

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

L.H. Nguyen and A.W. Roscoe
Oxford University Computing Laboratory
E-mail addresses: {Long.Nguyen, Bill.Roscoe@comlab.ox.ac.uk}

December 21, 2007

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 through 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. We introduce a number of new protocols, most of which are variants on existing ones.

1 Introduction

1.1 Outline of paper

In this paper, we give a survey of authentication protocols based on manual handling or comparisons of short authentication strings (SASs) over an assumed empirical 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 the publication of a variety of such protocols, frequently by groups working completely independently of each other [55, 15, 58, 36, 11, 20, 54]. Given the potential importance of this work, we feel that our survey is timely.¹ 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. 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 make the paper easier to read and understand, 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 1.3. A simple model for the computation cost of these cryptographic primitives is also provided to assess the complexity of these protocols as we move along.

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 Gehrman, Mitchell and Nyberg [15, 16, 22]. 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 3, 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. The main distinction we make here is between protocols that bind authenticated information *indirectly* or *directly* to the compared SAS. We study these two approaches in Sections 3.2 and 3.3 respectively. We find that direct binding

¹There has been another survey written by Suomalainen et al. [52] where the authors only concentrate on pairwise protocols bootstrapping security from scratch by either human interaction or secret shared passwords. As we will see, it is also significantly different from ours in a number of ways: we concentrate on one-way, mutual and group protocols based on human interaction, and classify and analyse them in term of information binding strategies and computational efficiency.

has a clear advantage which arises because of 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.

While there has been much recent literature on pairwise protocols, we find it strange that apart from the authors' group [11, 45, 45, 36, 37] and Valkonen et al. [54] there has been little on group protocols, although there appear to be many potential applications of these. We will discuss group protocols in Section 4 as well as presenting a number of newly invented and modified versions aiming to further improve the processing cost.

Finally, in Section 5.1, the computation cost of every protocol will be gathered into three tables that clearly demonstrate two things: that interactive schemes can be much more efficient in human work than non-interactive ones, and that computation efficiency is improved by the use of direct binding compared to indirect binding in both pairwise and group protocols.

Most 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: \square .
- Protocols that have been modified in minor ways, often by replacing one or more cryptographic primitive or other data operation, are cited \square^* .
- Protocols that are either major modifications to existing ones or just new are marked *New*.

1.2 Notation

Capital letters such as A , B , C , I , and S are used to identify parties, and $\forall A$ (or A') means that a message is sent or received by all parties in a group \mathbf{G} attempting to bootstrap 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) .

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 papers vary is in the assumptions they make

about the empirical channels. In this paper we use different notations for communications over a normal Dolev-Yao (insecure) channel and those over four types of empirical ones.

- \longrightarrow_N , the normal Dolev-Yao network where all messages transmitted between the laptops in this channel can be overheard, deleted or modified by the intruder.
- \longrightarrow_{WE} , this *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 [55, 42].
- \longrightarrow_E is the type of empirical channel assumed in [36, 37]. It is like a weak empirical channel except that it cannot be replayed. It can be delayed, but not sufficiently long so that a message from one session can be used in another. 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.
- \longrightarrow_{SE} , this 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. We term this a *strong empirical channel* (or a strong authentication channel). This was also defined in [55, 42].
- \longrightarrow_{BE}^t , is the same as \longrightarrow_E except that messages cannot take more than time t to arrive. 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*.

We will assume each node A in a group \mathbf{G} of N parties has some information $INFO_A$ of length K bits ($= K/32 = M$ words) 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;
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 network². In addition, we refer to $INFOS$ as the concatenation in an alphabet order of all the distinct pairs parties want to authenticate: NM words if they are all size M .

We note that if $INFOS$ contains N photographs or similar, it may well be of significant size.

1.3 Cryptographic primitives

We will be using a variety of cryptographic primitives related to hashing in this paper. Standard cryptographic hashes (such as SHA, MD5 or TIGER) intended to be inversion- and strongly collision resistant will be denoted $longhash(X)$: for the purposes of calculation, in common with some other papers, we will generally assume these have at least 160 bits. $hash(X)$ will be a function, perhaps, with sufficiently many bits to offer weak or short-term versions of these properties, perhaps down to 80 bits, or even 16 bits in Hoepman's protocol. $digest(k, X)$ will be a short universal hash or digest of X keyed by k – the specification and purpose of this function will be discussed at length later, as well as an additional property often required of the short $hash()$. These functions will sometimes be subscripted with the number of output bits.

Usually implemented using hashing, we will also use a non-deterministic commitment scheme, whose definition is given below.

1.3.1 Commitment scheme

A commitment scheme often consists of a relation and a partial function

$$c \parallel d \in commit(INFO, R) \text{ and } INFO \parallel R = open(c, d)$$

²We 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.

though the notation $\text{commit}(\text{INFO}, R)$ is usually used as though it were a function (perhaps nondeterministic in the sense discussed below). The intention here is that R is a random nonce invented by one of the parties and which is secret only to him. He intends to bind R and INFO together without revealing R by publishing c (the *commitment*). Eventually sending d (the *decommitment*) reveals R , and binds this value firmly to INFO in the eyes of the receiver.

A commitment scheme binds some information INFO and the random nonce R so that the sender is unable to change either value it has committed to once the commitment stage is over, this property is called *binding*. In other words, given c, d, INFO and R in the above relation, it must be infeasible to compute d', INFO' and R' where $(\text{INFO}, R) \neq (\text{INFO}', R')$ which are also in the same relation with c . And it is not possible for the intruder to determine from c the value of either INFO or R before the revealing stage (when d is made public). Neither can it determine one of these values given the other and c . This property is termed *hiding*. In some cases in this paper, R is absent: in that case $c \parallel d$ will just commit INFO .

In the style of protocol we are considering, R will frequently contain too little entropy³ to prevent combinatorial attacks if $\text{commit}(\text{INFO}, R)$ is a function. We therefore permit $\text{commit}(\text{INFO}, R)$ to be many-valued, with each possible $c \parallel d$ pair committing to the same pair (INFO, R) . One natural way of achieving this is to extend R by a randomly chosen secret nonce R' so that the combination (R, R') does have enough entropy and make commit a function of (INFO, R, R') . open is a partial function in the sense that only a matching $c \parallel d$ will produce a result. In some cases INFO will already have been transmitted to the receiver by the point of the decommitment, and if so we might want to write $R = \text{open}(\text{INFO}, c, d)$ for the highly partial function that delivers R when and only when $c \parallel d$ opens to this particular INFO and some R . We require that the commitment scheme is at least as secure as a standard hash function, and therefore each of the values c, d and (R, R') need to have the same bit-length as the output of a secure cryptographic hash function. (In [55] it is assumed to be 160 bits).

One simple example of such a commitment scheme, proposed in [43] by Pass, is described here. On inputting public INFO and a short random secret key R of b bits, the algorithm will pick another secret random nonce

³In some protocols such as the non-interactive Pasini-Vaudenay scheme discussed in Section 2.1, the random nonce R can be completely ignored in the notation. In such cases we assume that R is 0-bit.

$R' \in_{\text{random}} \{0, 1\}^{160-b}$. It then sets (for the second style of *open*, where *INFO* is known to the receiver) $d = R \parallel R'$ and $c = \text{longhash}(\text{INFO} \parallel d)$. *commit*(*INFO*, *R*) is then $c \parallel d$. To use *open*(*INFO*, *c*, *d*) = *R* one simply extracts *R* from *d* and verifies $c = \text{longhash}(\text{INFO} \parallel d)$. For the first style of *open* (where the receiver does not already know *INFO*), we would have to set $d' = R \parallel R' \parallel \text{INFO}$ or $c' = \text{longhash}(\text{INFO} \parallel R \parallel R') \parallel \text{INFO}$.

1.3.2 Short hash and digest functions

A normal cryptographic hash function is chosen so that it has enough bits to be essentially immune to combinatorial attacks such as the birthday attacks. In this paper, however, we will see various cunning uses of functions that, like (long) hashes, are intended to randomise and convey no useful information about the preimage, but which do not have enough bits for conventional applications. These uses will always produce values of SASs that are to be transmitted over the empirical channel: the *reason* for the function having a short output is that it is unattractive to transmit a longer value along this channel, which we expect to be implemented by human beings.

We will see two variants on this idea: the simpler is what we call a short hash: *hash*() or *hash_b*(). This has a single argument and is intended to be uniformly or near-uniformly distributed over its *b*-bit range as its argument varies. Due to the short output, properties such as collision and inversion resistance are not relevant, and instead what is required is the *strict avalanche criterion* which states any number of bit-changes in the input has an equal influence on every bit of the output [1].

In some of the protocols we consider (specifically the ones where *INFOS* is directly bound in the SAS) we need to construct a digest of the combination of *INFOS* and some key *k*. This digest cannot have the usually specified properties of a cryptographic hash, namely non-invertability and collision-freeness, due to its short output of perhaps 16–32 bits. On the other hand we do require that a high degree of randomness arises from the use of the key *k*. Specifically, in order not to degrade the efficiency of a protocol in terms of the amount of empirical work required, it must satisfy, or satisfy to within $\epsilon \ll 2^{-b}$, the following specification defined in [36, 37]. As the key *k* varies uniformly over its range:

1. *digest*(*k*, *m*) is uniformly distributed for any fixed *m*.
2. And for any fixed θ and $m \neq m'$:

$$\Pr(\text{digest}(k, m) = \text{digest}(k \oplus \theta, m')) \leq 2^{-b}$$

The rationale for these two specifications, especially the use of $\oplus\theta$, will become apparent when we analyse the group protocols presented in Section 4. These are required for all uses of digest functions in this paper, and also have similarities with the ideas of *universal hash functions* (or a family of hash functions) and the *strict avalanche property* in spite of being more restrictive than universal hash since digest collisions with respect to different keys are also considered.

As we will see in the majority of direct binding protocols presented in this paper, the SAS is often the output of a digest function. For this reason, there have been a number of algorithms proposed to compute the digest [22, 41, 27, 28, 16, 5, 37], but as far as we are aware of, the only ones that can be efficiently computed as well as being proved to satisfy the above specification exactly are based on the idealised framework invented by the authors and described in [36, 37]. We there provide several algorithms based on random numbers derived from the digest key, which can be simulated in practice by a pseudo-random number generation (PRNG). The most efficient are based on Toeplitz matrices of bits or words: ones in which all elements of each diagonal are equal, but where different diagonals are independent and uniformly distributed. These justify the advantage claimed for short output digests in the section below on the computation cost model. Our algorithms have no restriction on the length of the object (typically *INFOS*) being digested.

In contrast, [41, 27, 16, 22] make use of universal hash functions presented in [51, 24, 7] that put an upper bound on the input length, and consequently they have to compress a long message into a fixed number of bits (say 512 or 256 bits) initially by using a cryptographic hash prior to running the algorithm itself, which turns out to be neither ideally secure nor cost effective, [37]. Alternatively, others [5, 41] suggest using first or last b bits of a hash of a large message, which is inefficient and does not necessarily have the precise property we need of being an ideal digest.

Throughout this paper we will be trying to optimise the amount of security one can obtain from a given amount of empirical (human) communication. We can expect this to be most efficient when the only thing communicated is a single digest value: there we can hope for a security level (i.e. maximum probability of any attacker strategy being successful) of 2^{-b} for b bits of communication. We will see in later sections that, provided we can discount the probability of strong cryptographic devices being broken, this bound is attainable. It is easy to argue one cannot do better: if an attacker chooses an arbitrary *INFOS'* to substitute for the one that a set of nodes actually want to agree, if they only communicate $c < b$ bits empirically,

there will be a 2^{-c} chance that the bits they actually communicate coincide with the ones they would have created for *INFOS'*.

1.4 Computation cost model of cryptographic primitives

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 (termed a *longhash()*), a (shorter) *hash()* function, a digest, and a commitment scheme.

Let W and B be the number of words (32 bits) and respectively bits required to hold a long hash value: it is normal that $B = 160$ bits, so we assume $W = 5$. Many researchers in [55, 15, 37, 59] also suggest 15 or 16 bits are reasonable choices for b , the width of the digest output which is rounded up to 1 word in this case. We also assume that nonces and keys (used in a commitment scheme and *longhash()*) and other strong cryptographic values such as a commitment (c) and a decommitment (d) have the same bit-length B .

It is clear that the cost of computing the b -bit output $hash_b(m)$ increases linearly with the length of m . It also seems clear that it will increase significantly with b , and a simple model in which each word of a running temporary value of length b is combined with each input word suggests our overall model might be $b \times length(m)$. Therefore we will adopt that assumption in the following analysis. 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.⁴

With respect to the cost of computing a digest, as defined in Section 1.3.2, *digest(-, -)* is a family of short hash functions where each of them is indexed by a key k . Even though the key bit-length might be significantly longer than the output, according to the idealised framework [37], it is only used to derive random bits. The length itself does not play any part in the digest computation, and therefore will not have a big impact on the computation cost.⁵ As a result of this, we can assume the cost model of a digest is similar to a hash function which is mainly dependent on the lengths of the input message and the digest output as discussed in the previous paragraph. We

⁴In practice, one often constructs variable output-size hash function based on the idea of Key Generating Function (KGF). For example, given a 160-bit output hash function such as SHA or MD5, we can use concatenation operator to construct a $160 \times t$ -output hash function as follows: $HASH(m) = hash(1, m) \parallel hash(2, m) \parallel \dots \parallel hash(t, m)$. This of course clearly follows our computational cost model.

⁵In fact the longer is the key the fewer the number of random bits we have to generate, and subsequently the better.

also have experimented results which justify the difference in computation cost w.r.t the computational model between the digest function and SHA-1. We will discuss the issues in our subsequent paper on digest function [40].

We will assume that the commitment scheme, used to commit a message m of length M words and a nonce R of length b bits = 1 word, takes $\max(M, W)$ words as input. Since the security of the scheme is equivalent to a cryptographic hash, as pointed out in Section 1.3.1, it requires randomisation that introduces additional nondeterminism to that introduced by R . This is equivalent to adding a hidden random variable of length $W - \lceil \frac{b}{32} \rceil = W - 1$ in word. For ease of calculation, we assume $M \geq W$.

Every protocol in this paper except the first one in Section 4.1 only uses long/short-hashes, 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. And then in Section 5.1 all the calculation results will be put together into three different tables that summarise the efficiency of this class of protocols.

2 Non-interactive protocols

We now examine some protocols which attempt to transmit a (possibly very long) message from one party to another in such a way that the origin of the message is 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 one-way authentication protocol which works where there is a PKI. Here a message M and the name of the sender A are accompanied by the message authentication code (MAC) $\{longhash(M)\}_{sk(A)}$ and possibly the public-key certificate of the sender A . The receiver knows (up to the infinitesimal degree of uncertainty allowed in cryptographic reasoning) M really is from A because he can form the cryptographic hash of M and discover if it really was A who signed this value with her secret key.

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 paper than there might appear to be. For public key encryption and decryption are computationally expensive: this means that there is a strong incentive to keep the bandwidth of information transmitted under this form of cipher to a minimum. We might therefore regard the single message described above as the combination of a (perhaps large)

message M over an insecure channel with the smaller one $longhash(M)$ over an authenticated one.

Since in many cases the empirical channels we will be using are human mediated, the chief difference from this view of MACs will be that our empirical channels are much lower bandwidth yet: the amount of security delivered for a given amount of empirical communication becomes the most important measure of a protocol's effectiveness.

2.1 Long authentication string over the empirical channel

The above analysis of the use of MACs shows they are closely analogous to the following protocol, devised by Balfanz [6]. In this scheme, A wants to authenticate its information $INFO_A$ to B . And $longhash_{160}()$ denotes a 160-bit output cryptographic hash function.

Balfanz non-interactive protocol, [6]	
1.	$A \longrightarrow_N B : A, INFO_A$
2.	$A \longrightarrow_{WE} B : longhash_{160}(A, INFO_A)$ B verifies the longhash.

The 160-bit hash sent over the weak empirical channel can be delayed and the information $INFO_A$ is under the control of the intruder, hence s/he might carry out an off-line attack to find a different $INFO'_A$ with the same hash value.⁶ That is something which the standard specification of a hash function deems infeasible as it must take about $2^{160/2} = 2^{80}$ computation steps on average to find such a collision due to the birthday paradox. As a result of a single longhash whose in/output lengths are M and W words, the computation cost required at each node is $WM = 5M$ thanks to the simple model of computation cost given in Section 1.4.

In order to improve the number of authenticated bits, Pasini and Vaudenay [42] make use of a non-deterministic commitment scheme defined in Section 1.3.1 to commit to the authenticated information. The 80-bit hash of the commitment is then sent over the empirical channel.

Pasini-Vaudenay non-interactive protocol, [42]	
1.	$A \longrightarrow_N B : c \parallel d = commit(A, INFO_A)$ B computes $A \parallel INFO_A = open(c, d)$
2.	$A \longrightarrow_{WE} B : hash_{80}(c)$ B verifies the hash.

⁶In the original protocol [6], there is no restriction on the order of sending and receiving Messages 1 and 2.

Here the hash function is required to be weakly collision resistant, or in other words the second preimage resistance property defined in [3], (an intruder cannot find a second value v' such that $hash(v) = hash(v')$ for fixed v) as opposed to the strong collision resistance that is required for the Balfanz protocol (i.e., where both v and v' are allowed to vary).

In [42], Pasini and Vaudenay argue that this provides the same degree of authentication as the Balfanz protocol (namely 2^{80} computation steps) because the nondeterministic commitment scheme avoids the possibility of a birthday attack. The latter occurs in Balfanz if the intruder has the ability to choose the $INFO_A$ that A sends: it can search for pairs $INFO_A$ and $INFO'_A$ that have the same hash, but it cannot do this for commitments because of the nondeterminism that A injects into the latter through her use of a nondeterministic commitment scheme. At the point where A is influenced to use the given $INFO_A$ the intruder cannot know what this nondeterministic component (a hidden random nonce of 80 bits = $W/2$ says⁷) 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, but also halving the bit-length of the commitment scheme owing to the nondeterminism introduced. These together reduce the cost to $MW/2 + W^2/4 = 2.5M + 6.25$.

As far as the authors are aware, this is the best non-interactive scheme in term of the number of authentication bits at the moment. There seems to be a good reason to believe that this cannot be improved much further without loss of security. Suppose we had any scheme in which a message $INFO$ from A to B was supported by a b -bit empirical SAS that was delayable by the intruder. Then an attacker could simply perform a search on different $INFO'$, hunting for another one that would be verified by the same SAS. Evidently, searching through the order of 2^{80} $INFO'$ s needed to have a good chance of finding a collision in the above protocol is likely to be impracticable: that, presumably, is why this number was chosen. The number of bits for any protocol following essentially this pattern will always be at best the same calculation based on the desired improbability of an attack succeeding, the assumed power of an attacker's computer, the time available for a search and the difficulty of performing the various calculations in the protocol.

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

⁷We emphasise that this is the only time when an 80-bit commitment scheme is used. For all other subsequent employments of a commitment scheme, it is always 160-bit.

the recipient knows 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, 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.

2.2 Short authentication strings over strong empirical channels

Taking a different approach, Gehrman, Mitchell, and Nyberg [15, 16, 22] use an authentication or empirical channel, to transmit the b bits, in between

16-20, authenticated string that is the output of a check function $m_k()$ ⁸ together with a b -bit key that has been instrumental in its computation.

MANA I (Gehrmann, Mitchell and Nyberg), [15, 16, 22]	
1.	$A \longrightarrow_N B : A, INFO_A$
2.	$B \longrightarrow_E A : 1\text{-bit committed}$ A picks a b -bit random number k
3.	$A \longrightarrow_E B : k, m_k(A \parallel INFO_A)$ B verifies the digest.
4.	$B \longrightarrow_E A : 1\text{-bit committed}$

To eliminate 1-bit empirical signals in MANA I, Vaudenay proposes to use a strong empirical channel, which achieve stall-free or instant delivery (denoted \longrightarrow_{SE}), to send the key and the check-value.⁹ Thus $2b$ bits are transmitted in all. Coincidentally, 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. And the rest of this analysis applies to both versions.

V-MANA I, [55, 42]*	
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.

By directly binding $INFO_A$ of M words to the 1-word short authentication string (SAS), the protocol is very efficient as each node only has to compute a single digest at a cost of M that is much cheaper than a normal hash function in Balfanz or a commitment scheme in Pasini-Vaudenay.¹⁰ The advantage of direct binding will be discussed in more detail when we look at pairwise/group protocols in Sections 3.3 and 4.1. The protocol demonstrates that the use of the strong empirical channel that provides stall-free transmission will lead to a significant fewer number of authenticated bits transmitted from A to B : this is the first example we have seen of a protocol that, given

⁸As suggested in [15, 16], a check function $m_k()$ can be implemented by either CBC-MAC or universal hash functions based on error correcting code, which is not efficient as discussed in Section 1.3.2 and [37].

⁹We can replace the strong empirical channel with a bounded delay empirical one (\longrightarrow_{BE}^t) provided B checks that he has received Message 1 before Message 2 could have been sent.

¹⁰This measurement only applies to the modified version of V-MANA I where the digest function is used as opposed to CBC-MAC or longhash functions that will make it increase to $MW = 5M$.

the properties we have assumed of our cryptographic primitives, comes close to preventing the intruder performing any useful combinatorial search. This is because the distribution properties of the digest mean that it is impossible for the intruder to look for an $INFO_I$ that will digest to the same value as $INFO_A$ in ignorance of k .

However, the protocol is not perfect and the first weakness we want to describe is that any one can modify $INFO_A$ blindly in the first message transmitted over the insecure normal network, and hope that the digests come out the same in the second message. And this will occur with a probability of 2^{-b} irrespective of the value of the key provided that the b -bit digest meets the specification defined in Section 1.3.2. What this means is that $2b$ empirical bits only guarantee at best a 2^{-b} security level, and consequently the protocol is far from optimal in the human work: we will see a number of protocols in this paper that achieve 2^{-b} probability with b bits.

Whilst the security proofs presented in [15, 22, 42] are largely correct, what these authors have not discovered is that the bit-length b they choose for the key is too short compared to the digest output and the authenticated information: it is impossible to construct a digest function such that the probability of any one-shot attack is no better than 2^{-b} . Since the weakness has a very profound impact on every subsequent use 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 $\{k_1, \dots, k_c\}$. Based on an off-line brute force search at the cost of $2^{bc/2}$ computation steps, related to the birthday paradox, he can expect to find two different $INFO_A$ and $INFO'_A$ such that for all $k \in \{k_1, \dots, k_c\}$,¹¹ we have:

$$digest(k, A \parallel INFO_A) = digest(k, A \parallel INFO'_A)$$

Assuming that the adversary can influence A to send $INFO_A$ in the first message of the protocol, there is then an attack it can attempt.

- The adversary blocks the message $A, INFO_A$ that A sends, checking

¹¹It might be clearer if we define $H_{\{k_1, \dots, k_c\}}(X) = digest(k_1, X) \parallel \dots \parallel digest(k_c, X)$, and if $digest$ is an ideal digest function, then so is the function H w.r.t its $c \cdot b$ 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.

that it is the particular value that was desired.

- Immediately afterwards (to reduce the chance of A sending the empirical message too soon) it impersonates A to send $A, INFO'_A$ to B .

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 1st message, A picks a random key k : with a probability of $\frac{c}{2^r}$, $k \in \{k_1, \dots, k_c\}$, and the attack is successful. On the other hand, with a probability of $\frac{2^r - c}{2^r} \cdot 2^{-b}$, k is not in this set and so the attack is only successful with a probability of (presumably) 2^{-b} .

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

$$\Pr_r(c) = c \cdot 2^{-r} + \frac{2^r - c}{2^r} \cdot 2^{-b}$$

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 , this will allow the likelihood of a successful one-shot attack $-\Pr_r(c)$ – to converge to 2^{-b} . On the one hand, this is not feasible in this protocol because the key must be sent with the digest value over the strong empirical channel that is severely limited in bandwidth. On the other hand, an interesting question arises as we want to know how large the bit-length of the key should be (to make the function secure) in comparison with a fixed amount of information we want to authenticate and the output bit-length of the digest. Since this question is not within the scope of this survey, we would like to point interested readers to one of our papers [38] where we successfully derive a new combinatorial bound, and then compare it against the well known bound discovered by Stinson in [51].¹²

This result suggests we should aim always to have 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 \rightarrow_N rather than empirically.

¹²We term the digest collision probability ϵ . In the case $\epsilon = 2^{-b}$, then our combinatorial bound gives $r \geq \log \frac{K}{\epsilon b} = b + \log \frac{K}{b}$ that is smaller than Stinson's result: $r \geq K - b$. However, we have discovered that Stinson's bound is only accurate within an extremely short range of values of $\epsilon \in [2^{-b}, 2^{-b} \left(\frac{K}{K-b}\right)]$, and outside the range our combinatorial bound gives a tighter result. More details can be found at [38].

2.3 Improved version of V-MANA I

Given two weaknesses discussed in the previous section, we have presented three different improved versions of V-MANA I in [39] that all optimise the use of the expensive strong empirical channel, and these also apply to MANA I. In other words, human comparison/handling of a x -bit short authentication string (SAS) always corresponds to a probability of 2^{-x} 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.

Contrary to V-MANA I, 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 mentioned in Section 1.3.2. Furthermore it *must* have sufficient entropy that it is infeasible for an attacker, given $longhash(k)$, to deduce k itself.

Improved version of V-MANA I <i>New</i>	
1.	$A \longrightarrow_N B : INFO_A, longhash(k)$
2.	$A \xrightarrow{t}_{BE} B : digest(k, INFO_A)$
3.	$A \longrightarrow_N B : k$
	B then verifies the longhash and the digest.

Whereas in the V-MANA I protocol as discussed in [55] the empirical transmission is instantaneous in order to ensure that Message 1 is definitely received before Message 2 is sent, we can actually weaken this assumption with our improved version. In the first message, the party A will send the longhash of the long random key k along side $INFO_A$ over the normal channel. After that s/he uses the bounded delay empirical channel to transmit the digest value of that information with respect to the key k . The message order is more important here than in most protocols: we specify that B will not accept a Message 2 within t of receiving Message 1 and that A will not send Message 3 within t of sending Message 2. This is to ensure that B was committed to Message 1 when Message 2 was sent, and that Message 3 cannot be received by anyone before B has accepted (if he does) the only Message 2 that A will ever send that relates to it. 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 combinatorial attacks.

From the above description, we can deduce this revised version is optimal in the human work as a consequence of:

- Since the key can be as long as we want within reason (e.g. 160 bits),

the digest function will satisfy the specification given in Section 1.3.2.

- It is infeasible to invert the longhash to search for the key, and therefore the best the intruder can do is to modify $INFO_A$ blindly, and having no more than a 2^{-b} probability of a digest collision with respect to the same key k still unknown to him/her at this moment.

Similar to V-MANA I, the SAS is dependent functionally on $INFO_A$. And that is why we refer to this strategy as direct binding which will be formally defined later in Section 3.3. However, due to the extra longhash required in the first message, the computation cost is slightly increased to $W^2 + M = 25 + M$.

As mentioned earlier, the other two improved protocols presented in [39] are quite similar to this one in term of the overall structure, and so are not given here. However, they use different information binding strategies as well as cryptographic primitives that can be efficiently implemented in hard/software available in a variety of lightweight devices to obtain the optimal security level. Readers can find out more information about these schemes in [39].

3 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.¹³

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.

3.1 Multiple empirical short authentication strings

In this section we will describe two pairwise authentication protocols, the first by Hoepman [20, 21] and the second by Wong and Stajano [58, 59],

¹³Perhaps this could be worked around by having a logged recording mechanism for the empirical messages as part of the receiver's system.

where parties have to manually compare or handle two different short authentication strings (SASs) each of $b = 16$ bits, so 32 bits in all. We will also discover an important difference in how these two protocols process *INFOS*.

Hoepman in [21] defines SASs as the outputs of a b -bit (short) hash function, termed a $hash_b()$ function as mentioned in Section 1.3.2.¹⁴ In addition, the Diffie-Hellman tokens $g^{x_{A/B}}$ play the role of both $INFO_{A/B}$ and long fresh random nonces, and consequently must be unpredictable and fresh at each session.

Hoepman pairwise protocol, [21]	
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.	$A \longrightarrow_E B : hash_b(g^{x_A})$
2'.	$B \longrightarrow_E A : hash_b(g^{x_B})$
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'.	$B \longrightarrow_N A : longhash(g^{x_B x_A})$

As pointed out in that paper, it is vital here that both parties must be committed to their view of the final key $g^{x_A x_B}$ (in other words, they need to agree on when to finish inputting the first messages) before any of Messages 2, 2' and 3, 3' are sent out. In contrast, what has not been noticed in the two papers of Hoepman [20, 21] is that after the commitment phase, Messages 2, 2', 3 and 3' can be transmitted in any order without compromising the security. The fact that this protocol offers a good security despite the use of very short hashes can be explained through the idea of *commitment before knowledge* in which all parties are committed to some final value before they actually know what it is. This has influenced our improved V-MANA I protocol, as it does nearly every subsequent pairwise/group protocol presented in this paper.¹⁵ And we will see an example of what goes wrong without

¹⁴Hoepman [20, 21] did not specify how to construct a short hash function. Of course, one can truncate the first b bits of a longhash function but it will be expensive. One possibility is to use the digest function with a fixed key.

¹⁵In [20], Hoepman introduced a modified (pairwise) version of the above scheme in which each party can receive multiple longhashes/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 sent over the empirical channel in the second message. In this

this principle at the start of Section 4.1 that discusses group protocols.

We will assume that M is also the word length of the Diffie-Hellman tokens¹⁶ (g^{x_A} and g^{x_B}). Since Messages 4 only provide shared secret validation (using *longhash()* function in this case), they do not add any extra security to the protocol, and can be neglected in our cost analysis. As a result, each node has to compute 2 longhashes and 2 shorthashes, the computation cost therefore is of order $2(WM + M) = 12M$, where W and 1 are the output word-lengths of long and respectively short hashes.

Taking a very different approach, Ford-Long Wong and Frank Stajano [58] propose another scheme that neither requires a digest nor an efficient short-hash function which might be unavailable in lightweight devices. 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 $(160 - b)$ bits respectively, and are generated at each node.

Wong-Stajano pairwise protocol, [58]	
1.	$A \longrightarrow_N B : g^{x_A}$
1'.	$B \longrightarrow_N A : g^{x_B}$
2.	$A \longrightarrow_N B : \text{longhash}(A, g^{x_A}, g^{x_B}, R_A, K_A)$
2'.	$B \longrightarrow_N A : \text{longhash}(B, g^{x_B}, g^{x_A}, R_B, K_B)$
	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.

The security of this protocol comes from the intruder's inability to invert the longhashes or predict the non-determinism introduced by the pair of nonces (R_X, K_X) at the point when these are committed to. As in Hoepman, both parties must receive each others' longhashes before they reveal either the long or short nonce in the third and fourth messages. Once the

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 original protocol applies to the *non-anonymous* case.

¹⁶As the Diffie-Hellman tokens are the only information parties want to have authenticated here, we can treat them as $INFO_{A/B}$ whose lengths are M words.

commitment phase (sending out the longhashes) has passed, Messages 3, 3', 4 and 4' can also be transmitted in any order.

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

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} , instead of (g^{x_A}, g^{x_B}) , to the random nonces (R_A, K_A) . This will lead to 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 [59]. However, what has not been noticed in [59] is that Diffie-Hellman tokens can be made unpredictable, fresh in each session, and computed off-line. Therefore, using the same technique in Hoepman, we can replace (and remove) the long random nonces $K_{A/B}$ with $g^{x_{A/B}}$ that will further improve the computation cost. A detailed description of the modified version will be given in Section 3.2.1

In terms of human work, both Hoepman and Wong-Stajano are not optimal in the amount of work required by the humans implementing the empirical channel because they need the comparison of more than one string, whereas the same security level (i.e. improbability 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 also introduces another major disadvantage: if we want to generalise these protocols into multi-party versions then, even though this is possible, the number of different SASs each party has to compare or handle manually is always equal to the total number of nodes that is infeasible for human in practice.

We end this section with a crucial observation: Hoepman chooses to bind the Diffie-Hellman tokens directly to the SASs. This is not the case in Wong-Stajano. By this we mean that the *INFOs* we are trying to authenticate are, in the Hoepman, used directly in the evaluation of the empirically compared strings, while those compared in Wong-Stajano are not. These two different strategies can be termed *direct* and *indirect* bindings, and we will explore and compare them in detail when we study the classes of protocols that minimise human effort in the sections to come.

3.2 Indirect binding

In indirect binding protocols, the SASs manually communicated by parties are functionally independent of the information they want to authenticate. This is the idea we have briefly seen in Wong-Stajano, and it appears in many other schemes proposed in the literature such as in [55, 41, 8, 58, 59, 5, 27, 28]. We will analyse these here. What distinguishes all of these from Wong-Stajano is the single SAS 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 that binds random nonces and the authentic information together in a secure way. There is a tendency to use the commitment scheme in these protocols to obtain that binding.

3.2.1 Indirect pairwise

We will discuss protocols covering two different circumstances in bootstrapping security. These were devised by Vaudenay, Čagalj and others in [55, 8], which concentrate on the case of two parties in a *peer-to-peer* network, and establish one/two-way authentication via one/two-way empirical channels. We will extend their schemes into group versions in Section 4.1.

These protocols employ the *commitment before knowledge* principle, also used in Hoepman and Wong-Stajano protocols, to precommit two or multiple parties to some random short secrets/nonces by the output of the non-deterministic commitment scheme to each other at the start of the protocol. Thanks to the use of short random nonces or entropies and the indirect binding strategy, parties only need to manually compare a unique b -bit string which is the XOR of short entropies devised by all parties.

The following is the description of a pairwise scheme invented by Vaudenay in [55] 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, [55]

1. $A \longrightarrow_N B : INFO_A, c$
Where $c \parallel d = \text{commit}(INFO_A, R_A)$,
 R_A is a short random nonce of A .
2. $B \longrightarrow_N A : R_B$
3. $A \longrightarrow_N B : d$
 B computes $R_A = \text{open}(INFO_A, c, d)$
4. $A \longrightarrow_{WE} B : R_A \oplus R_B$
 B verifies the correctness of $R_A \oplus R_B$

Note that the guarantee of authenticity of $INFO_A$ that this protocol delivers is as a consequence of:

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

There is a single commitment used (committed by A , and decommitted/opened by B), hence the computing cost at each node is of order $MW = 5M$. Here W is the word-length of the commitment output that is either a commitment c or a decommitment d .¹⁷

Although Vaudenay's scheme is much more efficient in empirical communication than Hoepman and Wong-Stajano, it only provides one-way authentication. In practice, we often want to achieve more than this, and that is why we now consider a second protocol which performs message authentication in both directions at the same time. Suppose B also 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 the party A in the first message. Fortunately, this idea proposed in Appendix A of [55] makes the protocol structure completely symmetrical. It is also essentially the same as the protocol termed DH-SC invented by M. Čagalj, Čapkun, and J. Hubaux in [8].

¹⁷The cost of XORing two short random nonces R_A and R_B is very small compared to implementing the commitment scheme, and therefore is neglected here.

Čagalj-Čapkun-Hubaux two-way authentication protocol, [8]	
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.	$A \longleftrightarrow_E B : R_A \oplus R_B$

We note this scheme can be regarded as an upgraded version of Wong-Stajano (Section 3.1) 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 does not have to acquire $INFO_B$ at the time of computing the commitment. Secondly, the order of releasing the SAS and the decommitments has been reversed compared to Wong-Stajano.¹⁸ The effect of this is that, as the short entropies R_A and R_B can be implicitly derived from the decommitments, parties only need to compare a single SAS that is the XOR of these entropies over the empirical channel.

It is interesting to note that the same technique can be used to improve the human and processing cost of Wong-Stajano.¹⁹

As regards computation cost, two commitments need to be computed and verified at each node, which is twice as costly as Vaudenay's one-way authentication scheme, namely $2WM = 10M$. On the other hand, if we quantify the cost in term of the amount of information authenticated then Vaudenay and Čagalj-Čapkun-Hubaux will be equivalent to each other. The

¹⁸The SAS and the decommitments here correspond to the two different short nonces and the long nonces in Wong-Stajano, respectively.

¹⁹The 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 in Wong-Stajano can be demonstrated by the following revised scheme that is very similar to Čagalj-Čapkun-Hubaux two-way authentication protocol.

Improved version of Wong-Stajano <i>New</i>	
1.	$A \longrightarrow_N B : longhash(g^{x_A}, R_A)$
1'.	$B \longrightarrow_N A : longhash(g^{x_B}, R_B)$
2.	$A \longrightarrow_N B : R_A \parallel g^{x_A}$
2'.	$B \longrightarrow_N A : R_B \parallel g^{x_B}$
3.	$A \longleftrightarrow_E B : R_A \oplus R_B$

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 ($20M + 50$).

result also illustrates the gain in efficiency of these in comparison with Hoepman and Wong-Stajano in Section 3.1.

Another advantage of Čagalj-Čapkun-Hubaux is that the symmetrical structure and a single SAS subsequently led us to realise the possibility of generalising it into a group version which is described in Section 4.1.

We will discover later, however, that these protocols are not optimal in computational effort thanks to their use of indirect binding.

3.2.2 Hybrid protocol

Prior to discussing direct binding protocols, we will describe an important scheme which bridges the gap between the two strategies, both in terms of protocol structure and computational cost. Pasini and Vaudenay [41] propose a two-way authentication protocol based on the original one-way scheme of Vaudenay seen in Section 3.2.1, but also make use of a truncated hash function that we have improved to a digest. They devise the following protocol with a symmetric empirical channel.

Pasini-Vaudenay two-way authentication protocol, [41]*	
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.	$A \longleftrightarrow_E B : R_B \oplus digest(k_A, INFO_B)$

Though the SAS depends functionally on $INFO_B$, it is probabilistically independent from $INFO_A$ thanks both to $R_B \oplus$ and the properties of k_A and digest function. This observation is interesting because it makes the scheme stand as a hybrid of direct- and indirect-binding protocols. This is also reflected by the differences between 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.

We note that there is no need to use the commitment to bind $INFO_B$ to R_B any more, and each node now only needs to compute/verify one commitment and one digest. The processing cost at each node drops to

$WM + M = 6M$ thanks to the efficiency of digest.²⁰ This is significantly cheaper than Čagalj-Čapkun-Hubaux two-way authentication protocol ($10M$).

3.3 Direct binding pairwise protocols

The direct binding approach requires the SAS to be dependent on the information parties 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 others that are optimal in this respect. In this section we discuss the ones introduced as pairwise protocols, while in Section 4 we will discuss those introduced as group protocols. 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: Hoepman (direct) versus Wong-Stajano, and Pasini-Vaudenay (direct) versus Čagalj-Čapkun-Hubaux.

Our first task is to formalise the direct binding approach as the following principle **P1**, originally proposed in [36, 37].

- P1** Make all the parties, who are intended to be part of a protocol run, empirically agree a short-output hash or *digest* of a complete description of the run.

In all the protocols we introduce in this paper, the “complete description of the run” is identified with *INFOS*, the collection of all the information that any member of the group wishes to have authenticated to it: the concatenation of pairs of the form $(A, INFO_A)$. Once the agreement required in **P1** has occurred then, 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, three different protocols that provide mutual authentication are presented in an increasing order of computation efficiency and simplicity. In addition to the shared use of the direct binding strategy, they are all asymmetrical in structure which is very similar to the original one-way authentication protocol of Vaudenay [55] presented in Section 3.2.1.

²⁰There should have been no improvement ($WM + WM = 10M$) if we had not switched to the use of digest.

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 attack carried out by the intruder at the beginning.²¹ This, coupled with their subtle use/integration in the digest function to prevent the off-line attacks illustrated in Section 2.2, once again demonstrates the use of the principle – *commitment before knowledge* – that provides a common theme to this family of protocols.

Due to the different role that the sub-keys play in the digest computation in comparison with *INFOS*, we will also analyse how sub-keys are combined into a single digest key.

The security analysis of the following direct binding schemes is very similar to indirect binding ones provided the digest function is ideal as specified in Section 1.3.2. In every case the protocol is secure because nodes (and hence the intruder) do not know the final value of the key k until after they are committed to the final value of the digest, truncated hash, or universal hash output.

The following protocol is taken from Bluetooth whitepaper [5], where k_A and k_B are long fresh sub-keys generated by A and B .

Bluetooth 2, [5]	
1.	$A \longrightarrow_N B : INFO_A$
1'.	$B \longrightarrow_N A : INFO_B$
2.	$B \longrightarrow_N A : longhash(INFO_S, k_B)$
3.	$A \longrightarrow_N B : k_A$
3'.	$B \longrightarrow_N A : k_B$
	k_Y is a long fresh key of Y
4.	$A \longleftrightarrow_E B : trunc_b(longhash(k_A, k_B, INFO_S))$

In this protocol, the two different sub-keys are concatenated with *INFOS*: $f(k_A \oplus k_B) = k_A \parallel k_B = k^*$. Since the operator is not commutative, the 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*.

Due to the inefficiency in using a truncated hash function, the computation cost of the above “Bluetooth 2” will be $W(2M+W)+2WM = 20M+25$ compared to only $W(2M+W) + 2M = 12M + 25$ should we employ a digest.²² Unfortunately, the latter will still be more expensive than the two

²¹It is possible to regard these long fresh sub-keys as the extended versions of short random nonces used in the commitment schemes in indirect binding protocols.

²²If we want to use a digest then there is a serious problem in key distribution caused by the operator concatenation: each individual sub-key k_X does not randomise the output or every bit of the final key. This problem will get worse in a group formation in which

following schemes because it is redundant to bind *INFOS* and sub-keys together by using both longhash function in Message 2 and in the SAS, since either of them would be sufficient for the obtained security.

By removing this unnecessary binding in Message 2 of Bluetooth 2, we can increase its computational efficiency as well as simplicity (eliminating Messages 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 in [27, 28] by Laur and Nyberg:

Laur-Nyberg pairwise protocol, [27, 28]	
1.	$A \xrightarrow{N} B : INFO_A, c$ Where $c \parallel d = commit(k_A)$
2.	$B \xrightarrow{N} A : INFO_B, k_B$
3.	$A \xrightarrow{N} B : d$ <i>B</i> computes $k_A = open(c, d)$
4.	$A \xleftrightarrow{E} B : uhash(g(k_A, k_B), INFOS)$

Here *uhash* is a universal hash function [51] of the appropriate length whose specification is closely related to a digest as discussed in 1.3.2. The impact of removing redundancy can be seen in the computation cost of this protocol which declines to $W^2 + 2WM = 25 + 10M$ thanks to direct binding.

Unlike Bluetooth 2, Laur and Nyberg use a different function $g(k_A, k_B) = (k_A^1 \cdot k_B) \oplus k_A^2$ based on (polynomial) multiplication over the 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 . We note that 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 an irreducible polynomial of order $r/2$ used to manipulate the finite field prior to each session.

It would be equally satisfactory to use the combination of \oplus between k_A and k_B , and a digest function, in place of $uhash(g(k_A, k_B), INFOS)$, resulting in an improvement of computational cost ($W^2 + 2M = 25 + 2M$).

After this transformation and a replacement of a commit scheme with a longhash, the protocol becomes similar to the following, discovered independently by the authors in the summer of 2006 when combining the ideas of SHCBK (see Section 4.1) and Vaudenay's protocols (see Section 3.2.1):

there are multiple sub-keys, and it might end up that a big subset of the group \mathbf{G} can strongly influence the final digest value upon some partial knowledge of the digest key k^* . This problem leads us to propose exclusive-or to combine sub-keys in both pairwise and group protocols discussed later in this section and 4.1.

Pairwise authentication scheme in Vaudenay's style <i>New</i>	
1.	$A \longrightarrow_N B : INFO_A, longhash(k_A)$
2.	$B \longrightarrow_N A : INFO_B, k_B$
3.	$A \longrightarrow_N B : k_A$
4.	$A \longleftrightarrow_E B : digest(k_A \oplus k_B, INFOS)$

The same ideas can be used to devise a much more efficient version of the Hoepman protocol:

Improved Hoepman <i>New</i>	
1.	$A \longrightarrow_N B : longhash(g^{x_A})$
2.	$B \longrightarrow_N A : g^{x_B}$
3.	$A \longrightarrow_N B : g^{x_A}$
4.	$A \longleftrightarrow_E B : hash_b(g^{x_A} \oplus g^{x_B})$

The main difference between this and the previous schemes is that there is no $INFO_{A/B}$ because the Diffie-Hellman tokens revealed in the second and third messages 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.²³ Also because of this, the digest function which requires 2 inputs can be replaced by a single input shorthash; though the combination of this and the exponentiation of Diffie-Hellman needs to satisfy a specification similar to that of the ideal digest and the randomising effect of XOR in combining k_A 's into k^* .

In comparison with Hoepman, this only requires a single SAS, halving the human work. And similar to the three previous protocols, one longhash and one short hash required result in a computation cost of $WM + M = 6M$ that is exactly a half of Hoepman ($12M$).

It is worth thinking for a moment about how by which 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 tells the first of the agreement. We might therefore structure the above protocol

²³It 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' (One-way empirical channels)	
1.	$A \longrightarrow_N B : \text{longhash}(g^{x_A})$
2.	$B \longrightarrow_N A : g^{x_B}$
3.	$A \longrightarrow_N B : g^{x_A}$
4.	$A \longrightarrow_E B : \text{hash}_b(g^{x_A} \oplus g^{x_B})$
5.	$B \longrightarrow_E A : 1\text{-bit committed}$

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

We note that once A has received Message 2 from B , he can send Messages 3 and 4 in any order.

Wong and Stajano [59] give a protocol that uses this separated structure explicitly; it is however more expensive at $2WM = 10M$ as well as requiring another 1-bit empirical signal in Message 3:

Wong-Stajano (One-way empirical channel), [59]	
1.	$A \longrightarrow_N B : g^{x_A}$
2.	$B \longrightarrow_N A : B, g^{x_B}, \text{MAC}_{K_B}(B, g^{x_A}, g^{x_B}, R_B)$ R_B and K_B are short and long random nonces of B
3.	$A \longrightarrow_E B : 1\text{-bit committed}$
4.	$B \longrightarrow_E A : R_B$
5.	$B \longrightarrow_N A : K_B$
6.	$A \longrightarrow_E B : 1\text{-bit committed}$

We note that the order of Messages 4 and 5 can be interchanged. The resulting final pair of Messages (4 and 6) then just result 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 *commitment before knowledge* approach to work, we need to have token that randomises the final digest committed before one node knows it. This has to be done with strong cryptography, and the hash used in, for example, the second Laur-Nyberg protocol above appears to be as efficient as possible at doing this.

4 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. The reason may be because this type of application is very popular in practice ranging from the financial industry, mobile phones, to military applications. We believe there is similar potential for bootstrapping security in larger groups. For example, a group of people come might together and agree that they 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). Unfortunately 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.²⁴

Work in this area seems so far to have been restricted to the authors' group (including Creese, Goldsmith and Zakiuddin) and more recently Valkonen. This has resulted in a number of group protocols presented in [11, 12, 13, 14, 36, 37, 54]. In order to tackle the difficulties, the authors of [11, 12, 13, 14, 36, 37, 45] first identify the main challenges in pervasive computing that will be briefly discussed below. They then show that by using the combination of the principle **P1** (the direct binding approach), and the *commitment before knowledge* idea, it is possible to construct efficient and secure group protocols.

There is a slightly grey area for protocols that build 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 that result between 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 protocol which creates a whole-group shared secret is at least partially compromised by the presence of a corrupt participant. However protocols which merely authenticate public-key-like information to nodes are not automatically compromised: they could be said to be establishing a *local PKI*. And, as we will see, this will be successfully resolved by using the idea of *commitment before knowledge*.

The issue of scalability plays a crucial role in constructing pervasive group protocols because of the limited computation power of lightweight

²⁴Any pair of them of course can run one of the above pairwise schemes to agree a shared secret. But this will quickly become infeasible: even with a small group of 10 people there will be 45 pairs.

devices and the fact that the amount of computing required will inevitably grow as the size of the group does. Our priority is still to minimise the human work relative to the security obtained. The best we can hope for is the same as in the binary case: that 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, for pairwise protocols, indirect binding is significantly less efficient than direct. In this section we will see that the same is true for group protocols. We then demonstrate that it is possible to further improve the efficiency with a trade-off between the human and computation costs or by applying the technique used in the original one-way protocol of Vaudenay in [55].

4.1 Some existing group protocols

We start with a group protocol, originally proposed in [13] by Creese, Goldsmith, Roscoe and Zakiuddin. In the following description, Pk_A stands for an uncertificated public key of A . The superscripted “Message 2^d ” represents the decrypted content of Message 2. Messages 4 in this protocol do not add any extra security to the scheme; their presence only aims to provide a re-confirmation on the shared secret information.²⁵

Group protocol of Creese <i>et al</i> [13]	
1.	$\forall A \longrightarrow_N \forall A' : A, Pk_A, N_A$
2.	$\forall A \longrightarrow_N \forall A' : \{\text{all Messages 1, } N'_A\}_{Pk_{A'}}$
3a.	A displays $: hash_b(\{\text{all Messages } 2^d\})$, number of processes
3b.	$\forall A \longrightarrow_E \forall A' : \text{Users compare hashes and check numbers}$
4.	$\forall A \longrightarrow_N \forall A' : longhash(\{\text{all Messages } 2^d\})$

Their scheme is shown in [45, 36, 37] by Roscoe to be vulnerable to a combinatorial attack, related to the birthday paradox. The flaw arises from the short output of the shorthash function $hash_b()$ used in Messages 3a, and the intruder’s ability to manipulate the content of Messages 1 and 2. The details of the attack can also be found in [45, 36, 37]. In spite of the attack, the protocol invented in 2003 actually introduced, implicitly, the principle **P1** stated above, that minimises the empirical work.

In summer 2005 [45], Roscoe corrected this flaw by introducing a leader I , who is responsible for generating a fresh key k_I of order 160 bits that became the digest function used in this scheme. The following is a slightly

²⁵Messages 4 in this protocol are similar in purpose to Messages 4 in Hoepman (Section 3.1).

simplified version of this protocol, introduced in [36, 37]. Here, S represents a typical slave node, and A a typical node (either I or S). $init(I, A)$ is *true* if $I = A$ and *false* otherwise.

Hash Commitment Before Knowledge HCBK protocol, [45, 36, 37]			
0.	I	$\longrightarrow_N \forall S$	$: I$
1.	$\forall A$	$\longrightarrow_N \forall A'$	$: (A, INFO_A)$
2a.	I	$\longrightarrow_N \forall S$	$: longhash(k_I)$
2b.	$\forall S$	$\longrightarrow_E I$	$: committed$
3.	I	$\longrightarrow_N \forall S$	$: k_I$
4a.	$\forall A$	displays	$: digest(k_I, INFOS), init(I, A)$
4b.	$\forall A$	$\longrightarrow_E \forall A'$	$: \text{Users compare and check presence of } I$

In this scheme, the parties have to agree on the b -bit digest of $INFOS$ and the leader's key k_I over the empirical channel. In addition, Message 2b has all the slaves communicate to I that they have received Message 2a, and are *committed* to their final digest value (though none of them know it yet). Thus the 1-bit commitment signal must be sent over the unforgeable empirical channel that cannot be blocked.²⁶ We will see shortly this represents one side of an interesting trade-off. In term of computation cost, each node has to compute or verify a single cryptographic hash and the b -bit digest value that are equivalent to $W^2 + NM = 25 + NM$.

The protocol is termed HCBK standing for *Hash Commitment Before Knowledge*, and its security relies on the trustworthiness of the initiator I who generates the single digest key, and consequently has control over the final digest value.²⁷ 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 [59] from Section 3.3. 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 variant on pairwise HCBK.

In many circumstances, it is possible for such a leader to emerge (for example as the system whose owner initiates the protocol). However this

²⁶This requirement seems to be stronger than the properties of the empirical channel (\longrightarrow_E), defined in Section-1.2. However, in [45, 36, 37], Roscoe introduced timing limits to fix the problem, quoted as follows: "an upper bound on the time between I sending Message 3 and agreeing Message 4, and a lower bound between I receiving a Message 2b from S and S sending another Message 2b". Since the scheme is interactive, it is reasonable to ask the participants to follow such a timing condition.

²⁷The readers can find the full security analysis of HCBK in [36].

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, 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 [36, 37] we identified the following second principle, derived from the leader’s role in HCBK, that essentially makes parties committed to the final digest value before any of them knows what the value actually is.

P2 A node A is safe from effective manipulations of its final hash or digest d to equal others in a hash anomaly provided that there is a point at which the following things are both true.

- (a) A is committed to its final value $d = \text{digest}(k^*, \text{INFOS})$, though it may not yet know it.
- (b) There is a value k_A which A knows, randomising the calculation of k^* , which (i) no other party can know and (ii) no other party can have used in the protocol in a way that has influenced A ’s final digest d .

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

In the resulting protocol, every node will play a role similar to the leader in HCBK and thus follow **P2**: each node A now needs some fresh and unpredictable sub-key k_A (of 160 bits say) to contribute to the final digest value.

Symmetrised HCBK protocol (SHCBK), [36, 37]	
1.	$\forall A \longrightarrow_N \forall A' : A, \text{INFO}_A, \text{longhash}(A, k_A)$
2.	$\forall A \longrightarrow_N \forall A' : k_A$
3.	$\forall A \longrightarrow_E \forall A' : \text{digest}(k^*, \text{INFOS})$
where k^* is the XOR of all the k_A ’s for $A \in \mathbf{G}$	

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.²⁸ Note that this protocol eliminates the one-bit commitment signals from the slaves to I .

²⁸We note the same protection is required in Čagalj-Čapkun-Hubaux whenever there are two or more longhashes or commitments. This reflexive attack does not work against HCBK as there is only one cryptographic hash, generated by the initiator, $\text{longhash}(k_I)$.

This protocol is termed Symmetrised HCBK due to the similarity with HCBK and its symmetrical structure. Since everyone takes responsibility separately for influencing the final digest key k^* and the final digest, 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. As a result, this authenticates trustworthy parties to each other irrespective of what others (dishonest nodes) may have done. In other words this protocol is tolerant of corrupt parties: one of the main challenges in designing group protocols.

The robust security achieved here comes at the expense of increased computation compared to HCBK: 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 1 of HCBK. The computation cost of SHCBK 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.

The use of XOR to combine different sub-keys in SHCBK protocol was shown to be secure in [37]. 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 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). $k_{A/B}^*$ 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 specification given in Section 1.3.2. When they are dependent (which they will be – indeed equal – if all nodes are trustworthy and there is no intruder), as discussed in [37], 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 . But this again does not give him any advantage thanks to the second part (in particular “ $\oplus \theta$ ”) of the digest specification.

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

We have claimed that the direct binding approach used by HCBK and

SHCBK remains more efficient in group protocols as it was in pairwise ones. The argument remains the same: it is more efficient to have the short string created from the presumed large *INFOS* in a way where the intruder is prevented structurally from mounting a combinatorial attack rather than it is to have *INFOS* bound to a random short string by full-power cryptography that has to be able to resist such attacks. In order to illustrate this advantage, we will generalise the indirect binding mutual authentication schemes in Section 3.2.1 into a group protocol:

Indirect-binding group protocol <i>New</i>	
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.	$\forall A \longrightarrow_E \forall A' : \bigoplus_{A \in \mathbf{G}} R_A$

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

4.2 Modified versions of HCBK and SHCBK

The difference in efficiency between SHCBK and HCBK raises the question of whether it is possible to reduce the amount of computation processing without compromising the security obtained in SHCBK (i.e. the protocol is still tolerant of corrupt parties). A small improvement turns out to be possible if we make use of the technique originally applied in the one-way authentication protocol of Vaudenay as well as other mutual authentication schemes presented in Sections 3.2.1 and 3.3. On the one hand the technique can be used successfully to slightly reduce the number of commitments or longhashes at each node, on the other hand it makes the schemes asymmetrical in structure. And this can be explained explicitly as follows.

Let us assume there are $(N - 1)$ leaders C out of a total of N parties, where each leader has to generate a fresh sub-key, compute and send its longhash over the normal network. The single one left, termed S , is the unique slave that only has to transmit its fresh sub-key to other nodes in the clear after receiving longhashes from all other leaders.

De-symmetrised SHCBK protocol, [54]*	
0.	$S \longrightarrow_N \forall C : S$
1.	$\forall C \longrightarrow_N \forall A : INFO_C, longhash(C, k_C)$
2.	$S \longrightarrow_N \forall C : INFO_S, k_S$
3.	$\forall C \longrightarrow_N \forall A : k_C$
4.	$\forall A \longrightarrow_E \forall A' : digest(k^*, INFOS)$
where k^* is the XOR of all the k_A 's for $A \in \mathbf{G}$	

We discovered this shortly after SHCBK; it was independently invented by Valkonen, Asokan and Nyberg [54], who were not aware of SHCBK. The authors of that paper neither addressed the issue of tolerance of untrustworthy parties nor the use of a digest function.

As can be seen from the protocol, while there is no commitment attached to the sub-key k_S of the slave (that is sent out in the clear after all the longhashes and before the release of all the leaders' sub-keys), the fact that it is the only one treated in this way guarantees it will not be manipulated by the intruder. This is as resistant to corrupt participants as SHCBK, 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, 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 SHCBK by 25 units per node, though we suspect that the asymmetry it introduces 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 to be impossible to employ the same technique again 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 that reveals its k_A before having, for each other B , either k_B or a commitment like $longhash(k_B)$, is revealing its last piece of information too soon, namely before it is committed to the final digest. An example of this attack (applied to the case with one leader and two slaves) can be found in Appendix B.

We can, however, reduce computational cost if we are prepared to weaken our corruption tolerance requirement towards that of HCBK. With the addition of 1-bit empirical commitment signals like those in HCBK, and allowing

the number of leaders, l , to vary between 1 and N , we can create a *hybrid* protocol where each node requires l longhashes: rather than having a single leader generating the digest key by itself, we will now have l leaders each generating a sub-key in such a way that all of them would have to be corrupt for the protocol to fail. 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 2 longhashes.

Below, S represents a slave, C is a leader, and A is either a slave or a leader. L is the set of l leaders whose identities are informed to every one in Message 0 broadcasted by a single node I who knows this information.

Hybrid HCBK <i>New</i>		
0.	I	$\rightarrow_N \forall A : L$
1.	$\forall A$	$\rightarrow_N \forall A' : A, INFO_A$
2a.	$\forall C$	$\rightarrow_N \forall A : longhash(C, k_C)$
2b.	$\forall S$	$\rightarrow_E \forall C : committed$
3.	$\forall C$	$\rightarrow_N \forall A : k_C$
4.	$\forall A$	$\rightarrow_E \forall A' : digest(k^*, INFOS), leader(L, A)$
Where k^* is the XOR of all the k_C 's for $C \in L$		

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 HCBK and SHCBK.

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.

Let us assume that the set L 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 $leader(L, A)$ indicates whether A is a leader or not. Post hoc this establishes agreement on who the leaders are in 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 who is trustworthy. This is, perhaps surprisingly, sufficient. If A is any trustworthy node who was correct in this step, and the protocol has apparently completed successfully, then with probability at least $1 - 2^{-b}$ it has the correct $INFO_B$ for any other trustworthy node.

To see this note that A , B and C (the trustworthy leader 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 k_C . In this case the probability of A 's and C 's digests agreeing is no more than 2^{-b} by **P2** applied to C .
- A 's final digest was influenced by k_C , as was B 's. In this case we can use the same argument that applies to HCBK: even if B does not recognise C as a leader, C will still have waited for B 's commitment signal.
- A 's final digest was influenced by k_C but B 's was not. Again, C will have waited for B 's commitment, so in this case k_A^* and k_B^* are independent by the properties of digest functions.

Of course in order for the digests to agree it is necessary that the *nodes* as opposed to their human users to 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 increasing the trustworthiness requirement from each leader.

In the modified version of SHCBK, Vaudenay's trick is used to decrease the number of longhashes from N in SHCBK to $N - 1$ but without loss of security of the scheme.

5 Conclusions and further work

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

5.1 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.

The non-interactive protocols are shown in Table 1. As observed in Section 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 four protocols listed: here 160 and 80 are examples of the numbers of bits required to make a hash function strongly and weakly collision resistant, it assumed that 2^{-16} likelihood of attacker success is sufficiently small in the last two cases, but for reasons discussed earlier MANA I does 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 (say within a factor of 2), so that lessons that this table teaches us about relative cost will remain true.

The same will, naturally, be true of the other tables, and the reader is again advised to regard constants like 160, 16, and $25 = (160/32)^2 = W^2$ 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. In those protocols that require the extra confirmation message over the empirical channel (HCBK, HHCBK etc) we write "16+" as the amount of empirical effort.

Protocol	Binding	Human work(bit)	Computation cost
Balfanz (weak empirical)	Direct	160	$WM = 5M$
Pasini-Vaudenay (weak empirical)	Indirect	80	$MW/2 + W^2/4 = 2.5M + 6.25$
MANA I (CBC-MAC) (empirical channel)	Direct	32+	$WM = 5M$
V-MANA I (<i>digest</i>) (strong empirical)	Direct	32	$M = M$
Improved V-MANA I (bounded empirical)	Direct	16	$M + W^2 = M + 25$

Table 1: Non-interactive one-way authentication protocols

In these tables we have used the simple cost model of hash and digest functions described in Section 1.4: the cost is proportional to the product of the length of the information being digested and the width of the output. While the tables might suggest that the cost is *equal* to this product, what we are actually doing is ignoring the multiplicative constant because *all* the computational costs of these protocols that we have considered come from the same model.

It is also necessary to point out that since most direct binding protocols invented by other authors to date do not use a digest 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 ([51] that require a longhash to compress large messages into a fixed number of bits initially) will be denoted (*longhash*).

Thanks to the principle **P1** and the use of the digest function, in general direct binding protocols are much more efficient than indirect binding ones as can be seen from all three tables: about up to 5 times more efficient should M gets large.²⁹ The larger M (the length of *INFOS*) is, the more accurate this effect.³⁰ This is likely to be the case whenever

²⁹In Section 1.4 the digest output length is rounded up to 1 word (32 bits). However if we have a 16 bit digest and a 8/16 bit processor (which will often be the case in lightweight devices) then the advantage of digest over hashing grows (in other words direct over indirect bindings), potentially, to 10 times = $\frac{5W}{0.5W}$ (from 5).

³⁰This 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.

- (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 hashes.
- (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 confirmation message.

5.2 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_{AS}*, 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_{AS}*, 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.

Protocol	Binding	Human work(bit)	Computation cost
Hoepman (multiple SASs)	Direct	$16 \times 2 = 32$	$2(WM + M) = 12M$
Improved Hoepman	Direct	16	$M + WM = 6M$
Improved Hoepman' (one-way empirical)	Direct	16+	$M + WM = 6M$
Wong-Stajano (one-way empirical)	Direct	16+	$2WM = 10M$
Wong-Stajano (multiple SASs)	Indirect	$16 \times 2 = 32$	$2W(2M + W) = 20M + 50$
Improved Wong-Stajano	Indirect	16	$2W(1 + M) = 10M + 10$
Vaudenay (weak empirical) (one-way authentication)	Indirect	16	$WM = 5M$
Čagalj-Čapkun-Hubaux	Indirect	16	$2WM = 10M$
Pasini-Vaudenay (<i>longhash</i>)	Both	16	$WM + WM = 10M$
Pasini-Vaudenay (<i>digest</i>)	Both	16	$WM + M = 6M$
Bluetooth 2 (<i>longhash</i>)	Direct	16	$W(2M + W) + 2WM = 20M + 25$
Bluetooth 2 (<i>digest</i>)	Direct	16	$W(2M + W) + 2M = 12M + 25$
Laur-Nyberg (<i>longhash</i>)	Direct	16	$W^2 + 2WM = 10M + 25$
Laur-Nyberg (<i>digest</i>)	Direct	16	$W^2 + 2M = 2M + 25$
Vaudenay-style (<i>digest</i>)	Direct	16	$W^2 + 2M = 2M + 25$

Table 2: Interactive pairwise two-way authentication protocols (unless indicated they all use two-way empirical channels: \rightarrow_E)

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

Table 3: Group authentication protocols (they all use empirical channels: \rightarrow_E)

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, of 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.

5.3 Conclusions

In 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 notations, 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 interaction plays a central and very important role in the security of these 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 view of where this research area is heading to, and what can be done to make these protocols usable in practice.

5.4 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 in [54].

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 [4] 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) [35], auction protocols (anonymous physical broadcast channel) [48] and e-cash [10] 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) are also very important for the future of these protocols. As a result, this area has received much attention from many different research groups [31, 33, 34, 46, 53] recently.

References

- [1] See: http://en.wikipedia.org/wiki/Avalanche_effect
- [2] See: www.bluetooth.com/developer/specification/specification.asp
- [3] See: http://en.wikipedia.org/wiki/Cryptographic_hash_function
- [4] See: <http://www.prismmodelchecker.org/>
- [5] *Simple Pairing White Paper*. See:
[www.bluetooth.com/NR/rdonlyres/
0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/
SimplePairing_WP_V10r00.pdf](http://www.bluetooth.com/NR/rdonlyres/0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/SimplePairing_WP_V10r00.pdf)

- [6] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. *Talking to strangers: Authentication in Ad Hoc Wireless Networks*. In Symposium on Network and Distributed Systems Security, San Diego, California, USA, 2002.
- [7] J. Bierbrauer, T. Johansson, G.A. Kabatianskii, and B.J.M. Smeets. *On Families of Hash Functions via Geometric Codes and Concatenation*. Advances in Cryptology, CRYPTO'93, LNCS vol. 773.
- [8] M. Čagalj, S. Čapkun, and J. Hubaux. *Key agreement in peer-to-peer wireless networks*. Proceedings of the IEEE, Special Issue on Security and Cryptography, vol. 94, no. 2, February 2006.
- [9] J.L. Carter and M.N. Wegman. *Universal Classes of Hash Functions*. Journal of Computer and System Sciences, 18 (1979), pp. 143-154.
- [10] D. Chaum. *Secret-ballot receipts: True voter-verifiable elections*. Security and Privacy Magazine, IEEE, Jan.-Feb. 2004. Volume: 2, Issue: 1, pp. 38-47.
- [11] S.J. Creese, M.H. Goldsmith, R. Harrison, A.W. Roscoe, P. Whittaker, and I. Zakiuddin. *Exploiting empirical engagement in authentication protocol design*. In D. Hutter and M. Ullmann, editors, Proceedings of 2nd International Conference on Security in Pervasive Computing (SPC'05), vol. 3450, LNCS, Boppard, Germany, April 2005. Springer.
- [12] S.J. Creese, M.H. Goldsmith, A.W. Roscoe, and M. Xiao. *Bootstrapping multi-party ad-hoc security*. In Proceedings of IEEE Security Track, 2006.
- [13] S.J. Creese, M.H. Goldsmith, A.W. Roscoe, and I. Zakiuddin. *The attacker in ubiquitous computing environments: Formalising the threat model*. In T. Dimitrakos and F. Martinelli, editors. Workshop on Formal Aspects in Security and Trust, Pisa, Italy, September 2003. IIT-CNR Technical Report.
- [14] S.J. Creese, M.H. Goldsmith, A.W. Roscoe, and I. Zakiuddin. *Security properties and mechanisms in human-centric computing*. In P. Robinson, H. Vogt, and W. Wagealla, editors. Privacy, Security and Trust within the Context of Pervasive Computing, Kluwer International Series in Engineering and Computer Science. Springer, 2004. Proceedings of Workshop on Security and Privacy in Pervasive Computing, Wien, April 2004.

- [15] C. Gehrman, C. Mitchell, and K. Nyberg. *Manual Authentication for Wireless Devices*. RSA Cryptobytes, vol. 7, no. 1, pp. 29-37, 2004.
- [16] C. Gehrman and K. Nyberg. *Security in personal area networks*. In C. J. Mitchell, editor, *Security for Mobility*, pp. 191-230. IEE, London, 2004.
- [17] S.W. Golomb. *Shift Register Sequences*. ISBN: 0894120484, Aegean Park Press, 1981.
- [18] M.T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. *Loud and Clear: Human-Verifiable Authentication Based on Audio*. 26th IEEE International Conference on Distributed Computing Systems, (ICDCS'06).
- [19] J. Hastad, R. Impagliazzo, L.A. Levin, and M. Luby. *A Pseudorandom Generator from any One-way Function*. SIAM J. Comput., 1999.
- [20] J.-H. Hoepman. *Ephemeral Pairing on Anonymous Networks*. In D. Hutter and M. Ullmann, editors, *2nd International Conference on Security in Pervasive Computing (SPC'05)*, vol. 3450 on LNCS, pp. 101-116, Boppard, Germany, April 2005. Springer.
- [21] J.-H. Hoepman. *Ephemeral Pairing Problem*. In 8th Int. Conf. Fin. Crypt., LNCS 3110, Springer, pp. 212-226.
- [22] International Organisation for Standardisation, Geneve, Switzerland. *ISO/IEC 1st CD 97986, Information technology—Security techniques—Entity authentication—Part 6: Mechanisms using manual data transfer*, December 2003.
- [23] M. Jakobsson and S. Wetzel. *Security Weaknesses in Bluetooth*. CT-RSA 2001, LNCS vol. 2020, Springer-Verlag, pp. 176-191, 2001.
- [24] G.A. Kabatianskii, B. Smeets, and T. Johansson. *On the cardinality of systematic authentication codes via error-correcting codes*. IEEE Transactions on Information Theory, IT-42 (1996), pp. 566-578.
- [25] H. Krawczyk. *LFSR-based Hashing and Authentication*. CRYPTO 1994, LNCS vol. 839, pp. 129-139.
- [26] H. Krawczyk. *New Hash Functions For Message Authentication*. EUROCRYPT 1995, LNCS vol. 921, pp. 301-310.

- [27] S. Laur and K. Nyberg. *Efficient Mutual Data Authentication Using Manually Authenticated Strings*. Volume 4301 on LNCS, pages 90-107, Dec 2006. Springer.
- [28] S. Laur, N. Asokan and K. Nyberg. *Efficient mutual data authentication using manually authenticated strings: Extended version*. Cryptology ePrint Archive, Report 2005/424, 2006.
- [29] G. Lowe. *Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR*. Tools and Algorithms for the Construction and Analysis of Systems, LNCS vol. 1055, Springer Verlag, pp. 147-166, 1996.
- [30] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton Computer Science Notes. Princeton University Press. January, 1996
- [31] A. Madhavapeddy, D. Scott, R. Sharp, and E. Upton. *Using Camera Phones to Enhance Human-Computer Interaction*. Proceedings of Ubiquitous Computing (UbiComp 2004), 2004.
- [32] Y. Mansour, N. Nisan, and P. Tiwari. *The Computational Complexity of Universal Hashing*. Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, pp. 235-243. May 1990.
- [33] R. Mayrhofer and M. Welch. *A Human-Verifiable Authentication Protocol Using Visible Laser Light*. Availability, Reliability and Security. ARES 2007.
- [34] J.M. McCune, A. Perrig, and M.K. Reiter. *Seeing is Believing: Using Camera Phones for Human-Verifiable Authentication*. IEEE Symposium on Security and Privacy, May 2005. An early version appears as Computer Science Technical Report CMU-CS-04-174, School of Computer Science, Carnegie Mellon University, November 2004.
- [35] T. Moran and M. Naor. *Polling with Physical Envelopes: A Rigorous Analysis of a Human-Centric Protocol*. EUROCRYPT 2006.
- [36] L.H. Nguyen and A.W. Roscoe. *Efficient group authentication protocol based on human interaction*. Proceedings of Workshop on Foundation of Computer Security and Automated Reasoning Protocol Security Analysis, pp. 9-31, August 2006.
- [37] L.H. Nguyen and A.W. Roscoe. *Authenticating ad hoc networks by comparison of short digests*. To appear in journal of Information and

Computation. Special Issue of Information and Computation on Computer Security: Foundations and Automated Reasoning. Submitted in November 2006, accepted on 20th July 2007.

- [38] L.H. Nguyen and A.W. Roscoe. *New theoretical bounds for Universal hash functions*. Manuscript is available.
- [39] L.H. Nguyen and A.W. Roscoe. *Empirically authenticated one directional message transmission*. Manuscript is available.
- [40] L.H. Nguyen and A.W. Roscoe. *Efficient digest function based on Toeplitz matrix and integer multiplication*, in preparation.
- [41] S. Pasini and S. Vaudenay. *SAS-based Authenticated Key Agreement*. Public Key Cryptography - PKC'06: The 9th international workshop on theory and practice in public key cryptography, LNCS vol. 3958, pp. 395-409, Springer, 2006.
- [42] S. Pasini and S. Vaudenay. *An Optimal Non-interactive Message Authentication Protocol*. Topics in Cryptology - CT-RSA'06: The Cryptographers' Track at the RSA Conference 2006, LNCS vol. 3860, pp. 280-294, Springer, 2006.
- [43] R. Pass. *On Deniability in the Common Reference String and Random Oracle Model*. CRYPTO '03, LNCS vol.2729, pp.316–337, Springer 2003.
- [44] T. Peyrin and S. Vaudenay. *The Pairing Problem with User Interaction*. SEC 2005, pp. 251-266.
- [45] A.W. Roscoe. *Human-centred computer security*. See: <http://web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications/113.pdf>, 2005.
- [46] N. Saxena, J.-E. Ekberg, K. Kostianen, and N. Asokan. *Secure Device Pairing based on a Visual Channel*. In the Proceedings of the IEEE Symposium on Security and Privacy, 2006.
- [47] F. Stajano and R. Anderson. *The resurrecting duckling: Security issues for ad-hoc wireless networks*. Security Protocols 1999, LNCS vol. 1976, Springer-Verlag, pp. 172-194, 1999.

- [48] F. Stajano and R. Anderson. *The Cocaine Auction Protocol: on the Power of Anonymous Broadcast*. In the Proceedings of the 3rd International Workshop on Information Hiding, Lecture Notes in Computer Science, Springer-Verlag, 1999.
- [49] F. Stajano. *Security for Ubiquitous Computing*. Wiley, 2002.
- [50] F. Stajano and R. Anderson. *The Resurrecting Duckling – What Next?* In the Proceedings of the 8th International Workshop on Security protocols, Lecture Notes in Computer Science, Springer-Verlag, 2000.
- [51] D.R. Stinson. *Universal Hashing and Authentication Codes*. Advances in Cryptology - Crypto 1991, LNCS vol. 576, Springer-Verlag, pp. 74-85, 1992.
- [52] J. Suomalainen, J. Valkonen, and N. Asokan. *Security Associations in Personal Networks: A Comparative Analysis*. In the proceedings of the Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks, Cambridge, UK, July 2007. Vol. 4572 of LNCS, Springer. Nokia Research Center, Technical Report NRC-TR-2007-004, January 2007.
- [53] E. Uzun, K. Karvonen, and N. Asonka. *Usability Analysis of Secure Pairing Methods*. Nokia Research Center, Technical Report NRC-TR-2007-002, January 2007. In the Usable Security (USEC '07) workshop, Scarborough, Trinidad/Tobago, February 2007.
- [54] J. Valkonen, N. Asokan, and K. Nyberg. *Ad Hoc Security Associations for Groups*. In Proceedings of the Third European Workshop on Security and Privacy in Ad hoc and Sensor Networks, Hamburg, Germany, September 2006. Vol. 4357 of LNCS, Springer.
- [55] S. Vaudenay. *Secure Communications over Insecure Channels Based on Short Authenticated Strings*. Advances in Cryptology - Crypto 2005, LNCS vol. 3621, Springer-Verlag, pp. 309-326, 2005.
- [56] 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, pp. 265-279, 1981.
- [57] A.F. Webster and S.E. Tavares. *On the Design of S-Boxes*. CRYPTO 1985, LNCS vol. 218, Springer Verlag (1986), pp. 523-534.

- [58] Ford-Long Wong and F. Stajano. *Multi-channel Protocols*. Proceedings of the 13th International Workshop on Security Protocols, Cambridge, England, 20-22 Apr 2005, LNCS.
- [59] Ford-Long Wong and F. Stajano. *Multi-channel Security Protocols*. To appear in IEEE Pervasive Computing, 2007, IEEE Press.

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

In order to illustrate that if the information $leader(L, A)$ were not be communicated over the empirical channel, the 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 leader³¹. 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 agree. So they agree on the final digest, each believing the others to be the leader. Of course this works equally well with any two disjoint sets of "leaders". This attack works when we can block the commitment sent via 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 4.2 cannot be weakened further to have two slaves, even when all the nodes in the protocol are trustworthy.

If there are two slaves A and B , and one leader called C , who wants to have its information $INFO_C$ authenticated to the two slaves. In the first run α of the protocol, the intruder $I(A, B)$ impersonates slaves A and B to communicate with the leader C , and comes up with two random keys k'_A and k'_B .

³¹ A will receives $INFO'_B$ saying that B is the leader and vice versa.

- 1.α. $C \longrightarrow_N I(A, B) : INFO_C, longhash(k_C)$
- 2.α. $I(A) \longrightarrow_N C : k'_A$
 $I(B) \longrightarrow_N C : k'_B$
- 3.α. $C \longrightarrow_N I(A, B) : k_C$

After C sends out its own key k_C , the intruder can now determine the final digest value of run α that C is going to compare over the empirical network. If $k_S = k'_A \oplus k'_B$ then in order to fool the slaves A and B into thinking the fake $INFO'_C$ is authentic, the intruder needs to use brute force search to find k'_S such that the digests of both runs come out to be equal to each other:

$$digest(k_C \oplus k_S, INFO_C) = digest(k_C \oplus k'_S, INFO'_C)$$

This should not take too much time as the bit length of the digest output is very short, for example: 16 bits. Once he finds out k'_S he then starts the second run β playing the role of the leader $I(C)$ to talk to slaves A and B as well as modifying the original keys of the two slaves as follows

- 1.β. $I(C) \longrightarrow_N A, B : INFO'_C, longhash(k_C)$
- 2.β. $A \longrightarrow_N I(B, C) : k_A$
 $B \longrightarrow_N I(A, C) : k_B$
 $I(A) \longrightarrow_N B : k_B \oplus k'_S$
 $I(B) \longrightarrow_N A : k_A \oplus k'_S$
- 3.β. $I(C) \longrightarrow_N A, B : k_C$

After the key k_C is revealed to A and B , all three nodes should be able to empirically agree on two equal digests that have different antecedents, or another word the slaves accept $INFO'_C$ faked by the intruder.

- 4.β. $A, B \longrightarrow_E C : digest(k_C \oplus k'_S, INFO'_C)$
 $A \longleftrightarrow_E B : digest(k_C \oplus k'_S, INFO'_C)$
- 4.α. $C \longrightarrow_E A, B : digest(k_C \oplus k_S, INFO_C)$

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