

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}

February 17, 2009

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. In studying these protocols, we will discover that their security is underlined by the idea of *commitment without knowledge*, which is refined by two protocol design principles introduced in this survey.

Contents

1	Introduction	3
1.1	Notation	5
1.2	Communication links	7
1.3	Cryptographic primitives	8
1.3.1	Commitment scheme	8
1.3.2	Short hash and digest functions	10
1.4	Attack model	12
1.5	Cost model	13
1.5.1	Human effort	13
1.5.2	Computation cost model of cryptographic primitives	13
2	Non-interactive protocols	16
2.1	Long authentication string over the empirical channel	16
2.2	Short authentication strings over strong empirical channels	18
2.3	Improved version of (V-)MANA I	20
3	Interactive protocols	21
3.1	Multiple empirical short authentication strings	22
3.2	Indirect binding	25
3.2.1	Indirect pairwise	25
3.2.2	Hybrid protocol	27
3.3	Direct binding pairwise protocols	28
4	Group protocols	32
4.1	Some existing direct binding group protocols	32
4.2	Indirect binding group protocol	35
4.3	Modified versions of HCBK and SHCBK	36
5	Conclusions and further work	39
5.1	Efficiency	39
5.2	Short-term public key cryptography	40
5.3	Conclusions	42
5.4	Future research	43
A	The importance of empirical display of $leader(L, A)$ in Hybrid HCBK	48
B	Attack on group protocol with two slaves	49
C	Security proofs of Improved MANA I	50
C.1	Security proof of Improved (V-)MANA I (direct binding)	51
C.2	Improved (V-)MANA I (indirect binding) and security proof	53
C.3	Improved (V-)MANA I (Diffie-Hellman style) and security proof	55

1 Introduction

In this paper, we give a survey of authentication protocols using manual transfers 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, international standards and patent applications of a variety of such protocols, frequently by groups working completely independently of each other [63, 16, 18, 66, 39, 40, 12, 22, 62, 49, 50, 51]. 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 agree on, for example, the number and identities of protocol participants, since the information will significantly reduce the waiting time in a protocol session.

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

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 the Introduction. A simple model for the computation cost of these cryptographic primitives and an attack model are provided to assess the complexity and security of these protocols as we move along.

Although the majority of authentication protocols considered here have been independently introduced by a number of research groups using different notations, this survey will demonstrate that their security is derived from the idea of *commitment without knowledge*, formally defined in Section 1.3.1. What it does is to force protocol participants to be (jointly) committed to some value without knowing what it is until they reveal their respec-

¹There has been another survey written by Suomalainen et al. [60] 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.

tive shares of the decommitment in a later stage of a protocol run. This committed value will, in turn, always be *instrumental* in the computation of the SASs² compared by humans, and therefore the parties' state of knowledge of the SASs is a uniform distribution, i.e. this is the key to defeating any causal influence such as birthday attacks as well as ensuring that combinatorial search and multiple-shot attacks do not gain any advantage over one-shot and guess attacks. We will see that there are two different approaches of achieving this goal depending on whether the authenticated information is *directly* or *indirectly* bounded to SASs, as studied in Sections 3.2 and 3.3 respectively. In particular, the *direct* information binding strategy will be refined by two protocol design principles, termed **P1** and **P2**, in Sections 3.3 and 4.1.

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 [16, 17, 18]. 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.

While there has been much recent literature on pairwise protocols, we find it strange that apart from the authors' group [12, 48, 39, 40] and Valkonen et al. [62] 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.

In Section 5.1, the computation cost of every protocol will be gathered into three tables that clearly demonstrate two things:

- Interactive schemes (pairwise or group authentication) can be much more efficient in human work than non-interactive ones (one-way authentication);
- Although the security of the majority of protocols rely on the *commitment without knowledge* idea, the use of *direct binding* to achieve this goal has a clear advantage which arises from the potential to use a digest function designed to produce only the small number of bits required for empirical comparison as opposed to a conventional cryptographic hash used in *indirect binding*.

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: [].

²The committed value could be either used directly as the SAS or inputted as a private key of a digest, universal hash or MAC function.

- 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*.

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

To avoid the terminological longeurs given in the Introduction, readers can go straight to key protocols such as (V-)MANA I and its improved versions in Section 2.2 which give a concrete goal for the best protocol to achieve. They can come back to the Introduction to look for definitions of many functions, notations and concepts, which are comprehensively summarised there, along the way.

1.1 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) .

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 network³. In addition, we

³We 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.

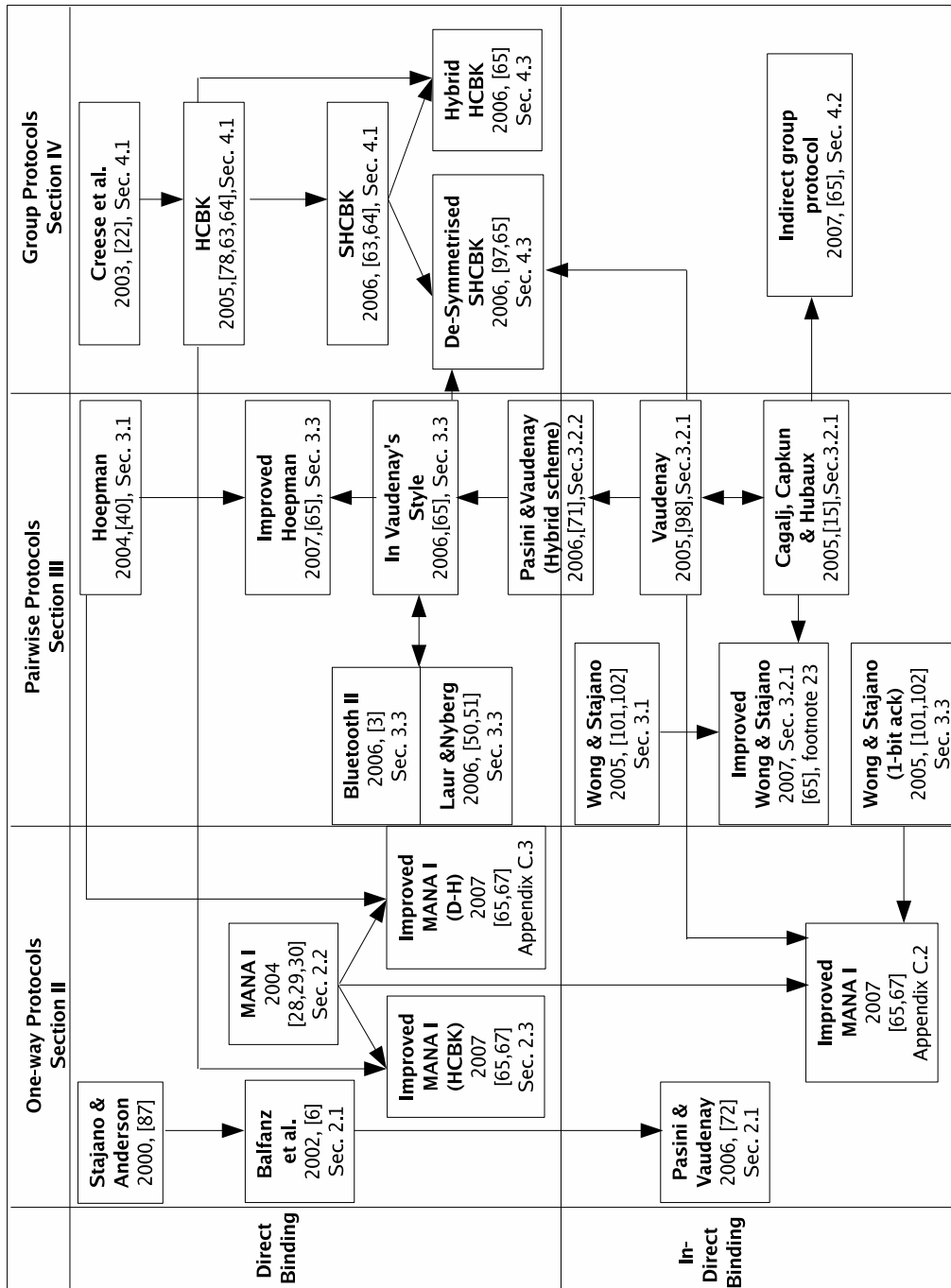


Figure 1: Summary of all protocols.

refer to *INFOS* as the concatenation in alphabet order of all the distinct pairs parties want to authenticate: NM words if they are all size M .

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

1.2 Communication links

In some cases, the protocols we quote from other papers are changed in appearance because we seek to use a consistent nomenclature and notation: we do not want a single piece of notation to have inconsistent meanings. One respect in which previous works and papers vary is in the assumptions they make about the empirical or authentic 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, which are presented in a descending order of generality.

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

- \longrightarrow_N , the normal Dolev-Yao network where all messages transmitted between the laptops in this channel can be overheard, deleted or modified by the intruder. Examples of this channel are the Internet, WiFi, or local network.
- \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 [63, 45]. A typical example of this channel could be telephone conversation, voice mail or messages, where messages can be delayed, blocked or replayed, but cannot be forged by the intruder.
- \longrightarrow_{BE}^t , this is similar as \longrightarrow_{WE} except that messages cannot take more than time t to arrive and cannot be replayed. In other words, no empirical message can be accepted more than t time units after it was sent. Such a channel might be implemented over a reliable medium with a known bound on transmission, or over an unreliable one with the addition of some sort of time-stamp. The latter might make sense if the empirical message is sent by some video means, but otherwise would add significantly to the communication burden. We will call this a *bounded delay empirical channel*.
- \longrightarrow_E is the type of empirical channel assumed in [39, 40]. Messages transmitted over the channel cannot be mistaken or delayed from one to another session. To some extent, this implies that \longrightarrow_E is a special case of \longrightarrow_{BE}^t where t is chosen to be some lower bound on the length of time between one session and a later one. This is the type we use most often. Sometimes the two-way arrow \longleftrightarrow_E is used to indicate (possibly) the same message is transmitted in both directions.

An example of this channel is manual data transfers [16, 17, 18], i.e. human users manually copy data from one to another device via their in/output interfaces such as screen, monitor and keyboard. In this case, empirical messages cannot be mistaken or delayed from one to another session because: (1) the humans involved are not away at any time during a protocol run, and (2) each device normally only has one in/output interface for displaying or reading data.

- \rightarrow_{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. This implies that \rightarrow_{SE} is the same as \rightarrow_{BE}^t where $t \approx 0$. We term this a *strong empirical channel* (or a strong authentication channel). This was also defined in [63, 45]. Face to face human conversation is an example of this channel.

1.3 Cryptographic primitives

We will be using a variety of cryptographic primitives, related to hashing. Since bitlengths of hash functions vary widely, for clarity, they will be subscripted with the number of output bits. For example:

- $hash_{160}(X)$ and $longhash(X)$ refer to standard cryptographic hashes, such as SHA, MD5 or TIGER, intended to be inversion-resistant, strongly collision-resistant and non-malleable. In common with the literature, we will generally assume that they have at least $B = 160$ bits.
- $hash_{80}(X)$ and $hash_b(X)$ will be functions with sufficiently many bits to offer weak or short-term versions of these security properties of $hash_{160}(X)$ or $longhash(X)$. Here b is in the range of [16,32].
- $digest(k, X)$ 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 in Section 1.3.2 as well as an additional property often required of the short output hash functions.
- $h_k(X)$ will be a universal hash function of X keyed by k .

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

1.3.1 Commitment scheme

Following is the definition of a commitment scheme used in some of the protocols we consider.

A commitment scheme [8]: binds a random and secret nonce R and some public data $INFO$ together by publishing the *commitment* c . Eventually sending the *decommitment* d reveals R , and binds this value firmly to $INFO$ in the eyes of the receiver. It consists of two components, namely $commit()$ and $open()$:

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

The following provides more information about commitment schemes as well as the idea of *commitment without knowledge* and its restrictive version called *joint commitment without knowledge*, which underly the security of nearly every protocol discussed in this paper.

Formally, a commitment scheme will have the two following properties:

- **Hiding:** it is infeasible for the intruder to determine from $\langle c, INFO \rangle$ the value of R before the decommitment d is revealed.
- **Binding:** it is infeasible for the sender to change the value of $INFO$ to which it has committed once the commitment stage is over.

In other words, given $(INFO, R, c, d)$ where $c \parallel d = \text{commit}(INFO, R)$, it must be infeasible to compute a different $INFO'$ such that $c \parallel d' = \text{commit}(INFO', R)$, where d' can be equal to or different from d .

The bitlength of R will be short, e.g. $b \in [16, 20]$ or even zero bits, in all protocols considered. To prevent brute-force and collision search, the commitment scheme (i.e. $\text{commit}()$) needs to extend R by a randomly chosen secret nonce R' so that the combination $\langle R \parallel R' \rangle$ does have enough entropy as demonstrated by an example given below.

A commitment scheme is desired to be at least as secure as a standard cryptographic hash function $\text{hash}_{160}()$ or $\text{longhash}()$, and so each of the values c , d and the combination $\langle R \parallel R' \rangle$ need to have the same bitlength as the output of a cryptographic hash function, i.e. $B = 160$ bits. In practice, a commitment schemes is usually built from a pseudorandom function such as a hash function, and therefore they have the same computational complexity, which will be discussed in more details in Section 1.5.2.

One simple example of such a commitment scheme, introduced in [46] by Pass, is described below.

- **Committing:** to bind some public information $INFO$ and a short random secret key R of b bits together, the algorithm picks another secret random nonce $R' \in_{\text{random}} \{0,1\}^{B-b}$. It then sets $d = R \parallel R'$ and $c = \text{longhash}(INFO \parallel d)$, i.e. $\text{commit}(INFO, R) = c \parallel d$. The committer then publishes the commitment c .⁴
- **Decommitting:** to decommit or to use $\text{open}(INFO, c, d) = R$, the committer first publishes the decommitment d . Any one can extract R from d (i.e. the first b bits) and verifies $c = \text{longhash}(INFO \parallel d)$.

The conventional understanding of a commitment scheme has been that the committer knows the value to which he or she is committed. In this paper, however, we will see several cunning uses of the commitment idea but *without* the knowledge of the committed value from protocol participants, who can be either the committer or other parties. We will see this can be done by distinguishing carefully between when nodes are committed without knowledge to a value and when they know it.

The first use of this idea is called *commitment without knowledge*, which directly influences the design of HCBK in Section 4.1, and some of the one-way and non-interactive protocols in Section 2.

Commitment without knowledge: Upon the receipt of a cryptographic hash, the receivers will be committed to the digest of the hash value's antecedents (e.g. a private key k) in combination with some public information $INFOS$ without actually knowing the digest (i.e. $\text{digest}(k, INFOS)$) and any of the cryptographic hash's antecedents.

⁴Instead of using a cryptographic hash function $\text{longhash}()$, one can switch to universal hash function, i.e. $c = \text{longhash}(INFO \parallel d)$ is replaced by $c = h_d(INFO)$. This of course should also apply to decommitting.

The chief purpose of this idea is to ensure that a powerful attack which might consist of combinatorial search and multiple protocol runs does not gain any advantage over a one-shot or guess attack, and this will be explained in more details in Section 1.4 which presents definitions of a variety of attacks carried out by the intruder.⁵

The same properties also apply to a more restrictive use of the idea, called *joint* commitment without knowledge, which influences the design of nearly every interactive protocol, i.e. pairwise and group authentication schemes in Sections 3 and 4.

Joint commitment without knowledge: every protocol participant is *jointly* committed without knowledge to some value (a SAS) by sending out their respective shares of the commitment. The committed value remains unknown to any one until all of the participants reveal their shares of the decommitment.

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

We will see later that there are two different strategies of achieving commitment without knowledge, i.e. *direct* and *indirect* information binding approaches which are formally introduced in Section 3. In addition, the combination of commitment without knowledge and *direct binding* approach will be refined by the principle **P2** to design several group authentication protocols in Section 4.1.

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 search such as the birthday attacks. In this paper, we will see various cunning uses of new functions that, like cryptographic hashes, are intended to randomise and convey no useful information about the preimage. However, their outputs are significantly shorter, since they will always produce values of short authentication strings (SASs) transmitted over the empirical channels, which are severely limited in bandwidth.

We will see two variants on this idea: the simpler is what we call a short hash: $hash_b()$. This has a single argument, and is intended to be uniformly or near-uniformly distributed over its b -bit range as its argument varies. Since the hash output is too short, it is impossible to have properties such as collision and inversion resistances like in cryptographic hash functions. What is required instead is the *strict avalanche criterion*, which states that any number of bit-changes in the input has an equal (but non-negligible) influence on every bit of the output [65].

In some of the protocols we consider, we need to construct a b -bit digest of *INFOS* and some key k . Similar to $hash_b()$, digests cannot have the usually specified properties of

⁵In the definition of *commitment without knowledge*, we specifically focus on being committed without knowledge to digest values because every protocol which directly uses this idea in this paper makes use of digest functions. Of course, the idea also applies to being committed without knowledge to private keys, nonces or outputs of some functions applying to these values as can be seen in *joint commitment without knowledge*.

a cryptographic hash, namely non-invertability and collision-freeness. On the other hand, we do require that a high degree of randomness arises from the use of the key k , as set out below and in [39, 40].

Specification of digest functions [39, 40]
1. For any fixed m and digest y : $\Pr_k[\text{digest}(k, m) = y] \leq 2^{-b}$
2. For any fixed θ and any pair of different messages (m, m') : $\Pr_k[\text{digest}(k, m) = \text{digest}(k \oplus \theta, m')] \leq 2^{-b}$

The rationale for these two specifications, especially the use of “ $\oplus \theta$ ”, will become apparent when we analyse group protocols such as SHCBK in Section 4.1. The digest specification is similar to *universal* hash functions, except the probability of digest collisions relative to different keys is also considered, i.e. the way digest keys are agreed between nodes in SHCBK protocol can be manipulated such that different nodes’ keys may be relatively shifted by a θ known to the intruder. The inclusion of the θ shift is therefore to ensure that this type of activity can never benefit the intruder.

Although both $\text{hash}_b()$ and $\text{digest}()$ are less secure than cryptographic hashes, they are potentially faster to compute, thanks to their short outputs. More details about their comparative speed performance can be found in Section 1.5.2. Designing (universal) hash functions has an exciting and long history in computer science and cryptography, however there does not seem to be much literature on the study and exploitation of short output to speed up computation efficiency. As a result, we believe that there should be a potential of new constructions for digest functions or adaptation of existing work to acquire computation efficiency. And this trend can be demonstrated as follows.

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 [18, 44, 28, 29, 17, 4, 40]. To the best of our knowledge, the only ones that explicitly exploit the short output to improve efficiency as well as being proved to satisfy the above specification are based on the idealised framework invented by the authors [39, 40], which are adaptations of several well-studied universal hash constructions of Mansour et al. [34], and Krawczyk [26, 27]. We there provide several algorithms using fully random numbers derived from the digest key, which can be simulated in practice by a pseudorandom number generator (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 Section 1.5.2 on the computation cost model. Our algorithms have no restriction on the length of the object (typically *INFOS*) being digested.

In contrast, several research authors in [44, 28, 17, 18] make use of universal hash functions presented in [59, 25, 7] to compute digest functions. These algorithms 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) by using a cryptographic hash prior to running the algorithms themselves, which turns out to be neither ideally secure nor cost effective, [40]. Alternatively, others [4, 44] 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.

1.4 Attack model

In Section 1.1, we have defined the intruder’s power over data transmitted over all kinds of channels used in the family of protocols. In addition to that, the following are definitions of attacks performed by the intruder, and which will be considered as we move along.

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

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

Our aim is to ensure that powerful or multiple-shot attacks give the intruder no advantage over a one-shot or guess attack on this family of protocols.⁶ This goal is achieved because once every protocol participant is (*jointly*) *committed without knowledge* to some short authentication strings (SASs, e.g. a digest value transmitted over empirical channels), then there is not any effective way in which the agents can determine anything about the value – the agents’ state of knowledge of the SAS is a uniform distribution. Moreover, depending on the type of empirical channels used in the family of authentication, we have:

- when SASs are transmitted over (strong) empirical channels (\longrightarrow_E and \longrightarrow_{SE}), they cannot be mistaken or delayed from one to another session. Hence, every protocol run is themselves independent as pointed out by several authors in [63, 45, 30, 42], and for any q we have:

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

This model is, however, very conservative, and is only valid when the intruder launch attacks on many (or perhaps q) *different* pairs or groups of parties. In practice, once a human has noticed a short authentication string disagreement, he or she will be suspicious or aware that an attack is taking place provided implementation is reliably constructed. This will mean that the human will either allow no more attempts or require longer authentication strings, i.e. extending the SAS by 1 bit after each mismatch makes the probability of a successful powerful attack be upper bounded by $2^{b-1} = \sum_{l=b}^{\infty} 2^{-l}$, where 2^{-l} is the likelihood of a successful one-shot attack on a l -bit SAS protocol implementation.

We will see later in Appendix C (Theorem 1) how to formalise this statement to give proofs of security for various protocols introduced in this paper.

- when SASs are transmitted over weak empirical channels \longrightarrow_{WE} , they can be stalled and replayed in later protocol runs. Hence, if the intruder launches a q -shot attack,

⁶This is similar to the goal of password-based authentication protocols, which have been studied extensively to date.

which involves q concurrent runs of either two parties in a pairwise scheme or two subsets of nodes in a group protocol, then the number of protocol sessions from the intruder’s view will become q^2 . This implies that:

$$\Pr(\text{a successful } q\text{-shot attack}) \leq q^2 \times \Pr(\text{a successful one-shot attack})$$

The origin of data transmitted over weak empirical channels cannot be forged, thus to take advantage of the birthday paradox the intruder will have to launch attacks on the *same* parties who are responsible for delivering the SASs. This therefore will make this model only accurate when q the number of protocol sessions or SASs, which are delayed concurrently, is small because in practice humans are very sensitive to delays or mismatches in the SASs as pointed out earlier.

Since the protocol design can reduce the probability of a successful attack to the chance of a one-shot attack, for simplicity we will refer successful attacks considered in all protocols to attacks that only involve a single protocol run, i.e. a one-shot attack.

We are also interested in *chosen plain-text attacks* [54] under which the intruder can influence data trustworthy parties want to authenticate. Although this attack might seem unrealistic, it is desirable that protocols are immune to it, i.e. it will become useful when we analyse protocol of Balfanz et al. in Section 2. Since the attack relies on combinatorial manipulation, we refer to it as a special case of the combinatorial search attack.

In authentication protocols where parties only want to authenticate their public-key-like information, there is no need to distinguish between honest and dishonest nodes. Conversely, every one has to be trustworthy in a key agreement protocol, whether it is pairwise or group schemes. More discussion about this issue could be found at the begin of Section 4.

1.5 Cost model

It seems reasonable to measure the efficiency of the family of protocols in two ways: the amount of empirical or human effort required to complete them; and the amount of processing required at the nodes. The following models for human effort and computation cost are adapted from two of our papers [39, 40].

1.5.1 Human effort

Our main measure of empirical work is the number of bits of the short authentication strings that are transmitted over the empirical channels. Throughout this paper, we always attempt to optimise the amount of security one can obtain from a given amount of empirical (human) communication. This is achieved when the only thing empirically communicated is a single short string, i.e. a b -bit SAS, and the maximum probability of any successful one-shot attack is upper bounded by 2^{-b} . We will see in later sections that this bound is attainable, provided we can discount the probability of strong cryptographic primitives being broken.

1.5.2 Computation cost model of cryptographic primitives

It is essential to optimise the human work in the families of protocols, but at the same time, we also want to minimise the computational cost. We are aware that the cost of agreeing a

private key through exponentiation (in Diffie-Hellman’s style) or public key cryptography always overtakes the cost of bootstrapping authenticity. However, if the authentication phase is carried out early on lightweight devices prior to key agreement achieved on more powerful devices at a much later stage. Then it is desirable to minimise the computation cost of the authentication protocols done on lightweight devices, whose computation power is severely limited.

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

Let B and W be the number of bits and respectively words required to hold a long hash value. It is normal that $B = 160$ bits, so we assume $W = B/w = 5$, here we assume that a word consists of $w = 32$ bits. Many researchers in [63, 16, 67, 40] suggest 15 or 16 bits are reasonable choices for b , the width of the digest output, which is rounded up to 1 word in our analysis. We assume that nonces, keys (used in a commitment scheme and as input of $longhash()$) and other strong cryptographic values, such as a commitment c and a decommitment d , have the same bitlength B .

It is clear that the cost of computing the b -bit output $hash_b(m)$ tends to increase linearly with the length of m , since the majority of customised cryptographic hash functions, such as MD5 and SHA, are *iterative* in nature, i.e. they operate on fixed size data blocks, such as 512 bits, in sequence, and so can take an arbitrary length input. It also seems clear that the cost will increase at least linearly with the output length b , as can be demonstrated by considering the type of operation of customised and block ciphers based hash functions widely used in practice: (1) The Merkle-Damgard construction [37], including MD4/5, SHA-0/1/256/384/512, always has an *internal state* whose size is equal to the output size, and this is updated by linear or bitwise operators (i.e. +, AND, OR, XOR and rotation) in each loop of the algorithms; and (2) In block ciphers based hash functions [37], such as Davies-Meyer, Matyas-Meyer-Oseas and Miyaguchi-Preneel, the block cipher which is a one-way compression function normally has a same size in bits as the hash output. As a result, 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:

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

Therefore, we will adopt that assumption in the computation analysis of hash functions. 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.⁷ Although the computational cost model does not take into account the number of clock cycles and implementation-specific (i.e. software or hardware), it does give an approximate comparison between the cost of computing long and (very) short output functions, for example, cryptographic hash versus digest function as can be illustrated in Table 1.

With respect to the cost of computing a digest. As defined in Section 1.3.2, $digest(-, -)$ is a family of short hash functions indexed by a key k . Even though the key bitlength

⁷In practice, one often constructs a variable output-size hash function based on the idea of Key Derivation Function (KDF). 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.

Cryptographic primitive	Computation cost
$longhash(INFO)$ or $hash_{160}(INFO)$	WM
$longhash(k)$ or $hash_{160}(k)$	WW
$hash_b(INFO)$	M
$digest(INFO)$	M
$commit(INFO, R)$	WM
$open(INFO, c, d)$	WM

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

might be significantly longer than the hash output in several constructions of universal hash functions invented to date [26, 27], it normally does not play any significant part in the computation. Consequently, key length will not have a big impact on the computation cost.⁸ Hence, we assume the cost model of a digest or universal hash functions is similar to a hash function, which is mainly dependent on the lengths of the input message and the digest output.

A commitment scheme defined in Section 1.3.1 inputs a message of length K bits or $\lceil K/w \rceil = M$ words and a pair of nonces $(R \parallel R')$ that add up to $B = 160$ bits or W words. Since the pair of nonces play the same role as key k in digest computation, we assume that a commitment scheme takes M words as input.⁹ These are true in both operations used to calculate a commitment and open/verify the commitment. We therefore conclude that the computational costs of computing and verifying a commitment are equal to each other as well as being equal to the cost of computing hash functions. The latter is true because commitment schemes are normally built from pseudorandom functions such as hash functions [46].

Table 1 summarises the computational cost of all cryptographic primitives introduced in this section. While the table 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 come from the same model.

Every protocol in this paper, except the first one in Section 4, only uses long or short hashes, commitment schemes or digest functions. For this reason, we shall apply the simple model to compute the cost for each of them as we move along. In Section 5.1, all of the computational costs and human effort will be put into tables summarising the efficiency of each class of protocols.

⁸In fact the longer is the key, the fewer the number of random bits we have to generate in our proposed construction of digest functions as well as universal hash constructions of Krawczyk [26, 27], and subsequently the better.

⁹In practice, a commitment scheme, such as one introduced in Section 1.3.1, takes both M -word messages and a 160-bit random nonce as its inputs, i.e. these are concatenated before being inputted to a cryptographic hash function: $longhash(INFO \parallel R)$. However, it has been noted that $longhash(INFO \parallel R)$ could be replaced by a universal hash function keyed by R , i.e. $h_R(INFO)$ to reduce the input length, and therefore computational cost. As a result, to give a fair comparative analysis which is independent of implementation, we will stick to this assumption.

2 Non-interactive protocols

We examine some protocols attempting to transmit a, possibly very long, message from one party to another efficiently in such a way that the origin and integrity of the message are authenticated. These all use just one-way communication and authentication strings and help to illustrate the power of authenticated empirical channels.

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

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

2.1 Long authentication string over the empirical channel

The above analysis of the use of signatures shows they are closely analogous to the following one-way authentication protocol, devised by Balfanz et al. [5]. In this scheme, A wants to authenticate its information $INFO_A$ to B .

Balfanz et al. non-interactive protocol, [5]	
1.	$A \rightarrow_N B : A, INFO_A$
2.	$A \rightarrow_{WE} B : longhash(A, INFO_A)$ B verifies the longhash.

Under the assumption of a *chosen plain-text attack* in Section 1.4, information $INFO_A$ is under the control of the intruder. Hence, the intruder might carry out an off-line search to find a different pair $(INFO_A, INFO'_A)$ both hashing to the same value.¹⁰

1. $A \rightarrow_N I(B) : A, INFO_A$
 $I(A) \rightarrow_N B : A, INFO'_A$
2. $A \rightarrow_{WE} B : longhash(A, INFO_A)$

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

As a result of a single longhash whose input and output lengths are M and W words, the computation cost is of order $WM = 5M$, thanks to the cost model in Section 1.5.2.

¹⁰In the original protocol [5], there is no restriction on the order of sending and receiving Messages 1 and 2.

In order to improve the number of authenticated bits, Pasini and Vaudenay [45] make use of a probabilistic commitment scheme¹¹ to commit to the authenticated information. The 80-bit hash of the commitment is then sent over the weak empirical channel.

Pasini-Vaudenay non-interactive protocol, [45]	
1.	$A \xrightarrow{N} B : c \parallel d = \text{commit}(A, INFO_A)$ B computes $A \parallel INFO_A = \text{open}(c, d)$
2.	$A \xrightarrow{WE} B : \text{hash}_{80}(c)$ B verifies the hash.

Here the hash function is required to be weakly collision resistant (i.e. the second preimage resistance property [53]: an intruder cannot find a second value v' such that $\text{hash}(v) = \text{hash}(v')$ for fixed v) as opposed to the strong collision resistance required for Balfanz et al. where both v and v' are allowed to vary.

In [45], Pasini and Vaudenay argue that this provides the same degree of authentication as the Balfanz et al. protocol, namely 2^{80} computation steps, because the probabilistic commitment scheme avoids the possibility of a birthday attack. At the point where A is influenced to use the given $INFO_A$, the intruder cannot know what a nondeterministic component (a hidden random nonce of 80 bits = $W/2$) that is injected by A into the commitment scheme will be, which is vital in obtaining a collision. Binding the information by a commitment scheme has the advantage of halving the number of empirical bits as well as halving the bitlength of the commitment scheme, due to the nondeterminism introduced. These together reduce the cost to $MW/2 + W^2/4 = 2.5M + 6.25$. As far as the author is aware, this is currently the best non-interactive scheme in terms of the number of empirical bits relative to the level of security obtained.

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

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

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

- There might be some mechanism which bounds the life of a delayed message. For

¹¹A commitment scheme was defined in Section 1.3.1. We note that there is no random nonce inputted into the commitment scheme used here. Therefore, its bitlength is assumed to be zero, and the commitment scheme has to generate a new long nonce (80 bits) every time the $\text{commit}()$ function is called. This is also the only time when an 80-bit commitment scheme is used. For all other employments of a commitment scheme, it is always 160-bit.

example, given sufficiently synchronised clocks, a time-stamp would produce a real bound on the delayability of the message.

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

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

It seems fair to remark that since even 80 bits will seem tedious for most humans to compare carefully, these one-way non-interactive protocols are not likely to find widespread use. Where it is humans who actually need to do this work: they would need to have a high level of commitment and possibly a well-designed user interface to ensure user compliance.

2.2 Short authentication strings over strong empirical channels

Gehrmann, Mitchell and Nyberg [16] took a different approach to preventing combinatorial search. They use empirical channels to transmit the b -bit output of a check function $\text{MAC}_k()$ ¹² together with a b -bit key that has been instrumental in its computation.

MANA I (Gehrmann, Mitchell and Nyberg), [16, 17, 18]	
1a.	$A \rightarrow_N B : A, \text{INFO}_A$
1b.	$B \rightarrow_E A : 1\text{-bit committed signal}$ A picks a b -bit random number k
2.	$A \rightarrow_E B : k, \text{MAC}_k(A \parallel \text{INFO}_A)$

To eliminate 1-bit empirical signals in MANA I,¹³ Vaudenay proposes to use a strong empirical channel (stall-free or instant delivery, and is denoted \rightarrow_{SE}) to send the key and the check-value.¹⁴ Thus $2b$ bits are transmitted in all. This idea turns the protocol into a non-interactive scheme. In the following description, we will modify the scheme slightly by using a digest function to compute the check-value. The rest of this analysis applies to both versions.

V-MANA I, [63, 45]*	
1.	$A \rightarrow_N B : A, \text{INFO}_A$ A picks a b -bit random number k
2.	$A \rightarrow_{SE} B : k, \text{digest}(k, A \parallel \text{INFO}_A)$ B verifies the digest.

¹²As suggested in [16, 17], a check function $\text{MAC}_k()$ can be implemented by either CBC-MAC or universal hash functions based on error correcting code, which is potentially less efficient than digest functions as discussed in Section 1.3.2 and [40]

¹³The 1-bit committed signal, which can be implemented by a red line or a single button, is not a primitive property of empirical channels. Its presence aims to indicate to A that B has received Message 1, which could be either original or fake. In the original description of MANA I, the pair of parties additionally need to agree on the success of the protocol with the help of some human interactions. Since this is not important with respect to security analysis, we ignore the step in our description of the protocol.

¹⁴We can replace the strong empirical channel with a bounded delay empirical one (\xrightarrow{t}_{BE}), provided B checks that he has received Message 1 before Message 2 could have been sent.

Binding $INFO_A$ (M words) directly to the SAS makes the protocol efficient because each node computes a single digest at a cost of M : much cheaper than a long output hash function in Balfanz et al. and a commitment scheme in Pasini-Vaudenay.¹⁵

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

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

Whilst the security proofs of this protocol given in [16, 18, 45] are largely correct, what these authors have not discovered is that the bitlength they choose for the key (which happens to be equal to b in this case) is too short compared to the digest output and the authenticated information $INFO_A$. As a consequence, it is impossible to construct a digest, MAC or check-value function such that the probability of any one-shot attack on the protocol is upper bounded by 2^{-b} . Since the weakness has a very profound impact on all other uses of the digest function, we are going to analyse the (off-line) computation complexity and its related probability of a successful one-shot attack on this protocol. We then deduce a longer key is required in order for the digest function to meet its specification.

We term b and r the bitlengths 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 he can expect to find two different $INFO_A$ and $INFO'_A$, such that for all $k \in \{k_1, \dots, k_c\}$ ¹⁶ we have:

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

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

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

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.

¹⁵This 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 $MW = 5M$.

¹⁶It might be clearer if we define $H_{\{k_1, \dots, k_c\}}(X) = \text{digest}(k_1, X) \parallel \dots \parallel \text{digest}(k_c, X)$, and if digest is an ideal digest function, then so is the function H w.r.t its $c \times b$ output-bits. As there is no limit on the bitlength of the input X , it normally takes $2^{cb/2}$ computation steps to search for a collision, due to the birthday paradox.

After sending the first 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 $\frac{2^r - c}{2^r}$, k is not in this set and the attack is only successful with a probability of (presumably) $2^{-b} \times \frac{2^r - c}{2^r}$.

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

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

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

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

An interesting question arises as we want to know how large the bitlength of the key should be in relation to a fixed amount of information we want to authenticate and the output bitlength of the digest. Since this question is not within the scope of this paper, we point readers to one of our papers [41] where we successfully derive a new combinatorial bound for an *almost universal* family of hash functions.¹⁷

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

2.3 Improved version of (V-)MANA I

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

In contrast to 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 in Section 1.3.2. In addition, we can weaken the assumption that empirical messages' transmission is instantaneous to being of bounded delay as follows.

Improved version of V-MANA I (direct binding) [42]	
1.	$A \rightarrow_N B : M, \text{longhash}(k)$
2.	$A \rightarrow_{BE}^t B : \text{digest}(k, M)$
3.	$A \rightarrow_N B : k$

¹⁷There is a known theoretical bound of Stinson [59] on the bitlength of the key that can guarantee the digest meets its specification: $\text{bitlength}(k) \geq \text{bitlength}(INFO_A) - b$. We should remark that the bound can be met except for an infinitesimal tolerance in the digest collision probability ϵ for very much smaller lengths than this, see [41]. However, it always has to be significantly longer than b in practice.

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

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

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

Interestingly, we can replace the bounded delay empirical channel and the need to wait by a simple acknowledgement from B to A . The resulting protocol turns out to be the pairwise (one-way authentication) version of HCBK protocol, described in Section 4 and [48].

Improved version of MANA I (direct binding) [48, 39, 40, 42]	
1a.	$A \rightarrow_N B : M, \text{longhash}(k)$
1b.	$B \rightarrow_E A : 1\text{-bit committed signal}$
2.	$A \rightarrow_E B : \text{digest}(k, M)$
3.	$A \rightarrow_N B : k$

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

Since the SAS in these schemes are functionally dependent on the authentic information M , we term these as the *direct binding* version of Improved (V-)MANA I, i.e. direct information binding strategy will be formally defined in Section 3. However, the computation cost is slightly increased to $W^2 + M = 25 + M$, due to the extra longhash required in the first message.

Readers who are interested in the formal security proof as well as variants using indirect binding and Diffie-Hellman can find them in Appendix C or [41].

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.¹⁸

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.

¹⁸Perhaps this could be worked around by having a logged recording mechanism for the empirical messages as part of the receiver's system.

What we will discover is the significance of the idea of *joint commitment without knowledge*, introduced by us in Section 1.3.1, in providing the same level of security for all protocols presented in this section, i.e. the probability of a successful one-shot attack¹⁹ is upper bounded by 2^{-b} , where b is the bitlength of the SASs. For this reason, in all pairwise schemes (except Hoepman and Wong-Stajano in Section 3.1), the value of the unique SAS is jointly committed to by both protocol participants. This therefore leads us to introduce the two following information binding strategies that help us achieve (joint) commitment without knowledge as well as classifying the many protocols considered in this survey.

- **Indirect information binding:** The SAS, jointly committed by every node, is *independent* of the information *INFOS* parties want to authenticate. Typically, the SAS is the exclusive-or of random nonces individually committed to by every party as well as being cryptographically bounded to *INFOS* at the begin of each protocol run.
- **Direct information binding:** The SAS, jointly committed by every node, is *dependent* on the information *INFOS* parties want to authenticate. Typically, the SAS is the output of some function applying to *INFOS* in combination with secret keys individually committed to by every party at the begin of a run. This is also closely related to two protocol design principles **P1** and **P2** introduced later in this survey.

When we study the two strategies in Sections 3.2 and 3.3, we find that direct binding has a clear advantage in efficiency over indirect one. This arises from the potential to use a short output digest function to process the large *INFOS* as opposed to a conventional long output cryptographic hash. The advantages will be demonstrated clearly when the cost of all schemes are gathered in three tables in Section 5.1.

3.1 Multiple empirical short authentication strings

We intend to describe two pairwise authentication protocols, the first by Hoepman [22, 23] and the second by Wong and Stajano [66, 67] in this subsection. In these schemes, parties manually compare or handle two different short authentication strings (SASs) each of $b = 16$ bits, so $2b = 32$ bits in all. We point out an important difference in how these two protocols process *INFOS*.

Hoepman [23] defines SASs as the outputs of a b -bit (short) hash function $hash_b()$, as mentioned in Section 1.3.2. In addition, the Diffie-Hellman tokens $g^{x_{A/B}}$ play the role of both $INFO_{A/B}$ and long fresh random nonces, and so must be unpredictable and fresh at each session.

¹⁹More information about the intruder's power and different types of attack can be found in Section 1.4.

Hoepman pairwise protocol, [23]	
1.	$A \rightarrow_N B : longhash(g^{x_A})$
1'.	$B \rightarrow_N A : longhash(g^{x_B})$
Where x_Y is a long random nonce of Y	
2.	$A \rightarrow_E B : hash_b(g^{x_A})$
2'.	$B \rightarrow_E A : hash_b(g^{x_B})$
3.	$A \rightarrow_N B : g^{x_A}$
3'.	$B \rightarrow_N A : g^{x_B}$
A and B verify the long and short hashes.	
A and B then share the key $k = g^{x_A x_B}$	
4.	$A \rightarrow_N B : longhash(g^{x_A x_B})$
4'.	$B \rightarrow_N A : longhash(g^{x_B x_A})$

This protocol offers a good security, i.e. the probability of a successful one-shot attack is bounded by 2^{-b} , despite the use of b -bit hashes can be explained through the idea of *joint commitment without knowledge*: both parties are jointly committed to $g^{x_A x_B}$ by publishing their shares of the commitment (i.e. $longhash(g^{x_{A/B}})$) in the first messages. It is therefore vital here that both parties must agree on when to finish inputting the first messages. Once the commitment phase is over, Messages 2, 2' and 3, 3' can be sent out in any order without compromising the security.²⁰ We will see an example of what goes wrong without (joint) *commitment without knowledge* at the start of Section 4.1, which discusses group protocols.

We assume that M is the word-length of the Diffie-Hellman tokens²¹ (g^{x_A} and g^{x_B}). Since Messages 4 provide shared secret validation (using $longhash()$ function in this case), they can be neglected in our cost analysis. As a result, each node has to compute 2 longhashes and 2 shorthashes. Using our cost model of computing hash functions given in Section 1.5.2, the computation cost of Hoepman is of order $2(WM + M) = 12M$, where W and 1 are the output word-lengths of long and respectively short hashes.

Taking a different approach, Ford-Long Wong and Frank Stajano [66, 67] propose another scheme which does not use a short hash function, but does give the same security with an equal number of empirical bits. The simplification comes with an extra cost of more than doubling the input size of the $longhash()$ function used in the commitment phase. This is the consequence of the inclusion of short and long nonces (R_Y and K_Y) of b and $(B - b) = (160 - b)$ bits, respectively.

²⁰In [22], Hoepman introduced a modified (pairwise) version of the above scheme in which each party can receive multiple longhashes or commitments from unknown nodes at the very beginning of a run. But (s)he only pairs up with the one, who provides the matched single shorthash $hash_b(X)$ sent over the empirical channel in the second message. In this circumstance, A only sends out the shorthash iff he receives the 1-bit commitment empirical signal from B at the first place and vice versa. Furthermore, these acknowledgement signals must be transmitted over the empirical channel because they must not be blocked or delayed by the intruder. In this version, A does not need to know the identity of B during Messages 1, so Hoepman refers to it as the *anonymous* case. Whereas the protocol above applies to the *non-anonymous* case.

²¹As the Diffie-Hellman tokens are the only information parties want to authenticate, we can treat them as $INFO_{A/B}$, whose lengths are M words.

Wong-Stajano pairwise protocol, [66]	
1.	$A \rightarrow_N B : g^{x_A}$
1'.	$B \rightarrow_N A : g^{x_B}$
2.	$A \rightarrow_N B : \text{longhash}(A, g^{x_A}, g^{x_B}, R_A, K_A)$
2'.	$B \rightarrow_N A : \text{longhash}(B, g^{x_B}, g^{x_A}, R_B, K_B)$
	R_Y and K_Y are short and long random nonces of Y
3.	$A \rightarrow_E B : R_A$
3'.	$B \rightarrow_E A : R_B$
4.	$A \rightarrow_N B : K_A$
4'.	$B \rightarrow_N A : K_B$
	A and B verify the longhashes.

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

With respect to computation cost, while there is no short hash function, the two longhashes (with long inputs) that need to be computed at each node result in a significantly larger cost of $2W(2M + W) = 20M + 50$ 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} to the pair of random nonces (R_A, K_A) . This leads to the elimination of Messages 1 and 2, and indeed the same problem has been independently found and corrected by the inventors in their revised version of the paper, published in October 2007 [67]. However, they have not noticed that the Diffie-Hellman tokens (g^{x_A} and g^{x_B}) can play the dual role of the authentic information and fresh nonces if they are made unpredictable and fresh in each session. For this reason, we can further eliminate the need for long random nonces $K_{A/B}$ to simplify the protocol. A detailed description of our modified version of the protocol will be given in Section 3.2 and Footnote 24.

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

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

come.

3.2 Indirect binding

In *indirect* binding protocols, the SASs, jointly committed to and manually communicated by parties, are functionally independent of the information they want to authenticate. This is the idea we have briefly seen in Wong-Stajano, and it appears in many other schemes proposed in the literature [63, 44, 9, 66, 67, 4, 28, 29]. We will analyse these here. What distinguishes all of these from Wong-Stajano is a 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 binding some random nonces, which are instrumental in the computation of SASs, and *INFOS* together in a secure way. Thus, there is a tendency to use commitment schemes (described in Section 1.3.1) 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 [63] and Čagalj et al. [9] to establish one- and two-way authentication via one- and two-way empirical channels in a peer-to-peer network. We will extend their schemes into group versions in Section 4.2.

The following is the description of a pairwise scheme, invented by Vaudenay [63], 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, [63]	
1.	$A \longrightarrow_N B : INFO_A, c$ Where $c \parallel d = commit(INFO_A, R_A)$, R_A is a short random nonce of A .
2.	$B \longrightarrow_N A : R_B$
3.	$A \longrightarrow_N B : d$ B computes $R_A = open(INFO_A, c, d)$
4.	$A \longrightarrow_{WE} B : R_A \oplus R_B$ B verifies the correctness of $R_A \oplus R_B$

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

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

The above analysis applies to a one-shot attack. If we consider a q -shot attack then we need to take into account that the unique SAS of this protocol is transmitted over the weak empirical channel, and therefore can be stalled and delayed in later protocol runs. As a consequence, with q concurrent runs of A and B , the chance of a successful attack is bounded by $q^2/2^b$ as pointed out by Vaudenay [63]. We however argue that this security analysis is only valid with a small value of q because humans are highly sensitive to delays, i.e. they will quickly become aware that an attack is taking place, and so stop any attempt of running the protocol again as pointed out in Section 1.4. In contrast, if we replace \rightarrow_{WE} with \rightarrow_E then a SAS cannot be delayed from one to later runs, and so to have a fair chance of a successful attack, the intruder needs to run 2^b concurrent runs of (perhaps different) pair of parties: an 2^b -shot attack.

There is a single commitment used (committed by A , and decommitted or opened by B), hence the computing cost at each node is of order $MW = 5M$. Here, both a commitment c and a decommitment d have the same length of W words.²²

Although Vaudenay’s scheme halves the amount empirical communication relative to Hoepman and Wong-Stajano, it only provides one-way authentication, representing one role of pairwise schemes in this paper. In practice, we often want to achieve more than this, and that is why we now consider another protocol performing message authentication in both directions at the same time. Suppose B has some $INFO_B$ and wants to have it authenticated to A , then the natural way to tackle this problem is to make B commit to its information as done by party A . This idea, proposed by Vaudenay in Appendix A of [63], fortunately makes the protocol structure completely symmetrical. It is essentially the same as another protocol termed DH-SC, invented by Čagalj, Čapkun and Hubaux [9].

Čagalj-Čapkun-Hubaux two-way authentication protocol, [9]	
1.	$A \rightarrow_N B : INFO_A, c_A$
1'.	$B \rightarrow_N A : INFO_B, c_B$
	Where $c_Y \parallel d_Y = \text{commit}(Y, INFO_Y, R_Y)$, R_Y is a short random nonce of Y .
2.	$A \rightarrow_N B : d_A$
2'.	$B \rightarrow_N A : d_B$
	Y' computes $R_Y = \text{open}(Y, INFO_Y, c_Y, d_Y)$
3.	$A \leftrightarrow_E B : R_A \oplus R_B$

Both of the above protocols use the *joint commitment without knowledge* principle to pre-commit two parties to the XOR of some random short secrets or nonces. This is achieved by parties outputting their shares of the commitment to each other in the first messages.

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 has not to acquire $INFO_B$ at the time of computing the commitment. Secondly, the order of releasing the SAS and the decommitments has been reversed relative to Wong-Stajano.²³ As a consequence, parties only need to manually

²²The cost of XORing two short random nonces R_A and R_B is small compared to implementing the commitment scheme, and therefore is neglected here.

²³The SAS and the decommitments here correspond to the two different short nonces and the long nonces

compare a single SAS, which is the XOR of short nonces R_A and R_B implicitly derived from the decommitments.

It is interesting to note that the same technique can be used to improve the human and processing cost of Wong-Stajano.²⁴

Regarding computation cost, the two commitments would double the cost of Vaudenay to an order of $2WM = 10M$. On the other hand, if we quantify the cost relative to the amount of information authenticated then Vaudenay and Čagalj-Čapkun-Hubaux will be equal to each other. The result 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, as described in Section 4.1.

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

3.2.2 Hybrid protocol

Prior to discussing direct binding protocols, we describe an important scheme bridging the gap between the two strategies both in terms of protocol structure and computational cost. Pasini and Vaudenay [44] propose a two-way authentication protocol using the idea of Vaudenay's one-way scheme in Section 3.2.1. They make use of a truncated hash function that we have improved to a digest and a symmetric empirical channel.

Pasini-Vaudenay two-way authentication protocol, [44]*	
1.	$A \rightarrow_N B : INFO_A, c$ Where $c \parallel d = commit(INFO_A, k_A)$ k_A is a long random nonce of A
2.	$B \rightarrow_N A : INFO_B, R_B$ Where R_B is a b -bit random nonce of B .
3.	$A \rightarrow_N B : d$ B computes $k_A = open(INFO_A, c, d)$
4.	$A \leftarrow_E B : R_B \oplus digest(k_A, INFO_B)$

Though the SAS = $R_B \oplus digest(k_A, INFO_B)$ depends functionally on $INFO_B$, it is probabilistically independent of $INFO_A$. This observation makes the scheme stand as a hybrid

in Wong-Stajano, respectively.

²⁴ The idea of eliminating the first two messages carrying $g^{x_A/B}$, removing the long random nonces as well as reducing the number of different SASs to a single one of b bits in Wong-Stajano can be demonstrated by our revised scheme. The scheme achieves the same level of security as Wong-Stajano in Section 3.1, i.e. the probability of a successful one-shot attack is bounded by 2^{-b} .

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

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 ($20M + 50$).

of direct- and indirect-binding protocols. Interestingly, the hybrid strategy is also reflected by the differences in the bitlengths and the functionality of the two random nonces: R_B and k_A . R_B is protected by the structure of the protocol from guessing attacks and so can be short, whereas k_A is not and so has to be long; the two influence the final empirical string in different ways.

There is no need to use a commitment scheme to bind $INFO_B$ to R_B , so each node needs to compute a digest and either a commitment or a decommitment. The processing cost drops to $WM + M = 6M$, thanks to the efficiency of a digest,²⁵ which is significantly cheaper than Čagalj-Čapkun-Hubaux ($10M$).

3.3 Direct binding pairwise protocols

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

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

Our first task is to formalise the direct binding approach as the following principle **P1**, introduced by us in [39, 40].

- P1** [39, 40] Make all the parties, who are intended to be part of a protocol run, empirically agree a short-output hash or *digest* of all the information parties want to authenticate.

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, unless there is a hash or digest anomaly – different nodes in the group computing the same hash value or digest from different antecedents – clearly all the parties agree on all the data transmitted during the protocol.

In this section, a number of protocols providing mutual authentication are presented in an ascending order of computation efficiency and simplicity. In addition to the common use of the direct binding strategy to obtain *joint commitment without knowledge*, they are all asymmetrical in structure, which is similar to the one-way authentication protocol of Vaudenay [63] in Section 3.2.1.

Unlike indirect binding schemes, parties need to generate fresh *long* random nonces or sub-keys, which have enough entropy to prevent them from being subject to a combinatorial search.²⁶ On the other hand, the security analysis of the direct binding schemes is similar to

²⁵There should have been no improvement ($WM + WM = 10M$), if we had not switched to the use of digest.

²⁶It is possible to regard these long fresh sub-keys as the extended versions of short random nonces, used in commitment schemes in indirect binding protocols.

indirect binding ones provided the digest function is ideal, as specified in Section 1.3.2. In every case, the protocols have the same security (i.e. the probability of a successful one-shot attack is bounded by 2^{-b}) because nodes (and hence the intruder) do not know the final value of the digest key k until they are committed to the final value of the digest, truncated hash or universal hash output, thanks to the *joint commitment without knowledge* idea that provides a common theme to this family of protocols.

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

The following protocol is taken from Bluetooth whitepaper [4], where k_A and k_B are long fresh sub-keys generated by A and B .

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

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

The inefficiency in using a truncated hash function²⁷ will increase the computation cost of the above “Bluetooth 2” to $W(2M + W) + 2W(M + W) = 20M + 75$ as opposed to $W(2M + W) + 2M = 12M + 25$ should we employ a digest and XOR to combine sub-keys. Unfortunately, the latter will still be more expensive than the two following schemes, as it is redundant to bind *INFOS* and sub-keys together by using both longhash function in Message 2 and in the SAS. Either of them would be sufficient for the obtained security.

Removing this unnecessary binding in Message 2 of Bluetooth 2 can increase its computational efficiency as well as simplicity (eliminating Messages 1 and 1'), since B does not need to know $INFO_A$ (and *INFOS*) at the point when he is committed to k_B . This is what was proposed in [28, 29] by Laur and Nyberg:

Laur-Nyberg pairwise protocol, [28, 29]	
1.	$A \rightarrow_N B : INFO_A, c$ Where $c \parallel d = commit(k_A)$
2.	$B \rightarrow_N A : INFO_B, k_B$
3.	$A \rightarrow_N B : d$ B computes $k_A = open(c, d)$
4.	$A \leftrightarrow_E B : h_{k^*}(INFO_S)$ Here $k^* = g(k_A, k_B)$

Here $h_{k^*}()$ is a universal hash function [59] of the appropriate length, i.e. b -bit in this case, whose specification is closely related to a digest as discussed in Section 1.3.2. The impact

²⁷ $trunc_b()$ takes the first b bits of its input.

of removing redundancy can be seen in the decline of the computation cost of this protocol: $W^2 + 2WM = 25 + 10M$.²⁸

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

We observe that it would be equally satisfactory to use the combination of $k_A \oplus k_B$ and a digest function in place of $h_{k^*}(INFOS)$, resulting in an improvement of computational cost ($W^2 + 2M = 25 + 2M$).

After this transformation and a replacement of a commit scheme with a longhash, the protocol becomes similar to the following. This was discovered independently by the author in the summer of 2006 when we combined ideas used in our SHCBK protocol (see Section 4.1) and Vaudenay's protocols (see Section 3.2.1).

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

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

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

The main difference between this and the previous schemes is that there is no $INFO_{A/B}$ because the Diffie-Hellman tokens 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.²⁹ Also because of this, the digest function (2-input function) can be replaced by a single input short hash function $hash_b()$; though the combination of this and the exponentiation of Diffie-Hellman needs to satisfy a specification similar to that of the digest function and the randomising effect of XOR in combining k_A 's.

In comparison with Hoepman, this requires a single SAS, halving the human work. Similar to previous protocols, the computation of one longhash and one short hash results in a cost of $WM + M = 6M$: exactly a half of Hoepman ($12M$).

²⁸We choose to ignore the cost of computing $g()$ to combine sub-keys in this calculation, since it is negligible relative to the computation of a universal hash, which involves applying a cryptographic hash to the $2M$ -word input message $INFOS$ in the first place, as specified by Laur and Nyberg in [28, 29], which results in a cost of $2WM = 10M$.

²⁹It is possible but not necessary to replace $g^{x_A} \oplus g^{x_B}$ with $g^{x_A x_B}$ in this scheme.

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

Improved Hoepman' (One-way empirical channels)	
1.A	$\rightarrow_N B : longhash(A, g^{x_A})$
2.B	$\rightarrow_N A : g^{x_B}$
3.A	$\rightarrow_N B : g^{x_A}$
4.A	$\rightarrow_E B : hash_b(g^{x_A} \oplus g^{x_B})$
5.B	$\rightarrow_E A : 1\text{-bit committed signal}$

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

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

Wong and Stajano [67] give a protocol using 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), [67]	
1.A	$\rightarrow_N B : g^{x_A}$
2.B	$\rightarrow_N A : B, g^{x_B}, MAC_{K_B}(B, g^{x_A}, g^{x_B}, R_B)$ R_B and K_B are short and long random nonces of B
3.A	$\rightarrow_E B : 1\text{-bit committed signal}$
4.B	$\rightarrow_E A : R_B$
5.B	$\rightarrow_N A : K_B$
6.A	$\rightarrow_E B : 1\text{-bit committed signal}$

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

It seems unlikely that the computation cost of the cheapest of these protocols can be reduced much further. It also seems clear that some sort of cryptographic binding of *INFOS* to the empirical message is necessary, and our assumed model of the digest function appears to be a lower bound on that as we want to bind the *whole* of *INFOS*. Similarly, it is clear that for the *joint commitment without knowledge* approach to work, we need to have a token randomising the final digest and being committed to before any node knows it. This has to be done with strong cryptography, and the hash used in, for example, Laur-Nyberg appears to be as efficient as possible at doing this.

In the above protocols, the use of a strong cryptographic primitive, such as a hash function or a commitment scheme, to protect the secrecy of long random nonces or keys, and in the mean time a much shorter (and therefore weaker) function to digest large *INFOS* clearly aims to block combinatorial search and guess attacks separately.

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

Work in this area seems so far to have been restricted to the author's group (including Roscoe, Creese, Goldsmith and Zakiuddin), and more recently Valkonen et al. This has resulted in several group protocols presented in [39, 40, 12, 13, 14, 15, 62]. In [39, 40], we identified the main challenges of bootstrapping group security in pervasive computing, and these can be explained as follows.

There is a slightly grey area for protocols building groups of more than 2. Should we or should we not be content if the presence of a corrupt party in a group means that communications between other trustworthy members of the group are themselves compromised? In some of the circumstances, where we may wish to use *ad hoc* group formation protocols, it would be much better if the protocols were tolerant of corrupt members. We will, therefore, be careful about our assumptions on this front. It is obvious any key agreement protocol is at least partially compromised by the presence of a corrupt participant. However, protocols which merely authenticate public-key-like information are not automatically compromised: they could be said to be establishing a *local PKI*. As we will see, this will be successfully resolved by using the idea of (joint) *commitment without knowledge*.

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

We have already seen that the direct information binding strategy is significantly more efficient than indirect one for pairwise protocols (i.e. both of these information binding strategies are instrumental to achieving *commitment without knowledge* as pointed out in Section 3). In this section, we will see the same is true for group protocols, namely HCBK and SHCBK versus the group version of the indirect binding pairwise protocol of Čagalj-Čapkun-Hubaux. Interestingly, it is possible to further improve the efficiency in direct group protocols with a trade-off between human and computational costs, or by making use of protocol structure of the one-way scheme of Vaudenay [63].

4.1 Some existing direct binding group protocols

The following protocol introduced by Creese et al. [14] is probably the very first group authentication protocol in the area of pervasive computing. Here, $\forall A$ means that a message is sent to, or received by, all parties in the group \mathbf{G} attempting to achieve a secure link

between their laptops or PDAs. Pk_A stands for an uncertificated public key that A wants to authenticate to the group, whereas T_A and N_A are A 's fresh nonces. The superscript 'all Messages 2^d' represents the concatenation of all the decrypted content of Messages 2 in alphabetical order, for example. In addition, Messages 4 do not add any extra security to the scheme, its presence aims to provide a confirmation of the shared secret information.³⁰

Group protocol of Creese et al. [14]	
1.	$\forall A \rightarrow_N \forall A' : A, Pk_A, T_A$
2.	$\forall A \rightarrow_N \forall A' : \{\text{all Messages 1}, N_A\}_{Pk_{A'}}$
3a.	A displays : $hash_b(\{\text{all Messages 2}^d\})$, number of processes
3b.	$\forall A \rightarrow_E \forall A' : \text{users compare hashes and check numbers}$
4.	$\forall A \rightarrow_N \forall A' : longhash(\{\text{all Messages 2}^d\})$

The protocol is shown in [48, 39, 40] by Roscoe to be vulnerable to a man-in-the-middle and one-shot attack, related to the birthday paradox. The flaw arises from the short output of the shorthash function $hash_b()$ 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 [48, 39, 40]. In spite of the attack, the protocol invented in 2003 introduced implicitly the principle **P1** in Section 3.3, which contributes to the optimisation of not only empirical work but also computational cost.

In summer 2005 [48], Roscoe corrected this flaw by introducing a trustworthy leader L , who is responsible for generating a fresh key k_L of order 160 bits that is inputted into the digest function used in this scheme. The following is a slightly simplified version of this protocol, introduced in [39, 40]. Here, S represents a typical slave node, and A a typical node (either L or S). $init(L, A)$ is *true* if $L = A$ and *false* otherwise.

Hash Commitment Before Knowledge HCBK protocol, [48, 39, 40]	
0.	$L \rightarrow_N \forall S : L$
1.	$\forall A \rightarrow_N \forall A' : (A, INFO_A)$
2a.	$L \rightarrow_N \forall S : longhash(k_L)$
2b.	$\forall S \rightarrow_E L : \text{committed}$
3.	$L \rightarrow_N \forall S : k_L$
4.	$\forall A \rightarrow_E \forall A' : digest(k_L, INFOS), init(L, A)$

In this scheme, the parties have to agree on the b -bit digest of $INFOS$ and the leader's key k_L over the empirical channel. In addition, Message 2b has all the slaves communicate to L that they have received Message 2a, and are *committed* to their final digest value (though none of them know it yet). Thus the 1-bit commitment signal must be sent over the unforgeable empirical channel that cannot be blocked. We will see shortly this represents one side of an interesting trade-off. In term of computation cost, each node has to compute a single cryptographic hash of key k_L and a b -bit digest value of $INFOS$, resulting in a cost of order: $W^2 + NM = 25 + NM$, thanks to the use of the cost model given in Section 1.5.2.

The protocol is termed HCBK standing for *Hash Commitment Before Knowledge*, and its security relies on the trustworthiness of the leader L who generates the single digest

³⁰Messages 4 in this protocol are similar in purpose to Messages 4 in Hoepman (Section 3.1).

key, and consequently has control over the final digest value.³¹ 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 [67] 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 binding 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 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 [39, 40] we identified the following second principle, derived from the leader’s role in HCBK and the *direct binding* strategy of commitment without knowledge, 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 (joint) *commitment without 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), [39, 40]	
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.³² This protocol also eliminates the one-bit commitment signals from the slaves to L .

³¹The readers can find the full security analysis of HCBK in [39, 40].

³²We 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 leader, $\text{longhash}(k_L)$.

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 use of XOR to combine different sub-keys in SHCBK protocol was shown to be secure in [40]. 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 [40], 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.

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.

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 leader then we will have a one-way authentication group protocol: all the slaves are authenticated to the leader.

4.2 Indirect binding group protocol

We have claimed that the direct binding approach (HCBK and SHCBK protocols) remains more efficient in group protocols than indirect binding as it was in pairwise ones. The argument is true because it is more efficient to have the SAS created from the presumed large *INFOS* rather than it is to have each *INFO_A* bounded to random nonces by full-power cryptography to resist combinatorial search. In order to illustrate this advantage, we will generalise the (symmetrical) indirect binding pairwise scheme of Čagalj, Čapkun and Hubaux [9] in Section 3.2.1 into a group protocol. The level of security achieved by this scheme is the same as SHCBK: (1) tolerant of corrupt parties; and (2) the probability of successful one-shot attack is upper bounded by 2^{-b} , here b is still the bitlength of the single SAS transmitted over the empirical channel.

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

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

1. $I \longrightarrow_N A : INFO_I, c_I$
 $I \longrightarrow_N B : INFO'_I, c'_I$
Here :
 $c_I \parallel d_I = commit(I, INFO_I, R_I)$,
 $c'_I \parallel d'_I = commit(I, INFO'_I, R_I)$
 $A \longleftrightarrow_N B : INFO_{A/B}, c_{A/B}$
2. $I \longrightarrow_N A : d_I$
 $I \longrightarrow_N B : d'_I$
 $A \longleftrightarrow_N B : d_{A/B}$
3. $\forall A \longrightarrow_E \forall A' : R_I \oplus R_A \oplus R_B$

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

4.3 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 in SHCBK without compromising its security, i.e. being tolerant of corrupt parties and the probability of a successful one-shot attack is bounded by 2^{-b} . A small improvement turns out to be possible if we make use of a technique used in Vaudenay's one-way scheme and direct binding pairwise protocols in Sections 3.2.1 and 3.3. On the one hand, this can slightly reduce the number of commitments or longhashes at each node. On the other hand, it makes the schemes asymmetrical in structure. This will be explained as follows.

Let us assume there are $(N - 1)$ leaders L 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 node left is the unique slave S , who transmits its fresh sub-key k_S to other nodes after receiving longhashes from every leader. Below, A is a typical node which is either S or L .

De-symmetrised SHCBK protocol, [62]*			
0.	S	$\rightarrow_N \forall L$	$: S$
1.	$\forall L$	$\rightarrow_N \forall A$	$: INFO_L, longhash(L, k_L)$
2.	S	$\rightarrow_N \forall L$	$: INFO_S, k_S$
3.	$\forall L$	$\rightarrow_N \forall A$	$: k_L$
4.	$\forall A$	$\rightarrow_E \forall A'$	$: digest(k^*, INFOS)$
where k^* is the XOR of all the k_A 's for $A \in \mathbf{G}$			

We discovered this shortly after SHCBK. It was also independently invented by Valkonen, Asokan and Nyberg [62], who were not aware of SHCBK and 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, the fact that it is the only one treated in this way guarantees that it will not be manipulated by the intruder. This is as resistant to corrupt participants as 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 and making the protocol asymmetrical, the total number of longhashes per node declines to $N - 1$, which corresponds to a processing cost of $(N - 1)W^2 + NM = 25N + NM - 25$. This is cheaper than SHCBK by $W^2 = 25$ units per node, though we suspect that the asymmetry introduced into the communication regime will in practice mean that it is no better: nodes will spend more time waiting. Nevertheless, it illustrates the possibility of further improving the computation efficiency by careful analysis.

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

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

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

Hybrid HCBK <i>New</i>		
0.	T	$\longrightarrow_N \forall A : SL$
1.	$\forall A$	$\longrightarrow_N \forall A' : A, INFO_A$
2a.	$\forall L$	$\longrightarrow_N \forall A : longhash(L, k_L)$
2b.	$\forall S$	$\longrightarrow_E \forall L : 1\text{-bit committed signal}$
3.	$\forall L$	$\longrightarrow_N \forall A : k_L$
4.	$\forall A$	$\longrightarrow_E \forall A' : digest(k^*, INFOS), leader(SL, A)$
Where k^* is the XOR of all the k_L 's for $L \in SL$		

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

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

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

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

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

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

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. In those protocols that require the extra confirmation message over the empirical channel (MANA I, Wong-Stajano, HCBK, HHCBK etc) we write " $b + 1$ " as the amount of empirical effort.

In these tables we have used the simple cost model of hash and digest functions described in Section 1.5.2: the cost is proportional to the product of the length of the information being digested and the width of the output.

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

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

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

It is also necessary to point that since most direct binding protocols invented by other authors to date do not use a digest 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 ([59] 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.³³ The larger M (the length of *INFOS*) is, the more accurate this effect.³⁴ This is likely to be the case whenever

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

³³In Section 1.5.2 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).

³⁴This 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.

Protocol	Binding	Human work (bit)	Computation cost
Balfanz et al.	Direct	$B=160 (\rightarrow_{WE})$	$WM = 5M$
Pasini-Vaudenay	Indirect	$B/2=80(\rightarrow_{WE})$	$\frac{MW}{2} + \frac{W^2}{4} = 2.5M + 6.25$
MANA I (CBC-MAC)	Direct	$2b+1=33 (\rightarrow_E)$	$WM = 5M$
V-MANA I (<i>digest</i>)	Direct	$2b = 32 (\rightarrow_{SE})$	M
Improved MANA I	Direct	$b + 1 = 17 (\rightarrow_E^t)$	$M + W^2 = M + 25$
Improved MANA I	Indirect	$b + 1 = 17 (\rightarrow_E^t)$	$W(M + W) = 5M + 25$
Improved MANA I	Direct(D-H)	$b + 1 = 17 (\rightarrow_E^t)$	$W^2 + W = 30$
Improved V-MANA I	Direct	$b = 16 (\rightarrow_{BE}^t)$	$M + W^2 = M + 25$
Improved V-MANA I	Indirect	$b = 16 (\rightarrow_{BE}^t)$	$W(M + W) = 5M + 25$
Improved V-MANA I	Direct(D-H)	$b = 16 (\rightarrow_{BE}^t)$	$W^2 + W = 30$

Table 2: One-way authentication protocols

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

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

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

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

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.

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

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

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

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 without knowledge* un-

derlies 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 [62].

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 [3] 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) [38], auction protocols (anonymous physical broadcast channel) [56] and e-cash [11] 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 [33, 35, 36, 52, 61] recently.

Acknowledgements

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

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

References

- [1] See: http://en.wikipedia.org/wiki/Avalanche_effect
- [2] See: www.bluetooth.com/developer/specification/specification.asp
- [3] See: <http://www.prismmodelchecker.org/>
- [4] *Simple Pairing White Paper*. See: www.bluetooth.com/NR/rdonlyres/0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/SimplePairing_WP_V10r00.pdf
- [5] 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.
- [6] M. Bellare and P. Rogaway. *Entity Authentication and Key Distribution*. Advances in Cryptology - Crypto 1993, LNCS vol. 773, Springer-Verlag, pp. 232-249, 1993.
- [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. Blum. *Coin Flipping by Telephone*. Advances in Cryptology - Crypto '81, pp. 11-15, 1981. Or see: http://en.wikipedia.org/wiki/Commitment_scheme
- [9] 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.
- [10] J.L. Carter and M.N. Wegman. *Universal Classes of Hash Functions*. Journal of Computer and System Sciences, 18 (1979), pp. 143-154.
- [11] D. Chaum. *Secret-ballot receipts: True voter-verifiable elections*. Security and Privacy Magazine, IEEE, Jan.-Feb. 2004. Volume: 2, Issue: 1, pp. 38-47.
- [12] 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 the second International Conference on Security in Pervasive Computing (*SPC'05*), vol. 3450, LNCS, Boppard, Germany, April 2005. Springer.
- [13] 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.
- [14] 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.

- [15] 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.
- [16] C. Gehrman, C. Mitchell, and K. Nyberg. *Manual Authentication for Wireless Devices*. RSA Cryptobytes, vol. 7, no. 1, pp. 29-37, 2004.
- [17] C. Gehrman and K. Nyberg. *Security in personal area networks*. In C. J. Mitchell, editor, Security for Mobility, pp. 191-230. IEE, London, 2004.
- [18] International Organisation for Standardisation, Geneva, Switzerland. *ISO/IEC 1st CD 97986, Information technology—Security techniques—Entity authentication—Part 6: Mechanisms using manual data transfer*, December 2003.
- [19] S.W. Golomb. *Shift Register Sequences*. ISBN: 0894120484, Aegean Park Press, 1981.
- [20] 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).
- [21] J. Hastad, R. Impagliazzo, L.A. Levin, and M. Luby. *A Pseudorandom Generator from any One-way Function*. SIAM J. Comput., 1999.
- [22] 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.
- [23] J.-H. Hoepman. *Ephemeral Pairing Problem*. In 8th Int. Conf. Fin. Crypt., LNCS 3110, Springer, pp. 212-226.
- [24] M. Jakobsson and S. Wetzel. *Security Weaknesses in Bluetooth*. CT-RSA 2001, LNCS vol. 2020, Springer-Verlag, pp. 176-191, 2001.
- [25] 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.
- [26] H. Krawczyk. *LFSR-based Hashing and Authentication*. CRYPTO 1994, LNCS vol. 839, pp. 129-139.
- [27] H. Krawczyk. *New Hash Functions For Message Authentication*. EUROCRYPT 1995, LNCS vol. 921, pp. 301-310.
- [28] S. Laur and K. Nyberg. *Efficient Mutual Data Authentication Using Manually Authenticated Strings*. Volume 4301 on LNCS, pages 90-107, Dec 2006. Springer.

- [29] S. Laur, N. Asokan and K. Nyberg. *Efficient mutual data authentication using manually authenticated strings: Extended version*. Cryptology ePrint Archive, Report 2005/424, 2006.
- [30] A.Y. Lindell *Comparison-Based Key Exchange and the Security of the Numeric Comparison Mode in Bluetooth v2.1*. To appear in RSA Conference 2009.
- [31] 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.
- [32] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton Computer Science Notes. Princeton University Press. January, 1996
- [33] A. Madhavapeddy, D. Scott, R. Sharp, and E. Upton. *Using Camera Phones to Enhance Human-Computer Interaction*. Proceedings of Ubiquitous Computing (UbiComp 2004), 2004.
- [34] 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.
- [35] R. Mayrhofer and M. Welch. *A Human-Verifiable Authentication Protocol Using Visible Laser Light*. Availability, Reliability and Security. ARES 2007.
- [36] 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.
- [37] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. *Handbook of Applied Cryptography*. ISBN: 0-8493-8523-7.
- [38] T. Moran and M. Naor. *Polling with Physical Envelopes: A Rigorous Analysis of a Human-Centric Protocol*. EUROCRYPT 2006.
- [39] 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.
- [40] L.H. Nguyen and A.W. Roscoe. *Authenticating ad hoc networks by comparison of short digests*. Information and Computation 206 (2008), 250-271. Special Issue of Information and Computation on Computer Security: Foundations and Automated Reasoning 2006.
- [41] L.H. Nguyen and A.W. Roscoe. *New combinatorial bounds for universal families of hash functions*. Submitted to EUROCRYPT'09.
- [42] L.H. Nguyen and A.W. Roscoe. *Separating two roles of hashing in one-way message authentication*. Proceedings of FCS-ARSPA-WITS 2008.

- [43] L.H. Nguyen and A.W. Roscoe. *Efficient digest function based on Toeplitz matrix and integer multiplication*, in preparation.
- [44] 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.
- [45] 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.
- [46] R. Pass. *On Deniability in the Common Reference String and Random Oracle Model*. CRYPTO '03, LNCS vol.2729, pp.316–337, Springer 2003.
- [47] T. Peyrin and S. Vaudenay. *The Pairing Problem with User Interaction*. SEC 2005, pp. 251-266.
- [48] A.W. Roscoe. *Human-centred computer security*. See: <http://web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications/113.pdf>, 2005.
- [49] A.W. Roscoe and L.H. Nguyen. *Security in computing networks*. Published by the World Intellectual Property Organization (WIPO). Publication Number: WO/2007/052045. Publication date: 10.05.2007. International Application No.: PCT/GB2006/004113. See: <http://www.wipo.int/pctdb/en/wo.jsp?wo=2007052045&IA=W02007052045&DISPLAY=STATUS>
- [50] A.W. Roscoe, B. Chen, and L.H. Nguyen. *Improvements in communications security*. International Patent Application No. PCT/GB07/004963, published by the World Intellectual Property Organization (WIPO), publication number: WO/2008/078101, publication date: 03.07.2008.
- [51] A.W. Roscoe and L.H. Nguyen. *Improvements related to the authentication of messages*. Priority patent application number 0811210.4, filed on 18 June 2008.
- [52] 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.
- [53] B. Schneier. *Applied Cryptography*. John Wiley and Sons, 1996. ISBN 0-471-11709-9. Or see: http://en.wikipedia.org/wiki/Cryptographic_hash_function
- [54] N. Smart. *Cryptography, An Introduction*. ISBN 0 077 09987 7 (PB). Or see: http://en.wikipedia.org/wiki/Chosenplaintext_attack
- [55] 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.

- [56] 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.
- [57] F. Stajano. *Security for Ubiquitous Computing*. Wiley, 2002.
- [58] 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.
- [59] D.R. Stinson. *Universal Hashing and Authentication Codes*. Advances in Cryptology - Crypto 1991, LNCS vol. 576, Springer-Verlag, pp. 74-85, 1992.
- [60] 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.
- [61] 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.
- [62] 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.
- [63] 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.
- [64] 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.
- [65] A.F. Webster and S.E. Tavares. *On the Design of S-Boxes*. CRYPTO 1985, LNCS vol. 218, Springer Verlag (1986), pp. 523-534.
- [66] 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.
- [67] 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 leader³⁵. 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 4.3 cannot be weakened further to have two slaves, even when all the nodes in the protocol are trustworthy.

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

1. α . $L \longrightarrow_N I(A, B) : INFO_L, longhash(k_L)$
2. α . $I(A) \longrightarrow_N L : k'_A$
 $I(B) \longrightarrow_N L : k'_B$
3. α . $L \longrightarrow_N I(A, B) : k_L$

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

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

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

1. β . $I(L) \longrightarrow_N A, B : INFO'_L, longhash(k_L)$
2. β . $A \longrightarrow_N I(B, L) : k_A$
 $B \longrightarrow_N I(A, L) : k_B$
 $I(A) \longrightarrow_N B : k_B \oplus k'_S$
 $I(B) \longrightarrow_N A : k_A \oplus k'_S$
3. β . $I(L) \longrightarrow_N A, B : k_L$

After the key k_L is revealed to A and B , all three nodes should be able to empirically agree

³⁵ A will receives $INFO'_B$ saying that B is the leader and vice versa.

on two equal digests that have different antecedents. In other words, the slaves accept $INFO'_L$ faked by the intruder.

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

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

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

C Security proofs of Improved MANA I

Adversarial model. Except for the authentic or empirical channels, we assume that the adversary has full control on the normal communication channel (\longrightarrow_N), for example: (s)he can block, delete, delay, redirect, fake and modify any information transmitted over this normal channel. In our security proofs, we will adopt the strong security model of Bellare and Rogaway [6] and its simplified version due to Vaudenay [63] which allow the adversary to have full control on which node a new instance of the protocol is launched.

In addition to this, following are the formal definitions of the terms: *one-shot* and *powerful* adversaries that will be used extensively in our security proofs.

1. **A powerful adversary** can launch multiple instances of participants (Alice and Bob in our case). As commonly known in the literature, the number of times that he can launch an instance of any participant is limited by a finite number, for example Q_A for Alice and Q_B for Bob. The time complexity of this adversary is bounded by a finite number say T . This is the kind of adversary we want to prove our protocols are resistant to in all security proofs presented here.
2. **A one-shot adversary** is a special case of the powerful adversary where the number of each participant's instances he can launch is at most once, in other words, $Q_A = Q_B = 1$.

The strategy of the proofs is to prove that our schemes are secure against a one-shot intruder in the first step, and then use Theorem 1 stated below to lift the one-shot intruder security proof to the powerful intruder's model.

The following theorem (and its proof) is a combined and slightly modified result of Lemma 6, introduced by Vaudenay [63], and Theorem 5 of Pasini and Vaudenay [45].

Theorem 1 [63, 45] *We consider three improved versions of Improved (V-)MANA I presented in Section 2.3, Appendixes C.2 and C.3 with claimant Alice (A) and verifier Bob (B). We consider powerful adversaries such that the number of instances of Alice (resp.*

Bob) is at most \mathcal{Q}_A (resp. \mathcal{Q}_B). If there exists a one-shot adversary against any of these protocols, bounded by time complexity t , has a probability of success smaller than p then the powerful adversary with time complexity $T = t \cdot \mathcal{Q}_A$ has a probability of success of $P \leq p \cdot \mathcal{Q}_A$.

Proof We say that an instance of A is compatible with an instance of B if B 's instance succeeded and received all messages in the right order, where one of which is transmitted over the (bounded delay) empirical channel from the corresponding A 's instance.

The number of possible compatible pairs of instances is therefore upper bounded by $\mathcal{Q}_A \mathcal{Q}_B$, which could be reduced to \mathcal{Q}_A because

- In improved versions of MANA I, the single SAS (i.e. digest or random nonce) transmitted over empirical channels by definition in Section 1.2 cannot be mistaken or delayed from one to another session.
- In improved versions of V-MANA I, B can always be offline. As a result, the intruder can simulate all instances of B and picks one who will make the attack succeed.

Thus the number of compatible pairs of instance is upper bounded by the number of times that A 's instance is launched by the powerful intruder.

When an attack is successful, there exists at least one compatible pair of instances of A and B . We note that these pairs of instances are probabilistically independent of one another, because the random nonce or key instrumental in the computation of each SAS is generated by the trustworthy party A in every instance, i.e. each SAS is a uniform distribution. Since the probability of success of each pair is limited to p in a time complexity t , we conclude that the powerful intruder bounded by a time complexity $T = t \cdot \mathcal{Q}_A$ is successful with a probability of $P \leq p \cdot \mathcal{Q}_A$.

C.1 Security proof of Improved (V-)MANA I (direct binding)

Improved version of V-MANA I (direct binding) [?]	
1. Alice \rightarrow_N	Bob : $INFO_A, longhash(k)$
2. Alice \rightarrow_{BE}^t	Bob : $digest(k, INFO_A)$
3. Alice \rightarrow_N	Bob : k

Followings are extra security properties of cryptographic primitives that we need to have in order to prove the direct binding version secure.

General specification of digest functions. Instead of using 2^{-b} , we will refer to ϵ_d as the digest collision probability to make our security proofs presented in this appendix more general. We will assume $\epsilon_d = 2^{-b} + \mu$, where μ is some negligible value relative to 2^{-b} .

Specification of digest functions [39, 40]
1. For any fixed m and digest y : $\Pr_k[digest(k, m) = y] \leq 2^{-b}$
2. For any fixed θ and any pair of different messages (m, m') : $\Pr_k[digest(k, m) = digest(k \oplus \theta, m')] \leq 2^{-b}$

Inversion-resistant hash functions. Inversion-resistance means that the Inversion-resistance game is hard. Assume that $longhash()$ is a (T_h, ϵ_h) inversion-resistant hash function then any adversary \mathcal{A} bounded by a time complexity T_h wins the Inversion-resistance game with probability at most ϵ_h . Since the bitlength of hash output is typically 160, we can assume that $\epsilon_h \ll 2^{-b}$. Here b is the length of digest, for example 16 or 20 bits.

Inversion-resistance game	
The challenger \mathcal{C} picks a random input x	
1.	$\mathcal{C} \rightarrow \mathcal{A} : longhash(x)$
2.	$\mathcal{A} \rightarrow \mathcal{C} : x'$
Winning condition: $x = x'$	

Weakly collision-resistant digest function. Since digest functions have a short output of b -bit, any adversary bounded by a time complexity T_d of order $\Theta(2^b)$ wins the weak collision-resistance game with probability of 1. In other word, digest is $(T_d, 1)$ -weakly collision-resistant.

Weak collision-resistance game against $digest()$	
The challenger \mathcal{C} picks k and m at random.	
1.	$\mathcal{C} \rightarrow \mathcal{A} : (k, m)$
2.	$\mathcal{A} \rightarrow \mathcal{C} : (k', m')$
Winning condition: $digest(k, m) = digest(k', m')$ and $m \neq m'$	

Theorem 2 Consider the direct binding version of Improved V-MANA I authentication protocol. We assume that the function $longhash()$ is a (T_h, ϵ_h) inversion-resistant hash function, and $digest(-, -)$ is $(T_d, 1)$ weakly collision-resistant. Then any powerful adversary against the protocol with time complexity bounded by $(T_h + T_d)Q_A$ and with number of Alice's (resp. Bob's) instances bounded by Q_A (resp. Q_B) has a probability of success at most $(\epsilon_h + \epsilon_d)Q_A$.

The following proof can be slightly modified to cope with the direct binding version of Improved MANA I presented in Section 2.3.

Proof We first find the time complexity and probability of success of a one-shot adversary.

A one-shot adversary has no advantage of sending $(INFO'_A, longhash(k'))$ after the digest is released in the second message thanks to the timing constraints of the bounded delay empirical channel. As a consequence, after $(INFO_A, longhash(k))$ is sent in the first message, there are two possibilities that can happen with the adversary, namely being *able* or *unable* to invert the hash to find out the key.

1. With a probability of ϵ_h , the adversary can invert the hash to discover the key within a time complexity of T_h . After that, with certainty (i.e. probability of 1) he will be able to find a different $INFO'_A$ that produces a collision on the digest under key k in a time complexity of T_d .

Game against the direct binding version of Improved V-MANA I – Inverting hash function	
1. Alice \rightarrow_N \mathcal{A} : $INFO_A, longhash(k)$ \mathcal{A} successfully inverts hash function to find out k	$\mathcal{A} \rightarrow_N$ Bob : $INFO'_A, longhash(k)$
<hr/>	
2. Alice \rightarrow_{BE}^t Bob : $digest(k, INFO_A)$	
<hr/>	
3. Alice \rightarrow_N Bob : k	
Winning condition: $INFO_A \neq INFO'_A$ and $digest(k, INFO_A) = digest(k, INFO'_A)$	

2. Conversely, with a probability $(1 - \epsilon_h)$, the adversary fails to invert the hash value in time complexity T_h . Thus the adversary must select a pair $(k', INFO'_A)$ blindly, and hope that the fresh and unpredictable key k chosen by Alice will result in a digest collision.

Game against the direct binding version of Improved V-MANA I – Failing to invert hash function	
1. Alice \rightarrow_N \mathcal{A} : $INFO_A, longhash(k)$ \mathcal{A} fails to invert hash. \mathcal{A} picks a random pair $(k', INFO'_A)$	$\mathcal{A} \rightarrow_N$ Bob : $INFO'_A, longhash(k')$
<hr/>	
2. Alice \rightarrow_{BE}^t Bob : $digest(k, INFO_A)$	
<hr/>	
3. Alice \rightarrow_N \mathcal{A} : k $\mathcal{A} \rightarrow_N$ Bob : k'	
Winning condition: $INFO_A \neq INFO'_A$ and $digest(k, INFO_A) = digest(k', INFO'_A)$	

Clearly, the probability of success of this case is limited to $(1 - \epsilon_h)\epsilon_d$ thanks to the digest specification.

We conclude that any one-shot adversary bounded by a time complexity $t = T_h + T_d$ has a probability of success

$$p \leq \epsilon_h + (1 - \epsilon_h)\epsilon_d = \epsilon_h + \epsilon_d - \epsilon_h\epsilon_d < \epsilon_h + \epsilon_d$$

We now can apply Theorem 1 to deduce that any powerful adversary has a probability of success at most $(\epsilon_h + \epsilon_d)\mathcal{Q}_A$ within a time complexity of $(T_h + T_d)\mathcal{Q}_A$.

C.2 Improved (V-)MANA I (indirect binding) and security proof

In the following description, R is a b -bit random nonce of party A , which is inputted to a commitment scheme.

Improved version of V-MANA I (indirect binding) [42]	
1. Alice \rightarrow_N Bob : $INFO_A, c$ $(c, d) = \text{commit}(INFO_A, R)$	
2. Alice \rightarrow_{BE}^t Bob : R	
3. Alice \rightarrow_N Bob : d	

While there is no relation between the SAS and $INFO_A$ (i.e. they are completely independent in the sense of probability: indirect binding strategy), the security of the protocol comes from the use of a commitment scheme that firmly binds random nonce R to $INFO_A$ in the eye of the receiver when the decommitment is released in Message 3. However, this protocol is expensive to run because the large $INFO_A$ must be processed by a long output commitment scheme, which is more expensive than a digest function: $W(M+W) = 25+5M$, i.e. an approximate ($W = 5$)-fold increase compared to the direct binding version.

This protocol might also be regarded as the non-interactive version of the pairwise (indirect binding) authentication protocol of Vaudenay [63].

Similar to the direct binding version, we can replace the bounded delay empirical channel with a simple acknowledgement to have the following scheme.

Improved version of MANA I (indirect binding) [42]	
1a.	$A \xrightarrow{N} B : INFO_A, c$ $(c, d) = \text{commit}(INFO_A, R)$
1b.	$B \xrightarrow{E} A : 1\text{-bit commitment}$
2.	$A \xrightarrow{E} B : R$
3.	$A \xrightarrow{N} B : d$

In order to support the security proof of the protocol, all we need is the following security property of a commitment scheme.

Binding property of commitment scheme. A commitment scheme is (T_c, ϵ_c) -binding if any adversary \mathcal{A} bounded by a time complexity T_c wins the following game with probability at most ϵ_c . When $T_c = +\infty$ and $\epsilon_c = 2^{-b}$, we say that the scheme is perfectly binding [63].

Binding game	
The adversary \mathcal{A} picks a random pair (m, c) .	
The challenger \mathcal{C} picks b -bit random number R .	
1.	$\mathcal{A} \rightarrow \mathcal{C} : (m, c)$
2.	$\mathcal{C} \rightarrow \mathcal{A} : R$
3.	$\mathcal{A} \rightarrow \mathcal{C} : d$
Winning condition: $\text{open}(m, c, d) = R$	

Theorem 3 Consider the indirect binding version of Improved V-MANA I authentication protocol. We assume that the commitment scheme is (T_c, ϵ_c) -binding. Then any powerful adversary against the protocol with time complexity bounded by $T_c Q_A$ and with number of Alice's (resp. Bob's) instances bounded by Q_A (resp. Q_B) has a probability of success at most $\epsilon_c Q_A$.

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

Proof A one-shot adversary against this protocol follows the game depicted below in which it runs a man-in-the middle attack.

Game against the indirect binding version of Improved V-MANA I protocol	
0.	$\mathcal{A} \longrightarrow$ Alice : $INFO_A$
1.	Alice $\longrightarrow_N \mathcal{A} : INFO_A, c$ $(c, d) = \text{commit}(INFO_A, R)$ $\mathcal{A} \longrightarrow_N$ Bob : $INFO'_A, c'$
2.	Alice \longrightarrow_{BE}^t Bob : R
3.	Alice $\longrightarrow_N \mathcal{A} : d$ $\mathcal{A} \longrightarrow_N$ Bob : d'
Winning condition: $INFO_A \neq INFO'_A$ and $\text{open}(INFO_A, c, d) = \text{open}(INFO'_A, c', d') = R$	

Clearly this can be reduced to an adversary which plays the binding game against the commitment scheme. As a result, the success probability of a one-shot attack is equivalent to ϵ_c within a time complexity T_c .

Binding game	
1.	$\mathcal{A} \longrightarrow \mathcal{C} : INFO'_A, c'$
2.	$\mathcal{C} \longrightarrow \mathcal{A} : R$
3.	$\mathcal{A} \longrightarrow \mathcal{C} : d'$
Winning condition: $\text{open}(INFO'_A, c', d') = R$	

We now can apply Theorem 1 to deduce that any powerful adversary has a probability of success at most $\epsilon_c Q_A$ within a time complexity of $T_c Q_A$.

C.3 Improved (V-)MANA I (Diffie-Hellman style) and security proof

We are going to describe another improved scheme whose main idea is taken root from the pairwise (direct binding) authentication protocol of Hoepman [22, 23].

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

Improved version of V-MANA I (Diffie-Hellman style) <i>New</i>	
1.	Alice \longrightarrow_N Bob : $\text{longhash}(g^k)$
2.	Alice \longrightarrow_{BE}^t Bob : $\text{hash}_b(g^k)$
3.	Alice \longrightarrow_N Bob : g^k

The main difference between this and the previous two schemes is that there is no $INFO_A$ sent in the first message because the Diffie-Hellman token revealed in the third message plays the dual-role of both $INFO_A$ and the long secret key. Also because of this, the digest function (used in the direct binding version) which requires 2 inputs can be replaced by

a single input $hash_b()$; though the combination of this and the exponentiation of Diffie-Hellman needs to satisfy a specification similar to that of the digest. This results in a very low cost of order $W^2 + W = 30$.

In addition to an inversion-resistant $longhash()$, we need to have the following security property for the $hash_b()$ function.

Weakly collision-resistant shorthash function. Since $hash_b()$ has a short output of b -bit, any adversary bounded by a time complexity T_{sh} of order $\Theta(2^b)$ wins the weak collision-resistance game with probability of 1. In other word, $hash_b()$ is $(T_{sh}, 1)$ -weakly collision-resistant.

Weak collision-resistance game against $hash_b()$
The challenger \mathcal{C} picks m at random.
1. $\mathcal{C} \rightarrow \mathcal{A} : m$
2. $\mathcal{A} \rightarrow \mathcal{C} : m'$
Winning condition: $hash_b(m) = hash_b(m')$ and $m \neq m'$

Theorem 4 Consider the Improved V-MANA I protocol in Diffie-Hellman style. We assume that the function $longhash()$ is a (T_h, ϵ_h) inversion-resistant hash function, and $hash_b()$ is $(T_{sh}, 1)$ weakly collision-resistant. Then any powerful adversary against the protocol with time complexity bounded by $(T_h + T_{sh}) \mathcal{Q}_A$ and with number of Alice's (resp. Bob's) instances bounded by \mathcal{Q}_A (resp. \mathcal{Q}_B) has a probability of success at most $(\epsilon_h + \epsilon_d) \mathcal{Q}_A$.

Proof The proof for this theorem is nearly identical to the proof of Theorem 2, and therefore not given here.