifications, especially the use of "$\oplus\ \theta$", will become apparent when we analyse group protocols such as the SHCBK protocol in Section 5.1. The digest function specification is similar to *universal* hash functions, except the probability of digest collisions relative to different keys is also considered, i.e. the way digest keys are agreed between nodes in the SHCBK protocol can be manipulated such that different nodes' keys may be relatively shifted by a θ known to the intruder. The inclusion of the θ shift is therefore to ensure that this type of activity can never benefit the intruder.

Although both $shorthash()$ and $digest()$ are less secure than cryptographic hash functions, they are potentially faster to compute, thanks to their short outputs. More details about their comparative speed performance can be found in Section 2.4.2. Designing (universal) hash functions has an exciting and long history in computer science and cryptography, however there does not seems to be much literature on the study and exploitation of short output to increase computation efficiency. Thus we believe that there is a potential of new constructions for digest functions or adaptation of existing work to acquire computation efficiency, as described below.

As we will see in the majority of direct binding protocols presented in this paper, the SAS is often the output of a short-output function such as a digest function. For this reason, there have been a number of algorithms proposed to compute short digest functions [2, 14, 15, 22, 23, 33, 38]. For example, [32, 33] introduced some constructions which exploit the short output of a digest function to increase computational efficiency. These are adapted from a well-studied universal hash function construction of Mansour et al. [26] and Krawczyk [20, 21]. The most efficient are based on Toeplitz matrices of bits or words which are generated out of ϵ-biased distribution sequences of random bits. In practice, these sequences of bits can be produced by a pseudorandom number generator, e.g. linear feedback shift registers, as pointed out in [21, 33]. These algorithms have no restriction on the length of the object (typically $INFOS$) being digested.

In contrast, other researchers [14, 15, 22, 38] make use of universal hash functions [5, 19, 49] to compute digest functions. Their algorithms put an upper bound on the input length, and consequently they have to compress a long input message into a fixed number of bits (say 512 or 256 bits) by using a cryptographic hash function prior to running the algorithms themselves. This strategy turns out to be neither ideally secure nor cost effective [33]. Alternatively, others [2, 38] suggest using the first b bits of a cryptographic hash value of a large input message, which is potentially inefficient and does not necessarily have the precise property we need of being a digest function.

2.3 Attack model

In Section 2, we have defined the intruder's power over data transmitted over all kinds of channels used in the family of protocols. In addition to that, the following are definitions of attacks performed by the intruder.

- **A general attack**: uses (off-line) combinatorial search, e.g. using the birthday paradox to search for hash collisions. This can be either interactive or non-interactive. The attack may consist of *multiple protocol runs*, and so this is also referred to as *multiple-shot* or *q-shot* attack, where q is the number of protocol runs involved.
- **A one-shot attack**: is a special case of a general attack, i.e. this only involves a single protocol run.

We will also use the term *combinatorial search* to refer to attacks, whether general or one-shot, which involve combinatorial search, i.e. this is the opposite of a *guess attack*.

Our aim is to ensure that general or multiple-shot attacks give the intruder no advantage over a one-shot or guess attack on this family of protocols, i.e. this is similar to the goal of password-based authentication protocols which have been studied extensively to date, e.g. [4]. This goal is achieved because once every protocol participant is (*jointly*) *committed before knowledge* to some short authentication strings (SASs, e.g. a digest value transmitted over empirical channels), then there is not any effective way in which the agents can determine anything about the value – the agents' state of knowledge of the SAS is a uniform distribution.

Moreover, in the majority of protocols considered in this paper, SASs are transmitted over (strong) empirical channels ($\longrightarrow_E$ and $\longrightarrow_{SE}$), and so cannot be mistaken or delayed from one to another session. Hence, the SASs in all protocol runs are themselves independent[4] as pointed out by Vaudenay and other authors [24, 35, 39, 53], and for any q we have:

$$\mathbf{Pr}(\text{a successful } q\text{-shot attack}) \leq q \times \mathbf{Pr}(\text{a successful one-shot attack})$$

We will see later in Appendix C (Theorem 1) how to formalise the above security argument to give proofs of security for various protocols introduced in this paper. We note that this security model is rather conservative because it is only valid when the intruder launch attacks on many (or perhaps q) *different* pairs or groups of parties. In practice, once a human has noticed a short authentication string disagreement, he or she will be suspicious or aware that an attack is taking place provided implementation is reliably constructed. This will mean that the human will either allow no more attempts or require longer authentication strings, i.e. extending the SAS by 1 bit after each mismatch makes the probability of a successful general attack be upper bounded by $2^{b-1} = \sum_{l=b}^{\infty} 2^{-l}$, where 2^{-l} is the likelihood of a successful one-shot attack on an optimal implementation of a l-bit SAS protocol, i.e. we will formally define optimality in human interactions of this type of protocols in Section 2.4.1.

Since the protocol design can reduce the probability of a successful attack to the chance of a one-shot attack, for simplicity we will refer successful attacks considered in all protocols to attacks that only involve a single protocol run, i.e. a one-shot attack. In addition, separate security analysis will be provided whenever protocols use weak empirical channels $\longrightarrow_{WE}$ to transmit SASs, which can be stalled and then replayed in other protocol runs.

We are also interested in *chosen plain-text attacks* [46] under which the intruder can influence data trustworthy parties want to authenticate. Although this attack might seem unrealistic, it is desirable that protocols are immune to it, i.e. it will become useful when we analyse protocol of Balfanz et al. of Section 3. Since the attack relies on combinatorial manipulation, we refer to it as a special case of the combinatorial search attack.

[4] A bit string X is independent of a bit string Y if for all random variable X of value x and for all random variable Y of value y: $\mathbf{Pr}[X = x] = \mathbf{Pr}[X = x \mid Y = y]$.

In authentication protocols where parties only want to authenticate their public-key-like information, there is no need to distinguish between honest and dishonest nodes. Conversely, every one has to be trustworthy in a key agreement protocol, whether it is pairwise or group schemes. More discussion about this issue could be found at the begin of Section 5.

2.4 Cost model

It seems reasonable to measure the efficiency of the family of protocols in two ways: the amount of empirical or human effort required to complete them; and the amount of processing required at the nodes. The following models for human effort and computation cost are adapted from [32, 33].

2.4.1 Human effort

Our main measure of empirical work is the number of bits of the short authentication strings that are transmitted over the empirical channels. Throughout this paper, we always attempt to optimise the amount of security one can obtain from a given amount of empirical (human) communication. The following definition which specifies when a protocol in this area optimises human interactions relative to a level of security obtained has been justified in the papers of the authors [32, 33, 35] and Vaudenay [53].

> **Optimality of human interactions** [32, 33, 35, 53]: A protocol using short authentication strings (SASs) is said to be optimal in human interactions iff there is only a single b-bit SAS that needs to be empirically communicated, and the probability of a successful one-shot attack is bounded above by 2^{-b}.

We will see in later sections that this bound is attainable, provided we can discount the probability of strong cryptographic primitives being broken.

2.4.2 Computation cost model of cryptographic primitives

It is essential to optimise the human work in the families of protocols, but at the same time, we also want to minimise the computational cost. We are aware that the cost of agreeing a private key through exponentiation (in Diffie-Hellman's style) or public key cryptography always overtakes the cost of bootstrapping authenticity. However, if the authentication phase is carried out early on lightweight devices prior to key agreement achieved on more powerful devices at a much later stage, then it is desirable to minimise the computation cost of the authentication protocols done on lightweight devices, whose computation power can be very limited.

In order to assess the complexity of protocols, we have to have a model of the complexity of computing cryptographic primitives, such as a cryptographic hash function $longhash()$, a shorthash function $shorthash()$, a digest function, and a commitment scheme.

Let B and W be the number of bits and respectively words required to hold a longhash value. It is normal that B =160 bits, so we assume $W = B/w = 5$, here we assume that a word consists of $w = 32$ bits. Many researchers [13, 33, 53, 57] suggest 15 or 16 bits are reasonable choices for b, the width of the digest output, which is rounded up to 1 word in our analysis. We assume that nonces, keys (used in a commitment scheme and as input of $longhash()$) and other strong cryptographic values, such as a commitment c and a decommitment d, have the same bit-length B and therefore word-length W.

For simplicity, we only look at the Merkle-Damgård construction based hash functions [30] (i.e. block cipher based and customised functions such as SHA-1, MD5, or Davies-Meyer, Matyas-Meyer-Oseas and Miyaguchi-Preneel) because they are provably secure given that a one-way and

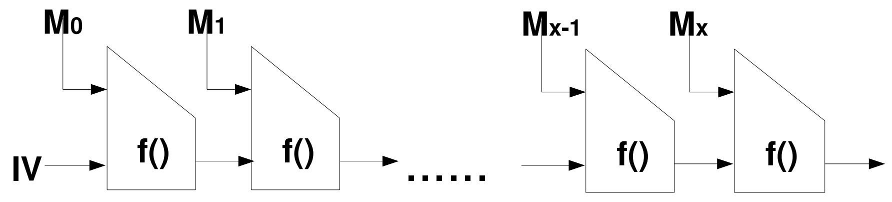


Figure 2: The Merkle-Damgård structure of conventional long-output cryptographic hash functions. Here $f()$ is a one-way and collision-resistant compression function, IV is some public initial value, and the input message m is partitioned into multiple fixed-size blocks $M_0, ..., M_x$, which are processed sequentially.

collision-resistant compression function exists, and also this family of hash functions is widely used in practice.

It is clear that the cost of computing the b-bit output $hash_b(m)$ whose design follows the sequential Merkle-Damgård structure as seen in Figure 2 increases linearly with the length of m, i.e. the majority of customised cryptographic hash functions in the literature are computed by calling a "compression" function $f()$ once for every (512-bit) block in sequence.

It also seems clear from Figure 2 that the cost will increase at least linearly with the complexity of the compression function $f()$ which can be quantified by its output length b. Considering the type of operation of the Merkle-Damgård structure based hash functions reveals that it always has an *internal state* whose bit-length is equal to or greater than the output bit-length. The internal state is updated by linear or bitwise operators (e.g. Shifting, AND, OR, XOR and rotation) in each loop of the algorithms to ensure that there is a computation between each message input bit and each bit of the internal state, i.e. the strict avalanche criterion. This implies that the computational cost is proportional to the size of the internal state, and a simple cost model of a b-bit hash function, which we will adopt in this paper, might be:

$$\text{Cost}(hash_b(m)) \approx b \times length(m)$$

Since well-known hash algorithms tend to be fixed width and vary significantly in their individual costs, it is hard to be too definite about this rule. The computational cost model does not take into account the number of clock cycles and implementation-specific (i.e. software or hardware), however it does give an approximate comparison between the cost of computing long and (very) short output functions, for example, cryptographic hash function versus digest function as can be illustrated in Table 1. To illustrate the validity of this cost model in practice, one can consider a variable-length hash function based on the idea of Key Derivation Function (KDF) or Mask Generation Function. For example, given a 160-bit output hash function such as SHA-1 or MD5, we can use the concatenation operator to construct a $(160 \times t)$-bit hash function as follows: $HASH(m) = hash(1, m) \parallel hash(2, m) \parallel \ldots \parallel hash(t, m)$. This of course clearly follows the above computational cost model.

With respect to the cost of computing a digest function. As defined in Section 2.2.2, $digest(\_,\_)$ is a family of shorthash functions indexed by a key k. Even though the key bit-length might be significantly longer than the hash output in several constructions of universal hash functions invented to date [20, 21], it normally does not play any significant part in the computation. Consequently, key length will not have a big impact on the computation cost. In fact the longer is the key, the

Cryptographic primitive	Computation cost
$longhash(INFO)$ or $hash_{160}(INFO)$	WM
$longhash(k)$ or $hash_{160}(k)$	W^2
$hash(INFO)$ or $hash_{80}(INFO)$	$WM/2$
$shorthash(INFO)$ or $hash_b(INFO)$	M
$digest(INFO)$	M
$commit(INFO, R)$	WM
$open(INFO, c, d)$	WM

Table 1: In the table, all calculations refer to the cost of computing functions that apply to either a single $INFO$ of M words or a key k of W words.

fewer the number of random bits we have to generate in our digest function constructions [32, 33, 36] as well as universal hash function constructions of Krawczyk [20, 21], and subsequently the better. Hence, we assume the cost model of a digest function or a universal hash function is similar to a hash function, which is mainly dependent on the lengths of the input message and the digest output.

A commitment scheme defined in Section 2.2.1 inputs a message of length K bits or $\lceil K/w \rceil = M$ words and a pair of nonces (R, R') that add up to $B = 160$ bits or W words. Since the pair of nonces play the same index role as key k in digest computation, we assume that a commitment scheme takes M words as input. These are true in both operations used to calculate a commitment and open/verify the commitment. We therefore conclude that the computational costs of computing and verifying a commitment are equal to each other as well as being equal to the cost of computing hash functions. The latter is true because commitment schemes are normally built from pseudorandom functions such as hash functions [40]. In practice, a commitment scheme, such as the one introduced in Section 2.2.1, takes both M-word messages and a 160-bit random nonce as its inputs, i.e. these are concatenated before being inputted to a cryptographic hash function: $longhash(INFO||R)$. However, it has been noted that $longhash(INFO||R)$ could be replaced by a universal hash function keyed by R, i.e. $h_R(INFO)$, to reduce the input length, and therefore the computational cost. As a result, to give a fair comparative analysis which is independent of implementation, we will stick to the above assumption.

Table 1 summarises the computational cost of all cryptographic primitives introduced in this section. While the table might suggest that the cost *equal* this product, what we are actually doing is ignoring the multiplicative constant because *all* the computational costs come from the same model.

Every protocol in this paper, except the first one in Section 5, only uses long or short hash functions, commitment schemes and/or digest functions. For this reason, we shall apply the simple model to compute the cost for each of them as we move along. In Section 6.1, all of the computational costs and human effort will be put into tables summarising the efficiency of each class of protocols.

3 Non-interactive protocols

We examine some protocols attempting to transmit a, possibly very long, message from one party to another efficiently in such a way that the origin and integrity of the message are authenticated.

These all use just one-way communication and authentication strings and help to illustrate the power of authenticated empirical channels.

To set this work in context, recall the classic (non-interactive) signature mechanism which works where there is a PKI. Here, a message $INFO_A$ of the sender A is accompanied by the *signature* $\{longhash(INFO_A)\}_{sk(A)}$. The receiver knows $INFO_A$ really is from A, since he can form the cryptographic hash of $INFO_A$ and discover if it really was A who signed this value with her secret key $sk(A)$. Although the whole of such a message may be assumed to be sent over a standard Dolev-Yao channel, there is in fact a closer tie-in with the subject matter of this section than there might appear to be. For public key encryption and decryption are computationally expensive, there is a strong incentive to keep the bandwidth of information transmitted under this form of cipher to a minimum. We might therefore regard a signature as the combination of a large message $INFO_A$ over an insecure channel with the smaller one $longhash(INFO_A)$ over an authenticated one.

Since in many cases the empirical channels are human mediated, the chief difference from this view of signature will be that our empirical channels are much lower bandwidth.

3.1 Long authentication string over the empirical channel

The above analysis of the use of signatures shows they are closely analogous to the following one-way authentication protocol devised by Balfanz et al. [3]. In this scheme, A wants to authenticate its information $INFO_A$ to B. In the original protocol [3], there is no restriction on the order of sending and receiving messages 1 and 2.

Balfanz et al. non-interactive protocol, [3]
1. $A \quad \longrightarrow_N B : A, INFO_A$
2. $A \quad \longrightarrow_{WE} B : longhash(A, INFO_A)$
B verifies the longhash.
Computational cost: $WM = 5M$

Balfanz et al. [3] did not specify the length of the hash function used. The main issue we have to decide in analysing this protocol is whether $INFO_A$ might have been manipulated by an attacker. This could be done, for example, by A accepting some piece of externally generated data, such as images and videos, to include in $INFO_A$. In several implementations, such as the specific one anticipated by Balfanz et al. [3] where $INFO_A$ is simply the public key, this attack may be impossible and here an $B/2 = 80$-bit hash function suffices. However, we wish to quote a protocol that is secure in general and, following the below analysis taken from [27, 39, 53], we assume that the hash length is $B = 160$ bits. When an intruder can influence some part of $INFO_A$, the intruder can (off-line) search for a different pair $(INFO_A, INFO'_A)$ both yielding the same hash value. $INFO_A$ is then given to A in the information gathering stage, and the intruder sends $INFO'_A$ to B masquerading as A.

$$\begin{array}{llll}
1. & A & \longrightarrow_N I(B) & : A, INFO_A \\
 & I(A) & \longrightarrow_N B & : A, INFO'_A \\
2. & A & \longrightarrow_{WE} B & : longhash(A, INFO_A)
\end{array}$$

However, this is something which deems infeasible as it must take about $2^{B/2} = 2^{160/2} = 2^{80}$ computation steps on average to find such a cryptographic hash collision, due to the birthday paradox. What this implies is that even though the protocol is secure, it is not optimal in the human work since B or 160 empirical bits only deliver $2^{B/2} = 2^{80}$ security level.

As a result of a single longhash whose input and output lengths are M and W words, the computation cost is of order $WM = 5M$, thanks to the cost model in Section 2.4.2.

In order to improve the number of authenticated bits, Pasini and Vaudenay [39] make use of a probabilistic commitment scheme to commit to the authenticated information. We note that there is no random nonce inputted into the commitment scheme used here, the commitment scheme therefore has to generate a new long nonce (80 bits) every time the *commit*() function is called. This is also the only time when an $(B/2 = 80)$-bit commitment scheme is used. For all other employments of a commitment scheme, it is always $(B = 160)$-bit. The $(B/2 = 80)$-bit hash of the commitment is then sent over the weak empirical channel.

Pasini-Vaudenay non-interactive protocol, [39]
1. $A \quad \longrightarrow_N B : c \parallel d = commit(A, INFO_A)$
$\qquad B$ computes $A \parallel INFO_A = open(c, d)$
2. $A \quad \longrightarrow_{WE} B : hash(c)$
$\qquad B$ verifies the hash.
Computational cost: $MW/2 + W^2/4 = 2.5M + 6.25$

Here the hash function is required to be second-preimage-resistant [30] (an intruder cannot find a second value v' such that $hash(v) = hash(v')$ for fixed v) as opposed to collision-resistance required for the protocol of Balfanz et al. where both v and v' are allowed to vary.

Pasini and Vaudenay [39] argue that this provides the same degree of authentication as the Balfanz et al. protocol, namely $2^{B/2} = 2^{80}$ computation steps, because the probabilistic commitment scheme avoids the possibility of a birthday attack. At the point where A is influenced to use the given $INFO_A$, the intruder cannot know what a nondeterministic component (a hidden random nonce of $B/280$ bits $= W/2$ words) that is injected by A into the commitment scheme will be, which is vital in obtaining a collision. Binding the information by a commitment scheme has the advantage of halving the number of empirical bits as well as halving the bit-length of the commitment scheme due to the nondeterminism introduced. These together reduce the cost to $MW/2 + W^2/4 = 2.5M + 6.25$.

More recently, Mashatan and Stinson [27] introduced another scheme which achieves the same level of security with the same number of empirical bits as the Pasini-Vaudenay protocol but does not require the use of a commitment scheme. Instead Mashatan and Stinson requires the use of a hybrid-collision-resistant (80-bit) hash function as defined in [27]. In the following protocol, k is a long random key of $B/2 = 80$ bits which is generated by A.

Mashatan-Stinson protocol [27]
1. $A \quad \longrightarrow_N B : INFO_A \parallel k$
2. $A \quad \longrightarrow_{WE} B : hash(INFO_A \parallel k)$
Computational cost: $(M + W/2)W/2 = 2.5M + 6.25$

Note that the random key k plays the same role as the non-deterministic component (also of $B/2$=80 bits) injected by A into the commitment scheme used in the Pasini-Vaudenay protocol. This therefore implies that both of these obtain the same level of security. The computational cost of the Mashatan-Stinson protocol is $(M + W/2)W/2 = 2.5M + 6.25$, which is the same as the Pasini-Vaudenay protocol. To the best of our knowledge, the above two protocols are currently the best non-interactive schemes in terms of the number of empirical bits relative to the level of security obtained.

The circumstances of non-interactive protocols bring the category of empirical channels $\rightarrow_{WE}$, as opposed to $\rightarrow_E$ or $\rightarrow_{SE}$, into question. For unless the recipient knows (s)he is in a protocol and confirms it by some explicit or implicit acknowledgement, how can we possibly state that an empirical message designed for one protocol run cannot be used for a second one? It seems to us that there are in fact three possibilities:

- There is, in fact, no bound on the life of a delayed empirical message. In this case an intruder can block a succession for messages from A to B, with the chances of success of each combinatorial search becoming greater as it has more and more empirical messages it can unblock – as in the birthday attack.

  It is clear that in any use of delayable empirical channels, one needs to be certain to ensure that this type of storage and re-use cannot occur. The feedback in interactive protocols is one, but in non-interactive channels it is a difficult question: we can avoid re-use through sequence numbers, but if all but one messages are blocked there is no need for re-use for the type of intruder strategy described above to work.

- There might be some mechanism which bounds the life of a delayed message. For example, given sufficiently synchronised clocks, a time-stamp would produce a real bound on the delayability of the message.

  Alternately there might actually be some feedback mechanism not explicitly mentioned in the protocol which tells A when her last empirical message has arrived. In the absence of signature mechanism for B that A can trust (unlikely in the circumstances we are considering) this feedback mechanism will probably have to be empirical.

- There may in fact be no significant delay possible: we actually have $\rightarrow_{SE}$. We discuss that case below.

It seems fair to remark that since even 80 bits will seem tedious for most humans to compare carefully, these one-way non-interactive protocols are not likely to find widespread use. Where it is humans who actually need to do this work: they would need to have a high level of commitment and possibly a well-designed user interface to ensure user compliance. For example, this is particularly a difficult task when 20 hexadecimal-digits (or 80 bits) numbers which need to be compared by humans attempting to set up a secure channels between some wearable sensors and laptops to upload medical data only differ in one or two positions or digits.

3.2 Short authentication strings over strong empirical channels

Gehrmann, Mitchell and Nyberg [13] took a different approach to preventing combinatorial search. They use empirical channels to transmit the b-bit output of a check function $\mathrm{MAC}_k()$ together with a b-bit key that has been instrumental in its computation. As suggested in [13, 14], a check function $\mathrm{MAC}_k()$ can be implemented by either CBC-MAC or universal hash functions based on error correcting code, which is potentially less efficient than digest functions as discussed in Section 2.2.2 and [33].

MANA I (Gehrmann, Mitchell and Nyberg), [13, 14, 15]		
1a.	A	$\longrightarrow_N B : A, INFO_A$
1b.	B	$\longrightarrow_E A$: 1-bit committed signal
		A picks a b-bit random number k
2.	A	$\longrightarrow_E B : k, \mathrm{MAC}_k(A \parallel INFO_A)$
Computational cost: $WM = 5M$		

The 1-bit committed signal, which can be implemented by a red line or a single button, is not a primitive property of empirical channels. It presence in the MANA I protocol aims to signal A that B has received Message 1 (which could be either original or fake). In the original description of the MANA I protocol, the pair of parties additionally need to agree on the success of the protocol with the help of some human interactions. Since this is not important with respect to the security analysis, we ignore the step in our description of the protocol.

To eliminate 1-bit empirical signals in the MANA I protocol, Vaudenay proposes to use a strong empirical channel (stall-free or instant delivery, and is denoted $\longrightarrow_{SE}$) to send the key and the check-value. Alternatively, we can replace the strong empirical channel with a bounded delay empirical one ($\longrightarrow^{t}_{BE}$), provided B checks that he has received Message 1 before Message 2 could have been sent. And thus $2b$ bits are transmitted in all. This idea turns the protocol into a non-interactive scheme. In the following description, we will modify the scheme slightly by using a digest function to compute the check-value. The rest of this analysis applies to both versions.

The V-MANA I protocol, [53, 39]*		
1.	A	$\longrightarrow_N B : A, INFO_A$
		A picks a b-bit random number k
2.	A	$\longrightarrow_{SE} B : k, digest(k, A \parallel INFO_A)$
		B verifies the digest.
Computational cost: M		

Binding $INFO_A$ (M words) directly to the SAS makes the protocol efficient because each node computes a single digest at a cost of M: much cheaper than a long output hash function in the protocol of Balfanz et al. ($5M$) and a commitment scheme in the Pasini-Vaudenay protocol ($2.5M + 6.25$). We note that this measurement only applies to the modified version of the V-MANA I protocol, where the digest function is used as opposed to CBC-MAC or longhash functions. The latter if used would rise the cost to $MW = 5M$ as in the MANA I protocol.

The protocol demonstrates that the use of the strong empirical channel, providing stall-free transmission, will lead to a significant fewer number of empirical bits in non-interactive schemes. Since the uniform distribution property of the digest function makes it impossible for the intruder to look for an $INFO_I$ digesting to the same value as $INFO_A$ in ignorance of k, this protocol comes close to preventing the intruder from performing any useful combinatorial search.

We note however that the protocol is suboptimal in human work relative to the level of security obtained. Any one can modify $INFO_A$ blindly in the first message and hope that the b-bit digests come out the same in the second one. This will occur with a probability of at least 2^{-b} irrespective of the value of the key, which means that $2b$ empirical bits only guarantee at best a 2^b security level.

Whilst the security proofs of this protocol given in [13, 15, 39] are largely correct, what these authors have not discovered is that the bit-length they choose for the key (which happens to equal b in this case) is too short compared to the digest output and the authenticated information $INFO_A$. As a consequence, it is impossible to construct a digest, MAC or check-value function such that the probability of any one-shot attack on the protocol is upper bounded by 2^{-b}. Since the weakness has a very profound impact on all other uses of the digest function, we are going to analyse the (off-line) computation complexity and its related probability of a successful one-shot attack on this protocol. We then deduce a longer key is required in order for the digest function to meet its specification.

We term b and r the bit-lengths of the digest output and the key k (in this protocol, $b = r = 16$ bits). The intruder first chooses some number c different keys $\{k_1, \cdots, k_c\}$. Using an off-line brute force search at the cost of $2^{bc/2}$ computation steps he can expect to find two different $INFO_A$ and $INFO'_A$,[5] such that:

$$\forall k \in \{k_1, \cdots, k_c\} : digest(k, A \parallel INFO_A) = digest(k, A \parallel INFO'_A)$$

[5]It might be clearer if we define $H_{\{k_1,\cdots,k_c\}}(X) = digest(k_1, X) \parallel \cdots \parallel digest(k_c, X)$, and if $digest$ is an ideal digest function, then so is the function H with respect to its $c \times b$ output-bits. As there is no limit on the bit-length of the input X, it normally takes $2^{cb/2}$ computation steps to search for a collision, due to the birthday paradox.

Assuming that the intruder can influence $INFO_A$ that A sends in the first message (i.e. chosen plain-text attacks), there is then an attack it can attempt.

$$\begin{array}{llll} 1. & A & \longrightarrow_N & I(B) & : A, INFO_A \\ & I(A) & \longrightarrow_N & B & : A, INFO'_A \\ 2. & A & \longrightarrow_{SE} & B & : k, digest(k, A \parallel INFO_A) \end{array}$$

Recall that in the above protocol, the key length r and digest length b are equal. The following calculations, where these numbers are kept separate, will allow us to draw more general conclusions.

After sending the first message, A picks a random key k: with probability $\frac{c}{2^r}$, $k \in \{k_1, \cdots, k_c\}$ and the attack is successful. On the other hand, with probability $(2^r - c)/2^r$, k is not in this set and the attack is only successful with probability (presumably) $2^{-b}(2^r - c)/2^r$.

Overall, at the cost of $\Theta(2^{cb/2})$ the chance of a successful one-shot attack is:

$$\mathbf{Pr}_r(c) = c \times 2^{-r} + \frac{2^r - c}{2^r} \times 2^{-b}$$

When $r = b$, this is significantly larger than the desired probability of 2^{-b}.

The above vulnerability indicates we need to increase the bit-length r of the key to avoid this type of attack. When r increases, 2^r will quickly become significantly bigger than 2^b and this will allow the likelihood of a successful one-shot attack – $\mathbf{Pr}_r(c)$ – to converge to 2^{-b}. This is however not feasible in this protocol, since the key must be sent with the digest value over the strong empirical channel that is severely limited in bandwidth.

An interesting question arises as we want to know how large the bit-length of the key should be in relation to a fixed amount of information we want to authenticate and the output bit-length of the digest function. There is a known theoretical bound of Stinson [49] on the bit-length of the key that can guarantee the digest function (or almost universal hash function) meets its specification: bit-length$(k) \geq$ bit-length$(INFO_A) - b$. We should remark that the bound can be met except for an infinitesimal tolerance in the digest collision probability ϵ for very much smaller lengths than this, see [34]. However, it always has to be significantly longer than b in practice.

This suggests that we should aim always to have key k noticeably longer than the digest in this style of protocol. Of course to do this without ruining efficiency in human effort, we need to find ways of communicating k over a high bandwidth (and insecure) communication link $\longrightarrow_N$ rather than empirically.

3.3 Improved version of the (V-)MANA I protocol

Given two weaknesses discussed in the previous section, we will present improved versions of the V-MANA I protocol that optimise the use of the expensive strong empirical channel. These improvements can also apply to the MANA I protocol. In other words, human comparison/handling of a b-bit short authentication string (SAS) always corresponds to probability 2^{-b} of a successful one-shot attack. Whilst this can only be done at the expense of introducing another (third) message sent over the Dolev-Yao channel we argue that this is not at all a bad trade-off since our highest priority is to minimise the empirical cost.

In contrast to the V-MANA I protocol, the key k generated by A in the following protocol can be as long as we want to ensure that the digest function meets the specification in Section 2.2.2. In addition, we can weaken the assumption that empirical messages' transmission is instantaneous to being of bounded delay as follows.

Improved version of the V-MANA I protocol (direct binding) [35]			
1.	$A \longrightarrow_N$	$B : M, longhash(k)$	
2.	$A \longrightarrow^t_{BE}$	$B : digest(k, M)$	
3.	$A \longrightarrow_N$	$B : k$	
Computational cost: $M + W^2 = M + 25$			

Note that the message order here, and in other improved schemes of the V-MANA I protocol, is more important than in all preceding protocols in this section. We specify that

- To ensure that B was committed without knowledge to key k when Message 2 was sent, B only accepts Message 2 after t time units or more of receiving Message 1.
- To ensure that B was committed to Message 2 when Message 3 was sent, A only sends Message 3 after t time units or more of sending Message 2.

Failure to follow these two principles in the implementation of the protocol, each of which uses the time bound on the empirical channel, can result in attacks that involve combinatorial searching.

Interestingly, we can replace the bounded delay empirical channel and the need to wait by a simple acknowledgement from B to A. The resulting protocol turns out to be the pairwise (one-way authentication) version of the HCBK protocol of Section 5 and [41].

Improved version of the MANA I protocol (direct binding) [41, 32, 33, 35]		
$1a.$	$A \longrightarrow_N$	$B : M, longhash(k)$
$1b.$	$B \longrightarrow_E$	A : 1-bit committed signal
2.	$A \longrightarrow_E$	$B : digest(k, M)$
3.	$A \longrightarrow_N$	$B : k$
Computational cost: $M + W^2 = M + 25$		

This scheme is flexible since the digest and key (messages 2 and 3) can be released in any order as long as A has received the commitment signal from B in the first message. It will often be the case that a bounded delay empirical channel and a one-bit acknowledgement signal are alternatives in this style of protocol design/structure.

Since the SAS in these schemes are functionally dependent on the authentic information M, we term these as the *direct binding* version of the Improved (V-)MANA I protocol, i.e. direct information binding strategy will be formally defined in Section 4. However, the computation cost is slightly increased to $W^2 + M = 25 + M$ due to the extra longhash required in the first message.

Readers who are interested in the formal security proof as well as variants using indirect binding and Diffie-Hellman style can find them in Appendix C or [34].

4 Interactive protocols

To authenticate a one-way message, it is obviously convenient to have a non-interactive protocol. We might observe that such a protocol in which the two human participants have to be active at the same time to implement a strong empirical channel (non-delayable) is less attractive: it must be seen as a long way along the road to being interactive.

Interactive protocols, where all parties contribute communications, have two clear advantages of their own. Firstly they can exchange messages without running the protocol multiple times. Secondly, as we shall see, the interaction makes it easier to reduce the number of bits that have to be passed empirically as well as the amount of computation power required at each node.

We show the significance of the idea of *joint commitment before knowledge*, introduced in Section 2.2.1, in providing the same level of security for all protocols presented in this section, i.e. the probability of a successful one-shot attack is upper bounded by 2^{-b}, where b is the bit-length of the SASs. For this reason, in all pairwise schemes (except the protocols of Hoepman and Wong-Stajano of Section 4.1), the value of the unique SAS is jointly committed to by both protocol participants. This therefore leads us to introduce the two following information binding strategies that help us achieve (joint) commitment before knowledge as well as classifying the many protocols considered in this survey.

- **Indirect information binding**: A protocol using a SAS is said to achieve *indirect information binding* if the SAS, jointly committed by every node, is *independent* of the information $INFOS$ parties want to authenticate.

  Typically, to construct indirect binding protocols considered in this paper, the SAS is the exclusive-or of random nonces to which every party has been individually committed at the beginning of each protocol run. In addition, these random nonces are also cryptographically bound to $INFOS$.

- **Direct information binding**: A protocol using a SAS is said to achieve *direct information binding* if the SAS, jointly committed by every node, is *dependent* on the information $INFOS$ parties want to authenticate.

  Typically, to construct direct binding protocols considered in this paper, the SAS is the output of some function applying to $INFOS$ in combination with secret keys individually committed to by every party at the beginning of a run. This binding strategy is also closely related to two protocol design principles **P1** and **P2** introduced later in this survey.

When we study the two strategies in sections 4.2 and 4.3, we find that a direct binding strategy has a clear advantage in efficiency over an indirect one. This arises from the potential to use a short output digest function to process the large $INFOS$ as opposed to a conventional long output cryptographic hash functions. The computational efficiency pay-off of direct binding strategy will be illustrated each time a protocol with direct binding is introduced, i.e. we will compare its efficiency explicitly with previous related protocols using indirect binding. The advantage will be demonstrated clearly when the cost of all schemes are gathered in three tables in Section 6.1.

4.1 Multiple empirical short authentication strings

We intend to describe two pairwise authentication protocols, the first by Hoepman [17, 18] and the second by Wong and Stajano [56, 57], in this subsection. In these schemes, parties manually compare or handle two different short authentication strings (SASs) each of $b = 16$ bits, so $2b = 32$ bits in all. We point out an important difference in how these two protocols process $INFOS$.

Hoepman [18] defines SASs as the outputs of a b-bit (short) hash function $shorthash()$, as mentioned in Section 2.2.2. In addition, the Diffie-Hellman tokens $g^{x_{A/B}}$ play the role of both $INFO_{A/B}$ and long fresh random nonces, and so must be unpredictable and fresh at each session.

Hoepman pairwise protocol, [18]		
1.	A	$\longrightarrow_N B : longhash(g^{x_A})$
1′.	B	$\longrightarrow_N A : longhash(g^{x_B})$
		Where x_Y is a long random nonce of Y
2.	A	$\longrightarrow_E B : shorthash(g^{x_A})$
2′.	B	$\longrightarrow_E A : shorthash(g^{x_B})$
3.	A	$\longrightarrow_N B : g^{x_A}$
3′.	B	$\longrightarrow_N A : g^{x_B}$
		A and B verify the long and short hashes.
		A and B then share the key $k = g^{x_A x_B}$
4.	A	$\longrightarrow_N B : longhash(g^{x_A x_B})$
4′.	B	$\longrightarrow_N A : longhash(g^{x_B x_A})$
Computational cost: $2(WM + M) = 12M$		

This protocol offers a good security, i.e. the probability of a successful one-shot attack is bounded by 2^{-b}, despite the use of b-bit shorthash functions can be explained through the idea of *joint commitment before knowledge*: both parties are jointly committed to $g^{x_A x_B}$ by publishing their shares of the commitment (i.e. $longhash(g^{x_{A/B}})$) in the first messages. It is therefore vital here that both parties must agree on when to finish inputting the first messages. Once the commitment phase is over, messages 2, 2' and 3, 3' can be sent out in any order without compromising the security. We will see an example of what goes wrong without (joint) *commitment before knowledge* at the start of Section 5.1, which discusses group protocols.[6]

Since the Diffie-Hellman tokens are the only information parties want to authenticate in this scheme, we can consider them as $INFO_{A/B}$ whose lengths are therefore M words. Since messages 4 provide shared secret validation (using $longhash()$ function in this case), they can be neglected in our cost analysis. As a result, each node has to compute 2 longhashes and 2 shorthashes of Diffie-Hellman tokens. Using our cost model of computing hash functions given in Section 2.4.2, the computation cost of the Hoepman protocol is of order $2(WM + M) = 12M$, where W and 1 are the word-lengths of long and respectively short hash functions.

Taking a different approach, Wong and Stajano [56, 57] propose another scheme which does not use a shorthash function, but does give the same security as in the Hoepman protocol with an equal number of empirical bits. The simplification comes with an extra cost of more than doubling the input size of the $longhash()$ function used in the commitment phase. This is the consequence of the inclusion of short and long nonces (R_Y and K_Y) of b and $B - b$ bits, respectively.

[6]Hoepman [17] introduced a modified (pairwise) version of the above scheme in which each party can receive multiple longhashes or commitments from unknown nodes at the very beginning of a run. But (s)he only pairs up with the one, who provides the matched single shorthash $shorthash(X)$ sent over the empirical channel in the second message. In this circumstance, A only sends out the shorthash iff he receives the 1-bit commitment empirical signal from B at the first place and vice versa. Furthermore, these acknowledgement signals must be transmitted over the empirical channel because they must not be blocked or delayed by the intruder. In this version, A does not need to know the identity of B during messages 1, so Hoepman refers to it as the *anonymous* case, whereas the protocol above applies to the *non-anonymous* case.

Wong-Stajano pairwise protocol, [56]		
1.	A	$\longrightarrow_N B : g^{x_A}$
1′.	B	$\longrightarrow_N A : g^{x_B}$
2.	A	$\longrightarrow_N B : longhash(A, g^{x_A}, g^{x_B}, R_A, K_A)$
2′.	B	$\longrightarrow_N A : longhash(B, g^{x_B}, g^{x_A}, R_B, K_B)$
		R_Y and K_Y are short and long random nonces of Y
3.	A	$\longrightarrow_E B : R_A$
3′.	B	$\longrightarrow_E A : R_B$
4.	A	$\longrightarrow_N B : K_A$
4′.	B	$\longrightarrow_N A : K_B$
		A and B verify the longhashes.
Computational cost: $2W(2M + W) = 20M + 50$		

The security of this protocol comes from the intruder's inability to invert the longhashes, or to predict the non-determinism introduced by the pair of nonces (R_X, K_X) at the point when these are committed to. As in the Hoepman protocol, both parties must receive each others' commitments (i.e. longhash) before they reveal their long and short nonces in the third and fourth messages. Once the commitment phase (sending out the longhashes) is over, messages 3, 3' and 4, 4' can also be transmitted in any order.

With respect to computation cost, while there is no shorthash function, the two longhashes (with long inputs) that need to be computed at each node result in a significantly larger cost of $2W(2M + W) = 20M + 50$ relative to $12M$ of the Hoepman protocol.

We now make two observations about the structure of this protocol. The high cost of computing *longhash* (due to a long input $\langle A, g^{x_A}, g^{x_B}, R_A, K_A\rangle : 2M + W$ words) can be improved slightly, as it is sufficient for A to bind g^{x_A} to the pair of random nonces (R_A, K_A). This leads to the elimination of messages 1 and 2, and indeed the same problem has been independently found and corrected by the inventors in their revised version of the paper, published in October 2007 [57]. However, they have not noticed that the Diffie-Hellman tokens (g^{x_A} and g^{x_B}) can play the dual role of the authentic information and fresh nonces if they are made unpredictable and fresh in each session. For this reason, we can further eliminate the need for long random nonces $K_{A/B}$ to simplify the protocol further. A detailed description of our modified version of the protocol will be given in Section 4.2.

Moreover both the protocols of Hoepman and Wong-Stajano are suboptimal in the amount of work required by the humans implementing the empirical channel, since they need to compare more than one string, whereas the same security level, i.e. the same probability of a successful attack, can be obtained in several ways by them comparing or sending a single SAS of the same length over the empirical channel. This weakness introduces another major disadvantage. If we want to generalise these protocols into multi-party versions then the number of different SASs (each party has to compare or handle manually) would always equal the total number of nodes: an unattractive prospect for the humans involved!

We end this section with a crucial observation: Hoepman chooses to bind Diffie-Hellman tokens *directly* to the SASs. This is not the case in the Wong-Stajano protocol, which is therefore more expensive in computational cost than the Hoepman protocol. By this we mean that the $INFO$s they are trying to authenticate are used directly in the evaluation of the empirically compared strings in the Hoepman protocol, while those compared in the Wong-Stajano protocol are not. These two different strategies are termed *direct* and *indirect* bindings, and we will explore and compare them in detail when we study protocols that can optimise human effort in the sections to come.

4.2 Indirect binding

In *indirect* binding protocols, the SASs, which are jointly committed to and manually communicated by parties, are functionally independent of the information they want to authenticate. This is the idea we have briefly seen in the Wong-Stajano protocol, and it appears in many other schemes in the literature [2, 6, 22, 23, 38, 53, 56, 57]. We will analyse these here. What distinguishes all of these from the Wong-Stajano protocol is a single SAS which is required to be compared over the empirical channel as opposed to multiple ones.

While there is no relation between the compared SAS and the authentic information $INFO$ (i.e. they are completely independent in the sense of probability), the security of the protocols comes from some mechanism binding some random nonces, which are instrumental in the computation of SASs, and $INFOS$ together in a secure way. Thus there is a tendency to use commitment schemes (described in Section 2.2.1) in these protocols to obtain that binding.

4.2.1 Indirect pairwise

We will discuss protocols covering two different circumstances in bootstrapping security. These were devised by Vaudenay [53] and Čagalj et al. [6] to establish one- and two-way authentication via one- and two-way empirical channels in a peer-to-peer network. We will extend their schemes into group versions in Section 5.2.

The following is the description of a pairwise scheme, invented by Vaudenay [53], that authenticates a single message $INFO_A$ from the party A to B using a one-way weak empirical channel.

Vaudenay pairwise one-way authentication protocol, [53]		
1.	A	$\longrightarrow_N B : INFO_A, c$
		Where $c \parallel d = commit(INFO_A, R_A)$,
		R_A is a short random nonce of A.
2.	B	$\longrightarrow_N A : R_B$
3.	A	$\longrightarrow_N B : d$
		B computes $R_A = open(INFO_A, c, d)$
4.	A	$\longrightarrow_{WE} B : R_A \oplus R_B$
		B verifies the correctness of $R_A \oplus R_B$
Computational cost: $MW = 5M$		

The protocol delivers the guarantee of authenticity of $INFO_A$, and even with a single b-bit SAS the probability of a successful one-shot attack is still bounded above by 2^{-b}. This is the consequence of:

- The exchange guarantees the value for R_A, that B has discovered by using the partial function $open()$, is the one that A intended.
- The commitment scheme ($commit()$) has strongly bound the message $INFO_A$ to R_A at a point where R_A is itself unknown to any attacker.

The above analysis applies to a one-shot attack. If we consider a q-shot attack then we need to take into account that the unique SAS of this protocol is transmitted over the weak empirical channel, and so can be stalled and delayed in other protocol runs. With q concurrent runs of A and B, the number of protocol sessions from the intruder's view will become q^2. Thus, the chance of a successful q-shot attack is upper bounded by $q^2/2^b$ as pointed out by Vaudenay [53]. We however argue that the origin of data transmitted over weak empirical channels cannot be forged, and to take advantage of the delayability of SASs the intruder will have to launch attacks on multiple instances of the *same* party A who is responsible for delivering the SASs. The above security analysis is therefore only valid with a small value of q because humans are highly sensitive to delays, i.e. they will quickly become aware that an attack is taking place, and so stop any attempt of running the protocol again as pointed out in Section 2.3. In contrast, if we replace $\longrightarrow_{WE}$ with $\longrightarrow_{E}$ then a SAS cannot be delayed from one to later runs, and so to have a fair chance of a successful attack, the intruder needs to run 2^b concurrent runs of (perhaps different) pair of parties: an 2^b-shot attack.

There is a single commitment used (committed by A, and decommitted or opened by B), hence the computing cost at each node is of order $MW = 5M$. Here both a commitment c and a decommitment d have the same length of W words. We note that the cost of XORing two short random nonces R_A and R_B is small compared to implementing the commitment scheme, and therefore is neglected.

Although Vaudenay's scheme halves the amount empirical communication relative to the protocols of Hoepman and Wong-Stajano, it only provides one-way authentication representing one role of pairwise schemes in this paper. In practice, we often want to achieve more than this, and that is why we now consider another protocol performing message authentication in both directions at the same time. Suppose B has some $INFO_B$ and wants to have it authenticated to A, then the natural way to tackle this problem is to make B commit to its information as done by party A. This idea, proposed by Vaudenay in Appendix A of [53], fortunately makes the protocol structure completely symmetrical. It is essentially the same as another protocol which is termed DH-SC and invented by Čagalj, Čapkun and Hubaux [6].

Čagalj-Čapkun-Hubaux two-way authentication protocol, [6]		
1.	A	$\longrightarrow_N B : INFO_A, c_A$
1′.	B	$\longrightarrow_N A : INFO_B, c_B$
		Where $c_Y \parallel d_Y = commit(Y, INFO_Y, R_Y)$,
		R_Y is a short random nonce of Y.
2.	A	$\longrightarrow_N B : d_A$
2′.	B	$\longrightarrow_N A : d_B$
		Y' computes $R_Y = open(Y, INFO_Y, c_Y, d_Y)$
3.	A	$\longleftrightarrow_E B : R_A \oplus R_B$
Computational cost: $2WM = 10M$		

Both of the above protocols use the *joint commitment before knowledge* principle to precommit two parties to the XOR of some random short secrets or nonces. This is achieved by parties outputting their shares of the commitment (of their own random nonces or keys) to each other in the first messages.

This scheme can be regarded as an upgraded version of the Wong-Stajano protocol of Section 4.1 in two ways. Firstly, the two initial messages in Wong-Stajano have been successfully eliminated. This is based on the ground that each node A only needs to commit to its $INFO_A$ at the beginning, so he has not to acquire $INFO_B$ at the time of computing the commitment. Secondly, the order of releasing the SAS and the decommitments has been reversed relative to the Wong-Stajano protocol, because the SAS and the decommitments here correspond to the two different short nonces and the long nonces in the Wong-Stajano protocol, respectively. As a consequence, parties only need to manually compare a single SAS, which is the XOR of short nonces R_A and R_B implicitly derived from the decommitments.

Regarding computation cost, the two commitments involved in this scheme double the cost of the protocol of Vaudenay, i.e. it is now $2WM = 10M$. On the other hand, if we quantify the cost relative to the amount of information authenticated then the protocols of Vaudenay and Čagalj-Čapkun-Hubaux will equal each other. The result illustrates the gain in efficiency of these in comparison with the protocols of Hoepman and Wong-Stajano of Section 4.1.

It is interesting to note that the same technique can be used to improve the human and processing cost of the Wong-Stajano protocol. The idea of eliminating the first two messages carrying $g^{x_{A/B}}$, removing the long random nonces as well as reducing the number of different SASs to a single one of b bits in the Wong-Stajano protocol can be demonstrated by our revised scheme. The scheme achieves the same level of security as the Wong-Stajano protocol of Section 4.1, i.e. the probability of a successful one-shot attack is bounded by 2^{-b}.

Improved version of the Wong-Stajano protocol *New*		
1.	A	$\longrightarrow_N B : longhash(A, g^{x_A}, R_A)$
1′.	B	$\longrightarrow_N A : longhash(B, g^{x_B}, R_B)$
2.	A	$\longrightarrow_N B : R_A \| g^{x_A}$
2′.	B	$\longrightarrow_N A : R_B \| g^{x_B}$
3.	A	$\longleftrightarrow_E B : R_A \oplus R_B$
Computational cost: $2W(1+M) = 10M + 10$		

Since there are two longhashes each node has to compute, the computation cost of this scheme is $2W(1+M) = 10M + 10$ which is less than a half of the original Wong-Stajano protocol $(20M + 50)$.

Another advantage of the protocols of Čagalj-Čapkun-Hubaux and Improved Wong-Stajano is that the symmetrical structure and a single SAS subsequently led us to realise the possibility of generalising it into a group version, as described in Section 5.2.

4.2.2 Hybrid protocol

Prior to discussing direct binding protocols, we describe an important scheme bridging the gap between the two strategies both in terms of protocol structure and computational cost. Pasini and Vaudenay [38] propose a two-way authentication protocol using the idea of Vaudenay's one-way scheme in Section 4.2.1. They make use of a truncated hash function that we have improved to a digest function and a symmetric empirical channel.

Pasini-Vaudenay two-way authentication protocol, [38]*		
1.	A	$\longrightarrow_N B : INFO_A, c$
		Where $c \parallel d = commit(INFO_A, k_A)$
		k_A is a long random nonce of A
2.	B	$\longrightarrow_N A : INFO_B, R_B$
		Where R_B is a b-bit random nonce of B.
3.	A	$\longrightarrow_N B : d$
		B computes $k_A = open(INFO_A, c, d)$
4.	A	$\longleftrightarrow_E B : R_B \oplus digest(k_A, INFO_B)$
Computational cost: $WM + M = 6M$		

Though the SAS $= R_B \oplus digest(k_A, INFO_B)$ depends functionally on $INFO_B$, it is probabilistically independent of $INFO_A$. This observation makes the scheme stand as a hybrid of direct- and indirect-binding protocols. Interestingly, the hybrid strategy is also reflected by the differences in the bit-lengths and the functionality of the two random nonces: R_B and k_A. R_B is protected by the structure of the protocol from guessing attacks and so can be short, whereas k_A is not and so has to be long; the two influence the final empirical string in different ways.

There is no need to use a commitment scheme to bind $INFO_B$ to R_B, so each node needs to compute a digest and either a commitment or a decommitment. The processing cost drops to $WM + M = 6M$, thanks to the efficiency of a digest function,[7] which is significantly cheaper than

[7]There should have been no improvement $(WM + WM = 10M)$, if we had not switched to the use of digest

the Čagalj-Čapkun-Hubaux protocol ($10M$) which is the fully indirect binding scheme.

4.3 Direct binding pairwise protocols

The direct binding approach requires the SAS, to which every party is jointly committed without knowledge, to be dependent on the information they want to authenticate. The Hoepman protocol that we have already studied falls into this category, but is not optimal in the human work. In this section and later ones we will discuss a number of other pairwise and respectively group protocols which are optimal in this respect. It should be noted, however, that any group protocol can be used to create a group of size 2, and, as we shall see, do so as efficiently as the ones in the present section.

Direct binding has been shown in two different situations to have an advantage in computation cost over indirect binding: Hoepman (direct) versus Wong-Stajano, and Pasini-Vaudenay (half direct or hybrid) versus Čagalj-Čapkun-Hubaux.

Our first task is to formalise the direct binding approach by using the following principle **P1** [32, 33].

P1 [32, 33] All the parties intended to be part of a protocol run should agree over an empirical channel on a short-output hash or digest of all the information that the parties want to authenticate. This method maximises the security of the authentication for a given amount of work on the empirical channel, and it leads to protocols that are as efficient computationally as any alternatives, and frequently considerably more efficient.

In all the protocols we consider, "all the information that the parties want to authenticate" is identified with $INFOS$ which is the concatenation of N pairs of the form $(A, INFO_A)$. Once the agreement required in **P1** has occurred, unless there is a hash or digest anomaly – different nodes in the group computing the same hash value or digest from different antecedents – clearly all the parties agree on all the data transmitted during the protocol.

In this section, a number of protocols providing mutual authentication are presented in an ascending order of computation efficiency and simplicity in protocol design. In addition to the common use of the direct binding strategy to obtain *joint commitment before knowledge*, they are all asymmetrical in structure, which is similar to the one-way authentication protocol of Vaudenay [53] in Section 4.2.1.

Unlike indirect binding schemes, parties need to generate fresh *long* random nonces or sub-keys, which have enough entropy to prevent them from being subject to a combinatorial search, i.e. it is possible to regard these long fresh sub-keys as the extended versions of short random nonces used in commitment schemes in indirect binding protocols. On the other hand, the security analysis of the direct binding schemes is similar to indirect binding ones provided the digest function is ideal, as specified in Section 2.2.2. In every case, the protocols have the same security (i.e. the probability of a successful one-shot attack is bounded by 2^{-b}) because nodes (and hence the intruder) do not know the final value of the digest key k until they are committed to the final value of the digest, truncated hash or universal hash output, thanks to the *joint commitment before knowledge* idea that provides a common theme to this family of protocols.

Since the roles of the sub-keys and $INFOS$ are different in digest computation, we will analyse how sub-keys are combined into a single digest key as we move along.

The following protocol is taken from the Bluetooth whitepaper [2], where k_A and k_B are long fresh sub-keys generated by A and B, and $trunc_b()$ takes the first b bits of its input.

function.

Bluetooth 2, [2]		
1.	A	$\longrightarrow_N B : INFO_A$
1′.	B	$\longrightarrow_N A : INFO_B$
2.	B	$\longrightarrow_N A : longhash(INFOS, k_B)$
3.	A	$\longrightarrow_N B : k_A$
3′.	B	$\longrightarrow_N A : k_B$
		k_Y is a long fresh key of Y
4.	A	$\longleftrightarrow_E B : trunc_b(longhash(f(k_A, k_B, INFOS)))$
Computational cost: $W(2M + W) + 2W(M + W) = 20M + 75$		

In this protocol, the sub-keys of A and B are concatenated with $INFOS$: $f(k_A, k_B, INFOS) = k_A \parallel k_B \parallel INFOS$. Since the operator is not commutative, parties have to arrange the sub-keys in the same order in which the distinct $(A, INFO_A)$s are concatenated into a single $INFOS$.

The inefficiency in using a truncated hash function will increase the computation cost of the above "Bluetooth 2" to $W(2M+W)+2W(M+W) = 20M+75$ as opposed to $W(2M+W)+2M = 12M + 25$ should we employ a digest function and XOR to combine sub-keys. Unfortunately, the latter will still be more expensive than the related protocols using indirect binding (the Čagalj-Čapkun-Hubaux protocol: $10M$, and Pasini-Vaudenay: $6M$) and the two following schemes, as it is redundant to bind $INFOS$ and sub-keys together by using both longhash function in Message 2 and in the SAS. Either of them would be sufficient for the obtained security.

Removing this unnecessary binding in Message 2 of Bluetooth 2 can increase its computational efficiency as well as simplicity (eliminating messages 1 and 1'), since B does not need to know $INFO_A$ (and $INFOS$) at the point when he is committed to k_B. This is what was proposed by Laur and Nyberg [22, 23]:

Laur-Nyberg protocol pairwise protocol, [22, 23]		
1.	A	$\longrightarrow_N B : INFO_A, c$
		Where $c \parallel d = commit(k_A)$
2.	B	$\longrightarrow_N A : INFO_B, k_B$
3.	A	$\longrightarrow_N B : d$
		B computes $k_A = open(c, d)$
4.	A	$\longleftrightarrow_E B : h_{k^*}(INFOS)$
		Here $k^* = g(k_A, k_B)$
Computational cost: $2WM = 10M$		

Here $h_{k^*}()$ is a universal hash function [49] of the appropriate length, i.e. b-bit in this case, whose specification is closely related to a digest function. The impact of removing redundancy can be seen in the decline of the computation cost of this protocol: $W^2 + 2WM = 25 + 10M$, which can be improved further because the Laur-Nyberg protocol has not exploited the short bit-length of SAS to increase efficiency. In this cost computation, we choose to ignore the cost of computing $g()$ to combine sub-keys in this calculation, since it is negligible relative to the computation of a universal hash function, which involves applying a cryptographic hash function to the $2M$-word input message $INFOS$ in the first place, as specified by Laur and Nyberg [22, 23], which results in a cost of order $2WM = 10M$.

Unlike Bluetooth 2, Laur and Nyberg use a different function $k^* = g(k_A, k_B) = (k_A^1 \cdot k_B) \oplus k_A^2$, using (polynomial) multiplication over a finite field GF($2^{r/2}$) to combine sub-keys. Here k_A^1 and k_A^2 are the first and second halves of k_A. Not only is this method expensive with long keys compared to concatenation and exclusive-or as we are going to propose, but also the parties need to agree on an irreducible polynomial of order $r/2$ prior to each session.

We observe that it would be equally satisfactory to use the combination of $k_A \oplus k_B$ and a digest function in place of $h_{k^*}(INFOS)$, resulting in an improvement of computational cost ($W^2 + 2M = 25 + 2M$) which is approximately 5 times cheaper than the Čagalj-Čapkun-Hubaux protocol ($10M$, the related protocol using indirect binding) should M get large. This clearly demonstrates the advantage in efficiency of direct binding protocols over indirect binding ones.

After this transformation and a replacement of a commit scheme with a longhash, the protocol becomes similar to the following, which also has the same cost of order $25+2M$. This was discovered independently by the author in the summer of 2006 when we combined ideas used in our SHCBK protocol (see Section 5.1) and Vaudenay's protocols (see Section 4.2.1).

Pairwise authentication scheme in Vaudenay's style *New*
$1.A \quad \longrightarrow_N B : INFO_A, longhash(k_A)$
$2.B \quad \longrightarrow_N A : INFO_B, k_B$
$3.A \quad \longrightarrow_N B : k_A$
$4.A \quad \longleftrightarrow_E B : digest(k_A \oplus k_B, INFOS)$
Computational cost: $W^2 + 2M = 2M + 25$

We subsequently discovered that the same ideas could be used to devise a more efficient version of the Hoepman protocol, which halves (and optimises) the amount of human work while achieving the same level of security, i.e. the probability of a successful one-shot attack is bounded by 2^{-b}:

Improved Hoepman protocol *New*
$1.A \quad \longrightarrow_N B : longhash(A, g^{x_A})$
$2.B \quad \longrightarrow_N A : g^{x_B}$
$3.A \quad \longrightarrow_N B : g^{x_A}$
$4.A \quad \longleftrightarrow_E B : shorthash(g^{x_A} \oplus g^{x_B})$
Computational cost: $WM + M = 6M$

The main difference between this and the previous schemes is that there is no $INFO_{A/B}$ because the Diffie-Hellman tokens play the dual-role of both $INFO_{A/B}$ and the long secret keys. In order for the protocol to be secure, the Diffie-Hellman tokens must be fresh at each session and unpredictable.[8] Also because of this, the digest function (an 2-input function) can be replaced by a single input shorthash function *shorthash*(); though the combination of this and the exponentiation of Diffie-Hellman needs to satisfy a specification similar to that of the digest function and the randomising effect of XOR in combining digest sub-keys.

In comparison with the Hoepman protocol, this requires a single SAS halving the human work. As in previous protocols, the computation of one longhash and one short hash of Diffie-Hellman tokens results in a cost of $WM + M = 6M$: exactly a half of the Hoepman protocol (direct binding, $12M$) and significantly lower than the improved version of Wong-Stajano protocol (the related protocol using indirect binding, $10M + 10$) of Section 4.2.1.

It is worth thinking for a moment about how a two-way agreement of a short string or similar occurs between a pair of people over an empirical channel. There are likely to be few situations where the string needs to be communicated both ways: all that is necessary is for one party to communicate it to the other, who checks equality with the data displayed on her machine and then tells the first of the agreement, i.e. sending a 1-bit committed signal over the empirical channel. We might therefore structure the above protocol as follows.

[8]It is possible but not necessary to replace $g^{x_A} \oplus g^{x_B}$ with $g^{x_A x_B}$ in this scheme.

Improved Hoepman′ protocol (One-way empirical channels)
1.$A \quad \longrightarrow_N B : longhash(A, g^{x_A})$
2.$B \quad \longrightarrow_N A : g^{x_B}$
3.$A \quad \longrightarrow_N B : g^{x_A}$
4.$A \quad \longrightarrow_E B : shorthash(g^{x_A} \oplus g^{x_B})$
5.$B \quad \longrightarrow_E A$: 1-bit committed signal
Computational cost: $WM + M = 6M$

And we could re-structure just about all the protocols in this paper similarly.

Once A has received Message 2 from B, he can send messages 3 and 4 in any order.

Wong and Stajano [57] give a protocol using this separated structure explicitly; it is however more expensive at $W(2M + 1) = 10M + 5$ as well as requiring another 1-bit empirical signal in Message 3:

Wong-Stajano protocol (One-way empirical channel), [57]
1.$A \quad \longrightarrow_N B : g^{x_A}$
2.$B \quad \longrightarrow_N A : B, g^{x_B}, MAC_{K_B}(B, g^{x_A}, g^{x_B}, R_B)$
R_B and K_B are short and long random nonces of B
3.$A \quad \longrightarrow_E B$: 1-bit committed signal
4.$B \quad \longrightarrow_E A : R_B$
5.$B \quad \longrightarrow_N A : K_B$
6.$A \quad \longrightarrow_E B$: 1-bit committed signal
Computational cost: $W(2M + 1) = 10M + 5$

The order of messages 4 and 5 can be interchanged in this protocol. The pair of messages 4 and 6 results in symmetric agreement on R_B: in fact they are just an implementation of "$A \longleftrightarrow_E B : R_B$".

It seems unlikely that the computation cost of the cheapest of these protocols can be reduced much further. It also seems clear that some sort of cryptographic binding of $INFOS$ to the empirical message is necessary, and our assumed model of the digest function appears to be a lower bound on that as we want to bind the *whole* of $INFOS$. Similarly, it is clear that for the *joint commitment before knowledge* approach to work, we need to have a token randomising the SAS value and being committed to before any node knows it. This has to be done with strong cryptography, and the hash function used in, for example, the Laur-Nyberg protocol appears to be as efficient as possible at doing this.

Another observation we want to make is that in the above protocols the use of a strong cryptographic primitive, such as a hash function or a commitment scheme, to protect the secrecy of long random nonces or keys, and in the meantime a much shorter (and therefore weaker) function to digest large $INFOS$ clearly aims to block combinatorial search and guess attacks separately. This idea is called *separation of security concerns* as discussed in [35].

5 Group protocols

The majority of work done in bootstrapping security in pervasive computing to date has focused on pairwise applications in a peer-to-peer network. However, we believe there is a similar potential for bootstrapping security in larger groups as can be shown by the following example. A group of people, who are present in the same location, might want to transfer data between them securely, meaning that they want it to be secret and of authenticated origin. They all have some pieces of computing hardware (e.g. a mobile phone or a PDA). However, none of them knows the unique name of any of the others' equipment, and in any case there is no PKI which encompasses them all.

Work in this area seems so far to have been restricted to the author's group (including Roscoe, Creese, Goldsmith and Zakiuddin), and more recently Valkonen et al. This has resulted in several group protocols presented in [9, 10, 11, 12, 32, 33, 41, 52]. In [32, 33], we identified the main challenges of bootstrapping group security in pervasive computing, and these can be explained as follows.

There is a slightly grey area for protocols building groups of more than 2. Should we or should we not be content if the presence of a corrupt party in a group means that communications between other trustworthy members of the group are themselves compromised? In some of the circumstances, where we may wish to use *ad-hoc* group formation protocols, it would be much better if the protocols were tolerant of corrupt members. We will, therefore, be careful about our assumptions on this front. It is obvious any key agreement protocol is at least partially compromised by the presence of a corrupt participant. However, protocols which merely authenticate public-key-like information are not automatically compromised: they could be said to be establishing a *local PKI*. As we will see, this will be successfully resolved by using the idea of (joint) *commitment before knowledge*.

The issue of scalability plays a crucial role in constructing group protocols because of the limited computation power of lightweight devices, and the fact that the amount of human work required will inevitably grow as the size of the group does. Our priority is still to optimise the human work relative to the security obtained. The best we can hope for is the same as in the binary case: it might be possible for a group to manually compare a *single* short authentication string (SAS) of b bits, and obtain the same 2^b level of security.

We have already seen that the direct information binding strategy is significantly more efficient than the indirect binding one for pairwise protocols (i.e. both of these information binding strategies are instrumental to achieving *commitment before knowledge* as pointed out in Section 4). In this section, we will see the same is true for group protocols, namely the HCBK and SHCBK protocols versus the group version of the indirect binding pairwise protocol of Čagalj-Čapkun-Hubaux. Interestingly, it is possible to further improve the efficiency in direct group protocols with a trade-off between human and computational costs, or by making use of protocol structure of the one-way scheme of Vaudenay [53].

5.1 Some existing direct binding group protocols

The following protocol introduced by Creese et al. [11] is probably the very first group authentication protocol in the area of pervasive computing. Here, $\forall A$ means that a message is sent to, or received by, all parties in the group $\mathbf{G}$ attempting to achieve a secure link between their laptops or PDAs. Pk_A stands for an uncertificated public key that A wants to authenticate to the group, whereas T_A and N_A are A's fresh nonces. The superscripted expression 'all Messages 2^d' represents the concatenation of all the decrypted content of messages 2 in alphabetical order. In addition, messages 4 do not add any extra security to the scheme: its presence aims to provide a confirmation of the shared secret information, i.e. this is similar in purpose to messages 4 in the Hoepman protocol of Section 4.1.

Group protocol of Creese et al. [11]			
1.	$\forall A \longrightarrow_N \forall A'$	:	A, Pk_A, T_A
2.	$\forall A \longrightarrow_N \forall A'$	:	$\{\text{all Messages } 1, N_A\}_{Pk_{A'}}$
$3a$.	A displays	:	$shorthash(\{\text{all Messages } 2^d\})$, number of processes
$3b$.	$\forall A \longrightarrow_E \forall A'$	:	users compare hashes and check numbers
4.	$\forall A \longrightarrow_N \forall A'$	:	$longhash(\{\text{all Messages } 2^d\})$

The protocol is shown by Roscoe [41, 32, 33] to be vulnerable to a man-in-the-middle and one-shot attack, related to the birthday paradox. The flaw arises from the short output of the shorthash function *shorthash*() used in messages 3a, and the intruder's ability to manipulate the content of messages 1 and 2. The details of the attack can be found in [32, 33, 41]. In spite of the attack, the protocol invented in 2003 introduced implicitly the principle **P1** in Section 4.3, which contributes to the optimisation of not only empirical work but also computational cost.

In summer 2005 [41], Roscoe corrected this flaw by introducing a trustworthy leader L who is responsible for generating a fresh key k_L of order $B = 160$ bits that is inputted into the digest function used in this scheme. The following scheme [32, 33] is a slightly simplified version of that protocol. Here, S represents a typical slave node, and A a typical node (either L or S). $init(L, A)$ is *true* if $L = A$ and *false* otherwise.

Hash Commitment Before Knowledge HCBK protocol, [32, 33, 41]				
0.	L	$\longrightarrow_N \forall S$	:	L
1.	$\forall A$	$\longrightarrow_N \forall A'$	:	$(A, INFO_A)$
$2a$.	L	$\longrightarrow_N \forall S$	:	$longhash(k_L)$
$2b$.	$\forall S$	$\longrightarrow_E L$	:	committed
3.	L	$\longrightarrow_N \forall S$	:	k_L
4.	$\forall A$	$\longrightarrow_E \forall A'$	:	$digest(k_L, INFOS), init(L, A)$
Computational cost: $W^2 + NM = 25 + NM$				

In this scheme, the parties have to agree on the b-bit digest of $INFOS$ and the leader's key k_L over the empirical channel. In addition, Message 2b has all the slaves communicate to L that they have received Message 2a, and are *committed* to their final digest value (though none of the slaves know it yet). Thus the 1-bit commitment signal must be sent over the unforgeable empirical channel that cannot be blocked. We will see shortly this represents one side of an interesting trade-off. As regards computation cost, each node has to compute a single cryptographic hash of key k_L and a b-bit digest value of $INFOS$, resulting in a cost of order $W^2 + NM = 25 + NM$.

The protocol is termed HCBK standing for *Hash Commitment Before Knowledge*, and its security relies on the trustworthiness of the leader L who generates the single digest key and consequently has control over the final digest value (the readers can find the security analysis of the HCBK protocol in [32, 33]). We have seen one previous protocol in which one party determines the final agreed value and there are two stages of commitment/agreement from the other parties (there the single other party). That is Wong and Stajano's "one-way empirical channel" protocol [57] from Section 4.3. In fact if messages 4 and 5 in that protocol are interchanged (a possibility we noted there), it is not hard to see that it becomes an indirect binding variant on the pairwise HCBK protocol.

In many circumstances, it is possible for such a leader to emerge (for example as the system whose owner initiates the protocol). However this is complicated if there may be an untrustworthy party present, since the leader *must* be trustworthy for the protocol to have any security.

In order to avoid this problem, the authors designed a protocol in which, provided the protocol has completed successfully, any pair or a sub-group of honest parties will have obtained the authentic information of each other irrespective of what other (dishonest) parties may have done. In [32, 33] we identified the following second principle, derived from the leader's role in the HCBK protocol and the *direct binding* strategy in obtaining commitment before knowledge, that essentially makes parties committed to the final digest value before any of them knows what the value actually is.

P2 A protocol offers the adversary no strategy to force digest agreement to be more likely than chance if, at some point in every partial run, for some node A,

1. A is committed to a value d such as $d = digest(k^*, INFOS)$; and
2. A has randomly selected a value k_A such that:
   (a) k_A randomises the value of k^*;
   (b) no other participant knows the value of k_A at this point in the run; and
   (c) no input received by A can eliminate A's randomising effect of k_A on k^*.

This is clearly a formalisation and refinement of the (joint) *commitment before knowledge* concept that we have used throughout this paper.

In the resulting protocol, every node will plays a role similar to the leader in the HCBK protocol and thus follow **P2**: each node A now needs some fresh and unpredictable sub-key k_A of B bits to contribute to the final digest value.

Symmetrised HCBK protocol (SHCBK), [32, 33]		
1. $\forall A \longrightarrow_N \forall A'$	:	$A, INFO_A, longhash(A, k_A)$
2. $\forall A \longrightarrow_N \forall A'$	:	k_A
3. $\forall A \longrightarrow_E \forall A'$	:	$digest(k^*, INFOS)$
		where k^* is the XOR of all the k_A's for $A \in \mathbf{G}$
Computational cost: $NW^2 + NM = 25N + NM$		

This protocol is termed Symmetrised HCBK (or SHCBK) due to the similarity with the HCBK protocol and its symmetrical structure. In the first messages, the purpose of the inclusion of the identity A inside the longhash is to prevent an intruder from eliminating A's randomising effect on k^* by simply copying its longhash value, i.e. this follows part 2(c) of principle **P2**. We note the same protection is required in the Čagalj-Čapkun-Hubaux protocol. This reflexive attack does not however work against the HCBK protocol as there is only one cryptographic hash $longhash(k_L)$ generated by the leader. A clear advantage of the SHCBK protocol over the HCBK protocol is the elimination of one-bit and empirical commitment signals (from all slaves to the leader in the HCBK protocol).

Since everyone takes responsibility separately for influencing the final digest key k^* and the final digest value, neither any one nor any proper subset of $\mathbf{G}$ can determine the digest value until all the sub-keys are revealed in messages 2. Indeed, whatever other parties do, the influence of a particular node A completely randomises the final digest value. As a result, this authenticates trustworthy parties to each other irrespective of what others (dishonest nodes) may have done as long as they agree on the same SAS. In other words, this protocol is tolerant of corrupt parties: one of the main challenges in designing group protocols as mentioned at the beginning of Section 5.

The use of XOR to combine different sub-keys in the SHCBK protocol was shown to be secure [33]. The intuitive reason behind this choice of the operator is that, thanks to the identities included in longhashes of messages 1 to avoid a reflexive attack, both final digest keys (denoted k^*_A and k^*_B) computed at trustworthy parties A and B are uniform random variables that can be considered independent of all k^*_C introduced by other parties (corrupt or otherwise). k^*_A and k^*_B

can either be independent or dependent. When they are independent of each other, the probability of a digest anomaly is 2^{-b} as defined in the first part of the digest function specification given in Section 2.2.2. When they are dependent (which they will be – indeed equal – if all nodes are trustworthy and there is no intruder), the only relation that can occur between them is linear of the form $k_B^* = \theta \oplus k_A^*$ where the intruder can choose θ, as discussed in [33]. But this again does not give him any advantage thanks to the second part (in particular "$\oplus\ \theta$") of the digest function specification.

The robust security achieved here comes at the expense of increased computation cost relative to the HCBK protocol: each node now has to compute N longhashes (one for generating its own Message 1 and $N-1$ for checking the coherence of what other nodes send) as opposed to the single longhash value of the HCBK protocol. The computation cost of the SHCBK protocol is thus $NW^2 + NM = 25N + NM$. This is the other side of the trade-off mentioned above: we have *gained* in increased corruption tolerance and the loss of the empirical commit signal, but *lost* computational efficiency.

Though designed as group protocols, both the HCBK and SHCBK protocols can be easily turned into pairwise ones. If we replace the two-way empirical channels used in the HCBK protocol with one-way channels from the slaves to the leader then we will have a one-way authentication group protocol: all the slaves are authenticated to the leader.

5.2 Indirect binding group protocol

We have claimed that the direct binding approach (the HCBK and SHCBK protocols) remains more efficient in group protocols than indirect binding as it was in pairwise ones. The argument is true because it is more efficient to have the SAS created from the presumed large $INFOS$ rather than it is to have each $INFO_A$ bound to random nonces by full-power cryptography to resist combinatorial search. In order to illustrate this advantage, we will generalise the (symmetrical) indirect binding pairwise scheme of Čagalj, Čapkun and Hubaux [6] in Section 4.2.1 into a group protocol. The level of security achieved by this scheme is the same as the SHCBK protocol: (1) tolerant of corrupt parties; and (2) the probability of successful one-shot attack is upper bounded by 2^{-b}, here b is still the bit-length of the single SAS transmitted over the empirical channel.

Indirect-binding group protocol *New*		
1.	$\forall A$	$\longrightarrow_N \forall A' : INFO_A, c_A$
		Where $c_A \parallel d_A = commit(A, INFO_A, R_A)$,
		R_A is randomly picked by A.
2.	$\forall A$	$\longrightarrow_N \forall A' : d_A$
		A' computes $R_A = open(A, INFO_A, c_A, d_A)$
3.	$\forall A$	$\longrightarrow_E \forall A' : \bigoplus_{A \in \mathbf{G}} R_A$
Computational cost: $NWM = 5NM$		

From the protocol, we can see that all of the $INFO_A$s must be committed separately: each node always has to commit once (for its own $INFO$), and de-commit or open $N-1$ times to verify the commitments of all other parties. This results in a computation cost of order $NWM = 5NM$, which is approximately $W = 5$ times as expensive as either HCBK or SHCBK.

An important observation we want to make is that in this scheme any untrustworthy party I can fool other participants of group $\mathbf{G}$ into accepting different versions of its own $INFO$, i.e. $INFO_I$ and $INFO'_I$. This can be easily done if I sends the commitments of different versions of its $INFO$ relative to the *same* short random nonce R_I to others in the first messages.

1. $I \longrightarrow_N A : INFO_I, c_I$
   $I \longrightarrow_N B : INFO'_I, c'_I$
   Here :
   $c_I \parallel d_I = commit(I, INFO_I, R_I)$,
   $c'_I \parallel d'_I = commit(I, INFO'_I, R_I)$
   $A \longleftrightarrow_N B : INFO_{A/B}, c_{A/B}$

2. $I \longrightarrow_N A : d_I$
   $I \longrightarrow_N B : d'_I$
   $A \longleftrightarrow_N B : d_{A/B}$

3. $\forall A \longrightarrow_E \forall A' : R_I \oplus R_A \oplus R_B$

Thus parties still agree the XOR of all short nonces manually in the third messages. However, we do not consider this as a valid attack because we do not care whether we get the right or wrong information about an untrustworthy node in an authentication protocol. Conversely, if we want to turn this into a key agreement protocol then the first assumption we have to make is that all participants are honest, as discussed at the start of this section.

5.3 Modified versions of the HCBK and SHCBK protocols

The difference in computational efficiency between the SHCBK and HCBK protocols raises the question of whether it is possible to reduce the amount of computation processing in the SHCBK protocol without compromising its security, i.e. being tolerant of corrupt parties and the probability of a successful one-shot attack is bounded by 2^{-b}. A small improvement turns out to be possible if we make use of a technique used in Vaudenay's one-way scheme and direct binding pairwise protocols in sections 4.2.1 and 4.3. On the one hand, this can slightly reduce the number of commitments or longhashes at each node. On the other hand, it makes the schemes asymmetrical in structure. This will be explained as follows.

Suppose there are $N-1$ leaders Ls out of a total of N parties, where each leader has to generate a fresh sub-key, and compute and send its longhash over the normal network. The single node left is the unique slave S, who transmits its fresh sub-key k_S to other nodes over the clear after receiving longhashes from every leader. Below A is a typical node which is either S or L.

De-symmetrised SHCBK protocol, [52]*			
0. S	$\longrightarrow_N$	$\forall L$	$: S$
1. $\forall L$	$\longrightarrow_N$	$\forall A$	$: INFO_L, longhash(L, k_L)$
2. S	$\longrightarrow_N$	$\forall L$	$: INFO_S, k_S$
3. $\forall L$	$\longrightarrow_N$	$\forall A$	$: k_L$
4. $\forall A$	$\longrightarrow_E$	$\forall A'$	$: digest(k^*, INFOS)$
			where k^* is the XOR of all the k_A's for $A \in \mathbf{G}$
Computational cost: $(N-1)W^2 + NM = 25N + NM - 25$			

We discovered this shortly after the SHCBK protocol. It was also independently invented by Valkonen, Asokan and Nyberg [52], who were not aware of the SHCBK protocol and neither addressed the issue of tolerance of untrustworthy parties nor the use of an efficient (short-output) digest function.

As can be seen from the protocol, while there is no commitment attached to the sub-key k_S of the slave, the fact that it is the only one treated in this way guarantees that it will not be manipulated by the intruder. This is as resistant to corrupt participants as the SHCBK protocol, but of course separate arguments are required in considering a pair of trustworthy ones, depending on whether one of them is the single slave or not.

At the expense of introducing the role of the slave and making the protocol asymmetrical, the total number of longhashes per node declines to $N-1$ which corresponds to a processing cost of $(N-1)W^2 + NM = 25N + NM - 25$. This is cheaper than the SHCBK protocol by $W^2 = 25$ units per node, though we suspect that the asymmetry introduced into the communication regime will in practice mean that it is no better: nodes will spend more time waiting. Nevertheless, it illustrates the possibility of further improving the computation efficiency by careful analysis.

Unfortunately, it appears impossible to employ the same technique to decrease the number of longhashes any further. Once there are two or more slaves in a single run, the scheme will be vulnerable to a *man-in-the-middle* attack in which the intruder impersonates all the slaves to talk to all the leaders and vice versa. Intuitively, this is because principle **P2** has been violated: any slave, who sends its k_A before having k_B (or a commitment like $longhash(k_B)$) for each other B, is revealing its last piece of information too soon before it is committed to the final digest value. An example of this attack, applied to the case of one leader and two slaves, can be demonstrated in Appendix B.

We can however reduce computational cost if we are prepared to weaken our corruption tolerance requirement towards that of the HCBK protocol. With the addition of 1-bit empirical commitment signals like those in the HCBK protocol and allowing the number of leaders l to vary between 2 and N, we propose a *hybrid* protocol. In other words, rather than having a single leader generating the digest key by itself as in the HCBK protocol, we will now have l leaders generating l sub-keys, here $l \in [2, N]$. The effect is that all of the leaders would have to be corrupt for the protocol to fail, otherwise the probability of a successful one-shot attack is upper bounded by 2^{-b}. For example, if everyone trusts A or B, then it may be appropriate to choose both as leaders, meaning that all nodes have to compute $l = 2$ longhashes.

Below, S represents a slave, L is a leader, and A is either a slave or a leader. SL is the set of l leaders' identities broadcast to every one in Message 0 by a single node T, who knows this information.

Hybrid HCBK protocol *New*		
0.	T	$\longrightarrow_N \forall A : SL$
1.	$\forall A$	$\longrightarrow_N \forall A' : A, INFO_A$
2a.	$\forall L$	$\longrightarrow_N \forall A : longhash(L, k_L)$
2b.	$\forall S$	$\longrightarrow_E \forall L$:1-bit committed signal
3.	$\forall L$	$\longrightarrow_N \forall A : k_L$
4.	$\forall A$	$\longrightarrow_E \forall A' : digest(k^*, INFOS), leader(SL, A)$
		Where k^* is the XOR of all the k_L's for $L \in SL$
Computational cost: $lW^2 + NM = 25l + NM$		

The protocol is termed the *Hybrid Hash Commitment Before Knowledge* (HHCBK) protocol because it applies to the hybrid case and is in effect a hybrid of the HCBK and SHCBK protocols.

One problem here is establishing which of the nodes are to be leaders in such a way that this does not add greatly to the empirical communication burden of the protocol.

Suppose that the set SL of leaders is actually established by insecure communications between the nodes. One way to make the protocol secure would be to have all nodes agree not only the digest but also the set of leaders with each other and with their systems' views on this subject: we assume that $leader(SL, A)$ indicates whether A is a leader or not. Post hoc this establishes agreement on who the leaders are a very strong way, but with a lot of leaders it could be expensive.

Imagine a weaker rule: a leader has no duty to check on the *leader* information from others, and a slave only has to convince himself that there is, amongst leaders who announce themselves, at least one leader who is trustworthy. This is perhaps surprisingly sufficient.

To see this note that slaves A and B, and a trustworthy leader L (amongst those identified by A) have all agreed the digest. We should consider a number of possibilities, all of which could be brought about by the intruder and the weaker use of the *leader* information.

- A's final digest was not influenced by sub-key k_L. In this case, the probability of A's and L's digests agreeing is no more than 2^{-b}, by **P2** applied to L.
- A's final digest was influenced by sub-key k_L, as was B's. In this case, we can use the same argument that applies to the HCBK protocol.
- A's final digest was influenced by k_L, but B's was not. In this case, final digest keys k^*_A and k^*_B are independent.

Of course, in order for the digests to agree, it is necessary that the *nodes* as opposed to their human users know who all the leaders are. What the argument above shows is that it is not always necessary for the humans to check every detail of this.

It is interesting to see the trade-off between the preliminary security assumption and the computation cost of $lW^2 + NM = 25l + NM$ in this protocol. What this formula tells us is if we want to improve the computation cost of the protocol, we need to decrease the number of leaders l in the group $\mathbf{G}$, and in effect increase the trustworthiness requirement from each leader.

6 Conclusions and further work

In this section, we tabulate efficiency analysis of the various protocols discussed in this paper, discuss the results and other topics relevant to these classes of protocol as well as looking ahead to work that still needs to be done.

6.1 Efficiency

In this section, we tabulate the efficiency of all the protocols we have described according to the two measures we have used throughout: the amount of empirical communication and the computation effort required for the cryptographic primitives. More complex models might take into account the amount of high bandwidth required and a measure of the concurrency that is possible between nodes, but we do not go into that level of detail.

We group them into three tables: non-interactive (one-way) authentication, interactive mutual authentication and group protocols.

Our main measure of empirical work is the number of bits that each user has to compare. Of course we do not imagine that they will compare actual *bits*, but some more friendly representation of the data! There is also a trade-off between how much work it is to compare information and the degree of certainty we have that human users will actually do the work required of them. At one extreme we can imagine the leader in an HCBK network announcing the final digest and asking the rest of the humans present to put up their hand if the value displayed on their PDA's does not agree; at the other we can imagine that an implementation allowing the connection of a credit card to a merchant might require the customer to type the merchant's digest into his card (or a device holding it) so the card can do the comparison itself. But both these last issues are implementation dependent and orthogonal to the logical structure of the underlying protocol, so we will stick to our simple measure. In those protocols that require the extra confirmation message over the empirical channel (MANA I, Wong-Stajano, HCBK, HHCBK etc) we write "$b+1$" as the amount of empirical effort.

In these tables we have used the simple cost model of hash and digest functions described in Section 2.4.2: the cost is proportional to the product of the length of the information being digested and the width of the output.

The non-interactive protocols are shown in Table 2. There is a relationship between how much we assume of the empirical channel and how much work is required over it. Unlike later tables, we might note that the levels of security are not the identical in the protocols listed: here $B = 160$ and $B/2 = 80$ are examples of the numbers of bits required to make a hash function strongly and weakly collision resistant, it is assumed that $2^{-b} = 2^{-16}$ likelihood of one-shot attacker success is sufficiently small in all cases (except the first two protocols), but for reasons discussed earlier the (V-)MANA I protocols do not attain this. Of course, one might want to change any of these numbers for good reason, but we believe that the relative differences of them will not be greatly different if this is done. Therefore, the lessons about relative cost that this table teaches us will remain true.

The same will, naturally, be true of the other tables. The reader is advised to regard constants like $160 = B$, $32 = w$, $16 = b$ and $25 = (160/32)^2 = (B/w)^2 = W^2$ as "variable constants", where exact numbers are given for illustrative purposes.

The other tables cover pairwise protocols and groups. For the latter, in each case we get another pairwise protocol by setting $N = 2$: these are all competitive in the pairwise table.

It is also necessary to point that since most direct binding protocols invented by other authors to date do not use a digest function to produce SASs, we will therefore illustrate the difference by giving the computation cost of both cases in 3 tables. The truncated longhash or universal hash functions ([49] that require a longhash function to compress large messages into a fixed number of bits initially) will be denoted (*longhash*).

Following the principle **P1** and the use of the digest function, direct binding protocols are much more efficient than indirect binding ones as can be seen from all three tables: about up to $B/w = W = 5$ times more efficient should M get large. The larger M (the length of $INFOS$) is, the more accurate this effect (this is not necessarily a clear advantage for direct binding in the case of the Hoepman and Wong-Stajano protocols because the information parties want to have authenticated only includes one or two Diffie-Hellman tokens that are quite small). This is likely to be the case whenever

(i) A large amount of authenticated information is being passed from one participant to another. We might note in this connection that direct binding protocols are a more efficient way of doing this than any method that the nodes are likely to use once a secure connection is up and running, since the latter is likely to use either conventional symmetric cryptography or standard length hash functions.

(ii) Large amounts of information needs to be passed to enable the users of the network to be able to associate the logical members of the network either to other human users (e.g. photographs) or function (e.g. manufacturer's certificate).

(iii) There are many nodes present in a group: $INFOS$ can be expected to expand proportionately to this.

With respect to group protocols we recall that there is a trade-off between processing cost and the amount of corruption resistance required as well as with eliminating the 1-bit confirmation message.

Protocol	Binding	Human work(bit)	Computation cost
Balfanz et al.	Direct	B=160 ($\rightarrow_{WE}$)	$WM = 5M$
Pasini-Vaudenay	Indirect	$B/2$=80($\rightarrow_{WE}$)	$\frac{MW}{2}+\frac{W^2}{4}$=2.5$M$+6.25
Mashatan-Stinson	Direct	$B/2$=80 ($\rightarrow_{WE}$)	$(M+W/2)W/2$ = 2.5M+6.25
MANA I (CBC-MAC)	Direct	2b+1=33 ($\rightarrow_E$)	$WM = 5M$
V-MANA I (*digest*)	Direct	$2b = 32$ ($\rightarrow_{SE}$)	M
Improved MANA I	Direct	$b + 1 = 17$ ($\rightarrow_E^t$)	$M + W^2 = M + 25$
Improved MANA I	Indirect	$b + 1 = 17$ ($\rightarrow_E^t$)	$W(M + W) = 5M + 25$
Improved MANA I	Direct(D-H)	$b + 1 = 17$ ($\rightarrow_E^t$)	$WM + M = 6M$
Improved V-MANA I	Direct	$b = 16$ ($\rightarrow_{BE}^t$)	$M + W^2 = M + 25$
Improved V-MANA I	Indirect	$b = 16$ ($\rightarrow_{BE}^t$)	$W(M + W) = 5M + 25$
Improved V-MANA I	Direct(D-H)	$b = 16$ ($\rightarrow_{BE}^t$)	$WM + M = 6M$

Table 2: One-way authentication protocols

Protocol	Binding	Human work(bit)	Computation cost
Hoepman	Direct	$2b = 32$	$2(WM + M) = 12M$
Improved Hoepman	Direct	b =16	$WM + M = 6M$
Improved Hoepman$'$ (one-way empirical)	Direct	$b+1$=17	$WM + M = 6M$
Wong-Stajano (one-way empirical)	Direct	$b+1$=17	$W(2M + 1) = 10M + 5$
Wong-Stajano	Indirect	2b=32	$2W(2M + W) = 20M + 50$
Improved Wong-Stajano	Indirect	b =16	$2W(M + 1) = 10M + 10$
Vaudenay ($\rightarrow_{WE}$)	Indirect	b =16	$WM = 5M$
Čagalj-Čapkun-Hubaux	Indirect	b =16	$2WM = 10M$
Pasini-Vaudenay (*longhash*)	Hybrid	b =16	$WM + WM = 10M$
Pasini-Vaudenay (*digest*)	Hybrid	b =16	$WM + M = 6M$
Bluetooth 2 (*longhash*)	Direct	b =16	$W(2M$+$W)$+$2W(M + W)$=20M+75
Bluetooth 2 (*digest*)	Direct	b =16	$W(2M$+$W)$+2M=12M+25
Laur-Nyberg (*longhash*)	Direct	b =16	$W^2 + 2WM = 10M + 25$
Laur-Nyberg (*digest*)	Direct	b =16	$W^2 + 2M = 2M + 25$
Vaudenay-style (*digest*)	Direct	b =16	$W^2 + 2M = 2M + 25$

Table 3: Interactive pairwise two-way authentication protocols (unless indicated they all use two-way empirical channels: $\longleftrightarrow_E$)

Protocol	Binding	Human work(bit)	Computation cost
Indirect binding	Indirect	16	$WNM = 5NM$
HCBK	Direct	$b+1$=17	$NM + W^2 = NM + 25$
SHCBK	Direct	b =16	$NM + NW^2 = NM + 25N$
De-symmetrised SHCBK(*longhash*)	Direct	b =16	WNM+$W^2(N$-1)=5NM+25(N-1)
De-symmetrised SHCBK (*digest*)	Direct	b =16	NM+$W^2(N$-1)=NM+25(N-1)
Hybrid HCBK	Direct	$b+1$ =17	$NM + W^2l = NM + 25l$

Table 4: Group authentication protocols (they all use empirical channels: $\longrightarrow_E$)

6.2 Short-term public key cryptography

We anticipate that in many, probably a majority, of the practical uses of the classes of protocol described in paper, one of the main objectives is the bootstrapping of a means of secret and authenticated communication between the parties. In almost all such cases, we expect that this will be done by establishing a symmetric session key to be used in conjunction with some encryption algorithm in a way that gives both secrecy and authentication.

One cannot establish such a session key directly in the $INFO_A$s, since all such information is public following the protocol run. Rather, as anticipated in many of the protocols and discussion earlier in this paper, we can expect that this is done either by including public keys in the $INFO_A$s, or alternatively Diffie-Hellman tokens. Of course Diffie-Hellman tokens can then be combined directly into session keys, whereas public keys have to be used properly to establish authenticated session keys.

In our environment where we desire low power consumption and perhaps simple processors, the large modulus calculations needed to perform either Diffie-Hellman or public-key cryptography are unattractive. It is worth noting however that there are opportunities for efficiencies in the use of public keys arising from the style in which we use them.

In a PKI, it is public keys themselves that are used for long-term authentication. Any breach of such a key will have disastrous long-term consequences. However, in our usage, public keys can be fresh for every run of a protocol and are only used once or twice in the initial set-up phase. So provided we can be confident that a public key cannot be broken during the length of a session, we can be sure that the communication in that session are properly authenticated, and that any computing power directed at cryptanalysing it subsequently can only reveal the secrets of a single session.

The generation of fresh public/private key pairs can, or course, be done in advance of a session or a collection of them might be "loaded" periodically onto a device that does not have the computational power to generate them. (This would, naturally, have to be from a trusted source – perhaps it is even an extra function built into the device's power supply!)

In any event, a security assessment of a particular application may well, because of the short-term nature of public keys, require shorter (and therefore easier to use) public keys than in a PKI.

6.3 Conclusions

It this paper we have surveyed the literature on a new and – we believe – important style of protocol, examining non-interactive, interactive and group protocols. We have also discovered that, even though groups of these protocols have been invented independently and presented in different notation, the basic principle of *commitment before knowledge* underlies all of those that either attain or nearly attain the optimal empirical performance.

Very different from any other families of security or cryptography protocols, human interactions play a central and very important role in the security of the authentication schemes presented in this survey. For this reason, we have tried to rigorously analyse how much human effort (measured in the number of bits the humans have to keep in their minds) is required, and more importantly whether it is optimal with respect to the obtained level of security. And we are glad to claim that the result has been very positive in all three types of authentication protocols.

On the one hand, our aim of this survey is to summarise and categorise all existing protocols invented so far into comprehensive groups. On the other hand, we also try to give the readers a better of view of where this research area is heading to, and what can be done to make these protocols usable in practice.

6.4 Future research

After running a successful session of one of the group protocols, the group has essentially bootstrapped a *local PKI.* If such a local PKI is going to be more than short term, we are going to have to address issues such as how to add extra nodes, form the union of two groups, and excluding nodes. In other words how does one maintain a local PKI? An initial, but somewhat inefficient, approach to this is described by Valkonen et al. [52].

The nature of the protocols we have described, and especially the need to take the combinatorial search power of attackers into account when quantifying security, apparently fall outside the range of the successful tools for protocol analysis produced in the last decade or so. If this new class of protocols is to be as important as we believe it is important either that these tools or their methodologies are developed or new tools created to handle them. It may well be appropriate to use or adapt probabilistic model checkers such as PRISM [1] for this purpose.

Another interesting possibility is to apply and extend our existing work in authentication protocols in pervasive computing into other security applications such as electronic polling/voting (physical envelopes) [31], auction protocols (anonymous physical broadcast channel) [48] and e-cash [8] where human interaction is also employed but little if any investigation has been undertaken to analytically quantify and optimise them.

And finally, designing efficient ways of comparing the SAS manually in different circumstances (and applications) is also very important for the future of these protocols. As a result, this area has received much attention from many different research groups [16, 25, 28, 29, 45, 51] recently.

Acknowledgements

Long Nguyen's work on this paper was supported by studentships from QinetiQ Trusted Information Management and the Ministry of Education and Training of Vietnam.

Roscoe's work on this paper was partially supported by funding from the US Office of Naval Research.

We are grateful to anonymous referees whose detailed comments allowed us to greatly improve the paper.

A The importance of empirical display of $leader(L, A)$ in the Hybrid HCBK protocol

In order to demonstrate if the information $leader(L, A)$ were not to be communicated over the empirical channel the HHCBK protocol would suffer from an attack, we shall look at the situation where there are two users A and B. The intruder invents $INFO'_X$ for each of them in which it says X is a leader, i.e. A will receives $INFO'_B$ saying that B is the leader and vice versa. In fact neither A nor B act as a leader, and the intruder is able to send hash keys to A and B such that the final digests computed at A and B agree, i.e. each believes the other to be the leader. Of course this works equally well with any two disjoint sets of "leaders". This attack works when the attacker can block the 1-bit commitment signal sent via the empirical channel. Otherwise both A and B will realise something wrong going on as both of them are not supposed to receive any commitment as neither of them created any longhash. This will depend on whether commitment signals are directed at only specific leaders in Message $2b$, which is specified in this protocol to save the amount of human work, however real life implementations might vary significantly from our specification.

B Attack on group protocol with two slaves

In this appendix, we demonstrate why the "De-symmetrised SHCBK" protocol of Section 5.3 cannot be weakened further to have two slaves, even when all the nodes in the protocol are trustworthy.

Suppose that there is a leader L trying to authenticate its information $INFO_L$ to two slaves A and B. In the first run α of the protocol, the intruder I impersonates slaves A and B to communicate with the leader L, and comes up with two random keys k'_A and k'_B.

$$\begin{array}{llll}
1.\alpha. & L & \longrightarrow_N I(A,B) & : INFO_L, longhash(k_L) \\
2.\alpha. & I(A) & \longrightarrow_N L & : k'_A \\
 & I(B) & \longrightarrow_N L & : k'_B \\
3.\alpha. & L & \longrightarrow_N I(A,B) & : k_L
\end{array}$$

After L sends out its own key k_L, the intruder can determine the final digest value of run α that L is going to compare over the empirical network in Message 4. Let us assume that $k_S = k'_A \oplus k'_B$. To fool slaves A and B into thinking that a fake $INFO'_L$ is authentic, the intruder needs to find k'_S such that the digests of both runs come out to be the same:

$$digest(k_L \oplus k_S, INFO_L) = digest(k_L \oplus k'_S, INFO'_L)$$

This should not take a long time as the bit-length of the digest output is short. Once he successfully searches for k'_S, he starts the second run β. In this run, he impersonates the leader L to talk to slaves A and B as well as modifying the their keys as follows

$$\begin{array}{llll}
1.\beta. & I(L) & \longrightarrow_N A, B & : INFO'_L, longhash(k_L) \\
2.\beta. & A & \longrightarrow_N I(B,L) & : k_A \\
 & B & \longrightarrow_N I(A,L) & : k_B \\
 & I(A) & \longrightarrow_N B & : k_B \oplus k'_S \\
 & I(B) & \longrightarrow_N A & : k_A \oplus k'_S \\
3.\beta. & I(L) & \longrightarrow_N A, B & : k_L
\end{array}$$

After the key k_L is revealed to A and B, all three nodes should be able to empirically agree on two equal digests that have different antecedents. In other words, the slaves accept $INFO'_L$ faked by the intruder.

$$\begin{array}{llll} 4.\beta. & A, B & \longrightarrow_E L & : digest(k_L \oplus k'_S, INFO'_L) \\ & A & \longleftrightarrow_E B & : digest(k_L \oplus k'_S, INFO'_L) \\ 4.\alpha. & L & \longrightarrow_E A, B & : digest(k_L \oplus k_S, INFO_L) \end{array}$$

The digests of all three nodes will agree despite them not agreeing on $INFO_L$.

Note that not only does the above attack work when we XOR digest sub-keys as in the SHCBK protocol, but also with any other ways to combine them. This is because the intruder will always be able to predetermine the final digest value before any slave is committed to the digest. Since a digest value is short, it is feasible for the intruder to search for digest sub-keys that map to the digest value regardless of how nodes choose to combine them.

C Improved MANA I protocols and their security analysis

In this Appendix, we will present another two versions of the Improved MANA I protocols, which are termed the *indirect binding* and Diffie-Hellman style (or D-H style) protocols.

C.1 Indirect binding and D-H style versions of the Improved MANA I protocols

An alternative solution for Improved V-MANA I is to use a commitment scheme to bind $INFO_A$ to a b-bit random nonce R, which is generated by A and released over the bounded empirical channel. This therefore makes use of the *indirect* information binding strategy, as can be seen below.

Improved version of V-MANA I (indirect binding) [35]		
1.	$A \longrightarrow_N$	$B : INFO_A, c$
		$(c, d) = \text{commit}(INFO_A, R)$
2.	$A \longrightarrow^t_{BE}$	$B : R$
3.	$A \longrightarrow_N$	$B : d$
Computational cost: $W(M + W) = 5M + 25$		

The order and time constraints of messages' arrival in this scheme must be the same as in the direct binding version of Improved V-MANA I protocol. However, the large $INFO_A$ is processed by a long-output commitment scheme, which is more expensive than a digest function, and thus the cost goes up to an order of $W(M + W) = 25 + 5M$, i.e. an approximate $(W = 5)$-fold increase compared to the direct binding version.

This protocol might be regarded as the non-interactive version of the pairwise (indirect binding) protocol of Vaudenay [53] of Section 4.2.

As in the direct binding version of the Improved MANA I protocol, we can replace the bounded delay empirical channel with a simple acknowledgement to have the following scheme.

Improved MANA I protocol (indirect binding) [35]		
$1a.$	$A \longrightarrow_N$	$B : INFO_A, c$
		$(c, d) = \text{commit}(INFO_A, R)$
$1b.$	$B \longrightarrow_E$	A : 1-bit committed signal
2.	$A \longrightarrow_E$	$B : R$
3.	$A \longrightarrow_N$	$B : d$
Computational cost: $W(M + W) = 5M + 25$		

Next we describe another improved scheme, whose structure resembles the pairwise (direct binding) authentication protocol of Hoepman [17, 18] of Section 4.1.

In the following description, k is a long secret key (160-bit) of A that corresponds to his Diffie-Hellman token g^k he wants to authenticate. In order for the following protocol to be secure, the Diffie-Hellman token g^k must be fresh at each session, unpredictable and kept secret to A when its longhash and b-bit shorthash are revealed in the first two messages.

Improved V-MANA I protocol (D-H style) [35]
1. $A \longrightarrow_N \quad B : longhash(g^k)$
2. $A \longrightarrow_{BE}^{t} \quad B : shorthash(g^k)$
3. $A \longrightarrow_N \quad B : g^k$
Computational cost: $WM + M = 6M$

The main difference between this and the direct/indirect binding versions is that there is no $INFO_A$ sent in Message 1 because the Diffie-Hellman token, revealed in Message 3, plays the dual-role of both $INFO_A$ and the long secret key. This results in a cost of order $WM + M = 6M$.

C.2 Security analysis of the Improved (V-)MANA I protocols

We adapt the Bellare-Rogaway security model where an intruder can control on which node a new protocol instance is launched. Below we define the two kinds of adversaries used in our security analysis.

1. **A general adversary** can launch multiple instances of participants (A and B in these protocols). As commonly the case in the literature, the number of times that (s)he can launch an instance of any participant is limited by a finite number, for example $\mathcal{Q}_A$ for A and $\mathcal{Q}_B$ for B. The time complexity of this adversary is bounded by a finite number say T. This is the kind of adversary we want to prove the protocols resist in the security analysis presented here.

2. **A one-shot adversary** is a special case of the general adversary where the number of each participant's instances he can launch is at most once, i.e. $\mathcal{Q}_A = \mathcal{Q}_B = 1$.

We are going to prove that the Improved (V-)MANA I protocols are secure against a one-shot attack in the first step, and then use Theorem 1 stated below to lift the one-shot attack's security ananlysis to a general attack's security analysis.

The following theorem is the combined result of Lemma 6 of [53] and Theorem 5 of [39] of Vaudenay and Pasini.

Theorem 1 [35, 39, 53] We consider a general attack such that the number of instances of A (respectively B) is at most $\mathcal{Q}_A$ (respectively $\mathcal{Q}_B$).

If there exists a one-shot attack against the three improved versions of the (V-)MANA I protocol which has success probability p in a time T, then the general attack is successful with probability $P \leq p \cdot \mathcal{Q}_A$ in a time $Q_A T$.

In the following and all subsequent security proofs, we only consider the case when the intruder cannot influence keys and random nonces, generated by party A whose instances are possibly launched by the intruder, and which are instrumental in the computation of SASs. This assumption must be made, for otherwise the intruder could easily fool B into accepting a fake $INFO'_A$ by searching for a digest or shorthash collision. Examples are long key k in the direct binding version of the Improved (V-)MANA I protocol, and short nonce R and commitment value c in the indirect binding ones. Note that we believe that the same assumption has also been made by Vaudenay in his proof of this theorem, i.e. Lemma 6 of [53].

Proof An instance of A is compatible with an instance of B if B's instance succeeded and received all messages in the right order, where Message 2 is transmitted over the empirical channels from the corresponding A's instance.

The number of possible compatible pairs of instances is upper bounded by $\mathcal{Q}_A\mathcal{Q}_B$, which can be reduced to $\mathcal{Q}_A$ in the Improved (V-)MANA I protocols because

- In the three versions of the Improved MANA I protocol, the single SAS (i.e. digest or random nonce) transmitted over empirical channels by definition in Section 2.1 cannot be mistaken, replayed or delayed from one to another session.
- In the three versions of the Improved V-MANA I protocol, B can always be offline. As a result, the intruder can simulate all instances of B and picks one who will make the attack succeed.

When an attack is successful, there should exist one compatible pair of instances of A and B which (1) have or compute the same SAS value sent over the empirical channel; and (2) do *not* share the same public data $INFO_A$ on which they try to agree.

Note that the SASs' values of all compatible pairs of instances are uniformly distributed and independent [9] from one another; because the SASs are randomised by either random keys (k in direct binding), random nonces (R in indirect binding), or random Diffie-Hellman tokens (g^k in the Diffie-Hellman style version). All of these random elements, which are instrumental in the computation of SASs, are unknown to (and are not influenced by) the intruder at the point when they were generated by A's instances thanks to the above assumption. (This argument remains true even when data $INFO_A$s are controlled by the intruder due to the use of digest functions).

We know that the probability of a successful one-shot attack on each compatible pair of instances is limited to p in a time T (e.g. a successful attack means A and B agree on the same digest or SAS of different preimage data $INFO_A$s). We further know that the SASs communicated in those compatible pairs of instances are themselves independent. We therefore have that the general adversary is successful with probability $P \leq p \cdot \mathcal{Q}_A$ in a time $Q_A T$. ▮

C.2.1 Security analysis of the direct binding Improved (V-)MANA I protocol

In the following theorem, the notation (ϵ_c, T_c)-collision-resistant indicates that the success probability of finding a hash collision is upper bounded by ϵ_c in a time T_c. Similarly (ϵ_i, T_i)-inversion-resistant indicates that the success probability of inverting a hash value is upper bounded by ϵ_i in a time T_i.

Theorem 2 [35] Given that *longhash*() is (ϵ_c, T_c)-collision-resistant and (ϵ_i, T_i)-inversion-resistant, a general attack with number of A's (respectively B's) instances bounded by $\mathcal{Q}_A$ (respectively $\mathcal{Q}_B$) is successful against the direct binding versions of Improved (V-)MANA protocol with probability $2^{-b}\mathcal{Q}_A(1 + \epsilon_i + \epsilon_c)$ in a time $Q_A(T_i + T_c)$.

The following proof applies to the direct binding version of the Improved V-MANA I protocol, but it can be slightly modified to cope with the direct binding version of the Improved MANA I protocol.

[9] See Footnote 4 for what independence means.

Proof We first find the probability of a successful one-shot attack.

A one-shot intruder has no advantage of sending fake $INFO'_A$ and $longhash(k')$ to B (masquerading as A) after the digest is released in Message 2. Therefore, after $INFO_A$ and $longhash(k)$ are sent in Message 1 where k is a private, fresh and long (160-bit) key generated by A in each session and is unknown to any one including the intruder, there are three possibilities that can happen: (1) with probability ϵ_c the intruder can find a hash collision in a time T_c; (2) with probability ϵ_i the intruder can invert the hash value in a time T_i; and (3) with probability $1 - \epsilon_c - \epsilon_i$ neither can the intruder find a hash collision nor invert the hash value. Note that the above analysis is only correct when we assume that given any $INFO_A$ and $longhash(k)$, it is infeasible to gain any advantage in predicting the value of $digest(k, INFO_A)$ without the knowledge of key k, i.e. the digest value should be uniformly distributed even in the presence of $INFO_A$ and $longhash(k)$. There is no need to consider the 2nd-preimage-resistance property of a hash function since the intruder does not know key k generated by the honest party A in Message 1.

1. With probability ϵ_c in a time T_c, the adversary can search (off-line) for two distinct keys k' and k'' for which $longhash(k') = longhash(k'')$. The adversary then sends an arbitrarily data $INFO'_A$ ($INFO'_A \neq INFO_A$) and $longhash(k')$ to B (masquerading as A).

   Game against the Improved V-MANA I (direct binding)– hash collision

1.	A	$\longrightarrow_N$	$I(B)$	$: INFO_A, longhash(k)$
	$I(A)$	$\longrightarrow_N$	B	$: INFO'_A, longhash(k')$
2.	A	$\longrightarrow_{SE}$	B	$: digest(k, INFO_A)$
3.	A	$\longrightarrow_N$	$I(B)$	$: k$

   Winning condition: $digest(k, INFO_A) = digest(k', INFO'_A)$ or $digest(k, INFO_A) = digest(k'', INFO'_A)$

   Prior to sending a key to B in Message 3 the adversary checks to see whether or not $digest(k, INFO_A) = digest(k', INFO'_A)$, and/or $digest(k, INFO_A) = digest(k'', INFO'_A)$. In the first case (which has probability 2^{-b}), the adversary sends k' to B. In the second case (which also has probability 2^{-b}), the adversary sends k'' to B. We conclude that a one-shot attack has probability $2\epsilon_c 2^{-b}$ of success in a time T_c.

2. With probability ϵ_i in a time T_i, the adversary can find a preimage k' such that $longhash(k') = longhash(k)$. The adversary then replaces $INFO_A$ with an arbitrarily data $INFO'_A$ ($INFO'_A \neq INFO_A$) in Message 1.

   Game against the Improved V-MANA I (direct binding)– hash inversion

1.	A	$\longrightarrow_N$	$I(B)$	$: INFO_A, longhash(k)$
	$I(A)$	$\longrightarrow_N$	B	$: INFO'_A, longhash(k)$
2.	A	$\longrightarrow_{SE}$	B	$: digest(k, INFO_A)$
3.	A	$\longrightarrow_N$	$I(B)$	$: k$

   Winning condition: $digest(k, INFO_A) = digest(k, INFO'_A)$ or $digest(k, INFO_A) = digest(k', INFO'_A)$

   Prior to sending a key to B the adversary checks to see whether or not $digest(k, INFO_A) = digest(k, INFO'_A)$, and/or $digest(k, INFO_A) = digest(k', INFO'_A)$. As in the previous case, a one-shot attack has probability $2\epsilon_i 2^{-b}$ of success in a time T_i.

3. On the other hand, with probability $1-\epsilon_i-\epsilon_c$ in a time T_i+T_c neither can the adversary search for a hash collision nor invert the hash value. Thus the adversary has to select a random pair $(k', INFO'_A)$ where $INFO_A \neq INFO'_A$.

Game against Improved V-MANA I (direct binding) No hash collision and no hash inversion				
1.	A	$\longrightarrow_N$	$I(B)$	$: INFO_A, longhash(k)$
	$I(A)$	$\longrightarrow_N$	B	$: INFO'_A, longhash(k')$
2.	A	$\longrightarrow_E$	B	$: digest(k, INFO_A)$
3.	A	$\longrightarrow_N$	$I(B)$	$: k$
	$I(A)$	$\longrightarrow_N$	B	$: k'$
Winning condition: $INFO_A \neq INFO'_A$ and $digest(k, INFO_A) = digest(k', INFO'_A)$				

Clearly, the probability of success of this case is $(1-\epsilon_i-\epsilon_c)2^{-b}$ in a time T_i+T_c thanks to the digest specification.

We conclude that any one-shot adversary in a time T_i+T_c has the following probability of success

$$p \leq 2\epsilon_c 2^{-b} + 2\epsilon_i 2^{-b} + (1-\epsilon_c-\epsilon_i)2^{-b} = 2^{-b}(1+\epsilon_c+\epsilon_i)$$

We now can apply Theorem 1 to deduce that any general adversary has probability $2^{-b}Q_A(1+\epsilon_c+\epsilon_i)$ of success in a time $Q_A(T_i+T_c)$. ▮

C.2.2 Security analysis of the indirect binding Improved (V-)MANA I protocol

Theorem 3 [35] Given that a commitment scheme is (ϵ_h, T_h)-hiding and (ϵ_b, T_b)-binding, a general attack with number of A's (respectively B's) instances bounded by Q_A (respectively Q_B) is successful against the indirect binding versions of Improved (V-)MANA protocols with probability $(\epsilon_h+\epsilon_b)Q_A$ in a time $Q_A(T_b+T_h)$.

Proof There are two possibilities that a one-shot attacker can do after receiving $INFO$ and c in Message 1 from A:

- Leaving c unchanged, the intruder sends $INFO'_A$ and c to B (masquerading as A) where $INFO'_A \neq INFO_A$. With probability ϵ_b in a time T_b, the intruder can come up with a d' (which can be either the same as or different from d revealed in Message 3) such that $open(INFO'_A, c, d') = R$ thanks to the binding property of a commitment scheme.
- With probability ϵ_h in a time T_h, the intruder can guess the value of R from $INFO_A$ and c, and then compute (c', d') such that $open(INFO'_A, c', d') = R$ thanks to the hiding property of a commitment scheme.

We can apply Theorem 1 to deduce that any general intruder has a success probability $Q_A(\epsilon_b+\epsilon_h)$ in a time $Q_A(T_h+T_b)$. ▮

C.2.3 Security analysis of the Improved V-MANA I protocol in Diffie-Hellman style

Theorem 4 [35] Given that $longhash()$ is (ϵ_c, T_c)-collision-resistant and (ϵ_i, T_i)-inversion-resistant, a general attack with number of A's (respectively B's) instances bounded by Q_A (respectively Q_B) is successful against the Improved V-MANA I protocol in Diffie-Hellman (D-H) style with probability $2^{-b}Q_A(1+\epsilon_c)$ in a time $Q_A(T_c+T_i)$.

Proof As in the proof of Theorem 2, there are three possibilities which can happen after A releases Message 1 and we also assume that given $longhash(g^k)$ it is infeasible for the intruder to gain any advantage in predicting the value of $shorthash(g^k)$.

1. With probability ϵ_c in a time T_c, the adversary can search for two distinct D-H tokens $g^{k'}$ and $g^{k''}$ for which $longhash(g^{k'}) = longhash(g^{k''})$. The adversary then sends $longhash(g^{k'})$ to B (masquerading as A).

Game against the Improved V-MANA I (D-H style)– hash collision				
1.	A	$\longrightarrow_N$	$I(B)$	$: longhash(g^k)$
	$I(A)$	$\longrightarrow_N$	B	$: longhash(g^{k'})$
2.	A	$\longrightarrow_{SE}$	B	$: shorthash(g^k)$
3.	A	$\longrightarrow_N$	$I(B)$	$: g^k$
Winning condition: $shorthash(g^k) = shorthash(g^{k'})$ or $shorthash(g^k) = shorthash(g^{k''})$				

A one-shot attack has probability $2\epsilon_h 2^{-b}$ of success in a time T_c.

2. With probability ϵ_i in a time T_i, the adversary can find a preimage $g^{k'}$ such that $longhash(g^k) = longhash(g^{k'})$. The adversary then replaces g^k with $g^{k'}$ in Message 3 and hopes that they produce the same b-bit hash output. Therefore, the probability of success is $\epsilon_i 2^{-b}$ in a time T_i.

Game against the Improved V-MANA I (D-H style)– hash inversion				
1.	A	$\longrightarrow_N$	B	$: longhash(g^k)$
2.	A	$\longrightarrow_{SE}$	B	$: shorthash(g^k)$
3.	A	$\longrightarrow_N$	$I(B)$	$: g^k$
	$I(A)$	$\longrightarrow_N$	B	$: g^{k'}$
Winning condition: $shorthash(g^k) = shorthash(g^{k'})$				

3. On the other hand, with probability $1 - \epsilon_i - \epsilon_c$ in a time $T_i + T_c$ neither can the adversary search for a hash collision nor invert the hash value. Thus the adversary has to select a random D-H token $g^{k'}$ and send $longhash(g^{k'})$ to B in Message 1 (masquerading as A).

Game against Improved V-MANA I (D-H style) No hash collision and no hash inversion				
1.	A	$\longrightarrow_N$	$I(B)$	$: longhash(g^k)$
	$I(A)$	$\longrightarrow_N$	B	$: longhash(g^{k'})$
2.	A	$\longrightarrow_{SE}$	B	$: shorthash(g^k)$
3.	A	$\longrightarrow_N$	$I(B)$	$: g^k$
	$I(A)$	$\longrightarrow_N$	B	$: g^{k'}$
Winning condition: $shorthash(g^k) = shorthash(g^{k'})$				

Clearly, the probability of success of this case is $(1 - \epsilon_i - \epsilon_c)2^{-b}$.

We conclude that any one-shot adversary in a time $T_i + T_c$ has the following probability of success

$$p \leq 2\epsilon_c 2^{-b} + \epsilon_i 2^{-b} + (1 - \epsilon_c - \epsilon_i)2^{-b} = 2^{-b}(1 + \epsilon_c)$$

We now can apply Theorem 1 to deduce that any general adversary has a success probability $2^{-b}Q_A(1+\epsilon_c)$ in a time $Q_A(T_i+T_c)$. ▮

References

[1] See: http://www.prismmodelchecker.org/

[2] *Simple Pairing Whitepaper*, Bluetooth Special Interest Group, 2006. See: www.bluetooth.com/NR/rdonlyres/ 0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/SimplePairing_WP_V10r00.pdf

[3] D. Balfanz, D. Smetters, P. Stewart and H. Wong, Talking to strangers: Authentication in ad-hoc wireless networks, in: *Proceedings of the 9th Annual Symposium on Network and Distributed System Security (NDSS)*, 2002.

[4] M. Bellare and P. Rogaway, Entity authentication and key distribution, in: *Advances in Cryptology - Crypto 1993*, Lecture Notes in Computer Science, Vol. 773, D.R. Stinson, ed., Springer, 1993, pp. 232-249.

[5] J. Bierbrauer, T. Johansson, G.A. Kabatianskii and B.J.M. Smeets, On families of hash functions via geometric codes and concatenation, in: *Advances in Cryptology - Crypto 1993*, Lecture Notes in Computer Science, Vol. 773, D.R. Stinson, ed., Springer, 1993, pp. 331-342.

[6] M. Čagalj, S. Čapkun and J. Hubaux, Key agreement in peer-to-peer wireless networks, in: *Proceedings of the IEEE Special Issue on Security and Cryptography* **94**(2) (2006), A. Mazzeo, ed., 467-478.

[7] J.L. Carter and M.N. Wegman, Universal classes of hash functions, *Journal of Computer and System Sciences* **18**(2) (1979), 143-154.

[8] D. Chaum, Secret-ballot receipts: true voter-verifiable elections, *Security and Privacy Magazine*, IEEE, **2**(1) (2004), 38-47.

[9] S.J. Creese, M.H. Goldsmith, R. Harrison, A.W. Roscoe, P. Whittaker and I. Zakiuddin, Exploiting empirical engagement in authentication protocol design, in: *Proceedings of the 2nd International Conference on Security in Pervasive Computing (SPC 2005)*, Lecture Notes in Computer Science, Vol. 3450, D. Hutter and M. Ullmann, eds., Springer, 2005, pp. 119-133.

[10] S.J. Creese, M.H. Goldsmith, A.W. Roscoe and M. Xiao, Bootstrapping multi-party ad-hoc security, in: *Proceedings of the 2006 ACM Symposium on Applied Computing*, H.M. Haddad, ed., 2006, pp. 369-375.

[11] S.J. Creese, M.H. Goldsmith, A.W. Roscoe and I. Zakiuddin, The attacker in ubiquitous computing environments: Formalising the threat model, in: *Proceedings of the 1st Workshop on Formal Aspects in Security and Trust*, 2003. IIT-CNR Technical Report.

[12] S.J. Creese, M.H. Goldsmith, A.W. Roscoe and I. Zakiuddin, Security properties and mechanisms in human-centric computing, in: *Proceedings of Workshop on Security and Privacy in Pervasive Computing*, 2004.

[13] C. Gehrmann, C. Mitchell and K. Nyberg, Manual authentication for wireless devices, *RSA Cryptobytes*, **7**(1) (2004), 29-37.

[14] C. Gehrmann and K. Nyberg, Security in personal area networks, in: *Security for Mobility*, C.J. Mitchell, ed., IEE, London, 2004, pp. 191-230.

[15] ISO/IEC 9798-6, C. Mitchell, ed., 2003, *Information technology – Security techniques – Entity authentication – Part 6: Mechanisms using manual data transfer.*

[16] M.T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik and E. Uzun, Loud and clear: Human-verifiable authentication based on audio, in: *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006)*, 2006, pp. 10-33.

[17] J.-H. Hoepman, Ephemeral pairing on anonymous networks, in: *Proceedings of the 2nd International Conference on Security in Pervasive Computing (SPC 2005)*, Lecture Notes in Computer Science, Vol. 3450, D. Hutter and M. Ullmann, eds., Springer, 2005, pp. 101-116.

[18] J.-H. Hoepman, Ephemeral pairing problem, in: *Proceeding of the 8th International Conference on Financial Cryptography*, Lecture Notes in Computer Science, Vol. 3110, A. Juels, ed., Springer, 2004, pp. 212-226.

[19] G.A. Kabatianskii, B. Smeets and T. Johansson, On the cardinality of systematic authentication codes via error-correcting codes, *IEEE Transactions on Information Theory*, **42**(2) (1996), 566-578.

[20] H. Krawczyk, LFSR-based hashing and authentication, in: *Advances in Cryptology - Crypto 1994*, Lecture Notes in Computer Science, Vol. 839, Y. Desmedt, ed., Springer, 1994, pp. 129-139.

[21] H. Krawczyk, New hash functions for message authentication, in: *Advances in Cryptology - Eurocrypt 1995*, Lecture Notes in Computer Science, Vol. 921, L.C. Guillou and J.-J. Quisquater, eds., Springer, 1995, pp. 301-310.

[22] S. Laur and K. Nyberg, Efficient mutual data authentication using manually authenticated strings, in: *Proceedings of the 5th International Conference on Cryptology and Network Security (CANS 2006)*, Lecture Notes in Computer Science, Vol. 4301, D. Pointcheval, ed., Springer, 2006, pp. 90-107.

[23] S. Laur, N. Asokan and K. Nyberg, Efficient mutual data authentication using manually authenticated strings: Extended version, in: *Cryptology ePrint Archive*, Report 2005/424, 2006.

[24] A.Y. Lindell, Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1, in: *Proceedings of the Cryptographers' Track at the RSA Conference 2009 on Topics in Cryptology*, Lecture Notes in Computer Science, Vol. 5473, M. Fischlin, ed., Springer, 2009, pp. 66-83.

[25] A. Madhavapeddy, D. Scott, R. Sharp and E. Upton, Using camera phones to enhance human-computer interaction, in: *Proceedings of the 6th International Conference on Ubiquitous Computing, (UbiComp 2004)*, 2004, pp. 1-2.

[26] Y. Mansour, N. Nisan and P. Tiwari, The computational complexity of universal hashing, in: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990, pp. 235-243.

[27] A. Mashatan and D.R. Stinson, Non-interactive two-channel message authentication based on hybrid-collision resistant hash functions, in: *IET Information Security* **1**(3) (2007), 111-118.

[28] R. Mayrhofer and M. Welch, A human-verifiable authentication protocol using visible laser light, in: *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES)*, 2007, pp. 1143-1148.

[29] J.M. McCune, A. Perrig and M.K. Reiter, Seeing is believing: Using camera phones for human-verifiable authentication, *International Journal of Security and Networks* **4**(1/2) (2009), 43-56.

[30] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, ISBN: 0-8493-8523-7, 1996.

[31] T. Moran and M. Naor, Polling with physical envelopes: A rigorous analysis of a human-centric protocol, in: *Advances in Cryptology - Eurocrypt 2006*, Lecture Notes in Computer Science, Vol. 4004, S. Vaudenay, ed., Springer, 2006, pp. 88-108.

[32] L.H. Nguyen and A.W. Roscoe, Efficient group authentication protocol based on human interaction, in: *Proceedings of the Joint Workshop on Foundation of Computer Security and Automated Reasoning Protocol Security Analysis (FCS-ARSPA 2006)*, 2006, pp. 9-31.

[33] L.H. Nguyen and A.W. Roscoe, Authenticating ad-hoc networks by comparison of short digests, *Information and Computation* **206**(2-4) (2008), 250-271.

[34] L.H. Nguyen and A.W. Roscoe, New combinatorial bounds for universal families of hash functions, in: *Cryptology ePrint Archive*, Report 2009/153, 2009.

[35] L.H. Nguyen and A.W. Roscoe, Separating two roles of hashing in one-way message authentication, in: *Proceedings of the Joint Workshop on Foundations of Computer Security, Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (FCS-ARSPA-WITS 2008)*, 2008, pp. 195-210.

[36] L.H. Nguyen, *Authentication protocols in pervasive computing*, D.Phil. Thesis, University of Oxford, 2010.

[37] ISO/IEC 9798-6 (revision), L.H. Nguyen, ed., 2010, *Information Technology – Security Techniques – Entity authentication – Part 6: Mechanisms using manual data transfer.*

[38] S. Pasini and S. Vaudenay, SAS-based authenticated key agreement, in: *Proceedings of the 9th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2006)*, Lecture Notes in Computer Science, Vol. 3958, M. Yung, Y. Dodis, A. Kiayias and T. Malkin, eds., Springer, 2006, pp. 395-409.

[39] S. Pasini and S. Vaudenay, An optimal non-interactive message authentication protocol, in: *Proceedings of the Cryptographers' Track at the RSA Conference 2006 on Topics in Cryptology*, Lecture Notes in Computer Science, Vol. 3860, D. Pointcheval, ed., Springer, 2006, pp. 280-294.

[40] R. Pass, On deniability in the common reference string and random oracle model, in: *Advances in Cryptology - Crypto 2003*, Lecture Notes in Computer Science, Vol. 2729, D. Boneh, ed., Springer, 2003, pp. 316-337.

[41] A.W. Roscoe, Human-centred computer security, 2005. See:
http://web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications/113.pdf.

[42] A.W. Roscoe and L.H. Nguyen, *Security in computing networks*, WO Patent 2007052045.

[43] A.W. Roscoe, B. Chen and L.H. Nguyen, *Improvements in communications security*, WO Patent 2008078101.

[44] A.W. Roscoe and L.H. Nguyen, *Improvements related to the authentication of messages*, WO Patent 2009153585.

[45] N. Saxena, J.-E. Ekberg, K. Kostiainen and N. Asokan, Secure device pairing based on a visual channel, in: *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, 2006, pp. 306-313.

[46] N. Smart, *Cryptography, An Introduction*, McGraw-Hill, ISBN 0-0770-9987-7 (PB), 2002.

[47] F. Stajano and R. Anderson, The resurrecting duckling: Security issues for ad-hoc wireless networks, in: *Proceedings of the 7th International Workshop on Security Protocols*, Lecture Notes in Computer Science, Vol. 1796, B. Christianson, B. Crispo, J. A. Malcolm and M. Roe, eds., Springer, 1999, pp. 172-194.

[48] F. Stajano and R. Anderson, The cocaine auction protocol: on the power of anonymous broadcast, in: *Proceedings of the 3rd International Workshop on Information Hiding*, Lecture Notes in Computer Science, Vol. 1768, A. Pfitzmann, ed., Springer, 2000, pp. 434-447.

[49] D.R. Stinson, Universal hashing and authentication codes, in: *Advances in Cryptology - Crypto 1991*, Lecture Notes in Computer Science, Vol. 576, J. Feigenbaum, ed., Springer, 1992, pp. 74-85.

[50] J. Suomalainen, J. Valkonen and N. Asokan, Security associations in personal networks: A comparative analysis, in: *Proceedings of the 4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks 2007*, Lecture Notes in Computer Science, Vol. 4572, F. Stajano, C. Meadows, S. Capkun and T. Moore, eds., Springer, 2007, pp. 43-57.

[51] E. Uzun, K. Karvonen and N. Asonka, Usability analysis of secure pairing methods, in: *Proceedings of the 11th International Conference on Financial Cryptography and Data Security (FC 2007) and the 1st International Workshop on Usable Security (USEC 2007)*, Lecture Notes in Computer Science, Vol. 4886, S. Dietrich and R. Dhamija, eds., Springer, 2008, pp. 307-324.

[52] J. Valkonen, N. Asokan and K. Nyberg, Ad-hoc security associations for groups, in: *Proceedings of the 3rd European Workshop on Security and Privacy in Ad-hoc and Sensor Networks*, Lecture Notes in Computer Science, Vol. 4357, L. Butty, V.D. Gligor and D. Westhoff, eds., Springer, 2006, pp. 150-164.

[53] S. Vaudenay, Secure communications over insecure channels based on short authenticated strings, in: *Advances in Cryptology - Crypto 2005*, Lecture Notes in Computer Science, Vol. 3621, V. Shoup, ed., Springer, 2005, pp. 309-326.

[54] 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) (1981), 265-279.

[55] A.F. Webster and S.E. Tavares, On the design of S-Boxes, in: *Advances in Cryptology - Crypto 1985*, Lecture Notes in Computer Science, Vol. 218, H.C. Williams, ed., Springer, 1986, pp. 523-534.

[56] F.-L. Wong and F. Stajano, Multi-channel protocols, in: *Proceedings of the 13th International Workshop on Security Protocols*, Lecture Notes in Computer Science, Vol. 4631, B. Christianson, B. Crispo, J.A. Malcolm and M. Roe, eds., Springer, 2005, pp. 128-132.

[57] F.-L. Wong and F. Stajano, Multi-channel security protocols, *IEEE Pervasive Computing* **6**(4) (2007), 31-39.