Authentication protocols in pervasive computing

Nguyen Hoang Long
University College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Hilary Term 2009

Abstract

The popularity of personal computing devices (e.g. smart cards) exposes users to risks, notably *identity theft*, and creates new requirements for secure communication. A recently proposed approach to creating secure communication is to use *human trust* and *human interactions*. These approaches potentially eliminate the need for passwords as in Bluetooth, shared secrets or trusted parties, which are often too complex and expensive to use in portable devices.

In this new technology, handheld devices exchange data (e.g. payment, heart rates or public keys) over some medium (e.g. WiFi) and then display a short and non-secret digest of the protocol's run that the devices' human owners *manually* compare to ensure they agree on the same data, i.e. human interactions are used to prevent fraud.

In this thesis, we present several new protocols of this type which are designed to optimise the work required of humans to achieve a given level of security. We discover that the design of these protocols is influenced by several principles, including the ideas of *commitment without knowledge* and *separation of security concerns*, where random and cryptographic attacks should be tackled separately.

Underpinning the technology is a new cryptographic function, termed a *keyed digest function*, which produces a short number for humans to compare. This is similar to the notion of a universal hash function, but its output length is shorter (e.g. 16 bits). Hence, it should be faster to compute. We propose several digest constructions using Toeplitz matrices, integer multiplication and pseudorandom numbers. The application of digest functions leads us to develop more efficient alternatives to standard digital signatures.

Our protocol security analysis leads to a new bound on the key length for an almost universal hash function, which can be derived by the *pigeon-hole* principle. The new bound turns out to be tighter than another similar bound derived from the combination of the *Singleton* bound in coding theory and an equivalence between error-correcting codes and almost universal hash functions.

Acknowledgements

I would like to thank my supervisor, Professor Bill Roscoe, who has made many large contributions to this work as well as given me much advice throughout my time at Oxford.

This work is supported by studentships from the Oxford University Computing Laboratory.

I would like to thank Professor Gavin Lowe (my D.Phil. advisor), Dr. Andrew Ker (my College advisor), Dr. Dusko Pavlovic (a long-term visiting academic at Oxford), Dr. Andrew Simpson (my D.Phil. internal examiner) and Professor Chris Mitchell (my external D.Phil. examiner from Royal Holloway, University of London) for making suggestions as how the structure of a thesis should be organised as well as correcting many technical and English mistakes in my D.Phil. transfer status report (Hilary 2007), D.Phil. status confirmation report (Michaelmas 2008) and the preliminary versions of this thesis (Hilary and Summer 2009).

In addition, thanks also go to my colleagues from other universities for many fruitful conversations and new ideas as well as inviting me to give talks about my work at their institutions, especially Professor Richard Brent (Australian National University) for helping me understand pseudo-random numbers, Professor Bart Preneel (Katholieke Universiteit Leuven) for telling me the relationship between bounds for *almost* XOR *universal* and *almost strongly universal* families of hash functions, Professor Chris Mitchell for suggesting me to serve as editor of an international standard, and Professor Nigel Smart (Bristol University) for information about cost model of hashing as well as Professor Kenny Paterson (Royal Holloway, University of London), Dr. Siraj Shaikh (Cranfield University), Professor Ross Anderson (Cambridge University), Professor Bruce Christianson (Hertfordshire University), Professor Serge Vaudenay, Professor Jean-Pierre Hubaux and Dr. Atefeh Mashatan (Ecole Polytechnique Federale de Lausanne), Professor Srdjan Čapkun (ETH Zurich), Dr. Sasa Radomirovic and Dr. Baptiste Alcalde (University of Luxembourg), Professor Jaap-Henk Hoepman (Radboud University Nijmegen), Professor Kaisa Nyberg (Helsinki University of Technology), Professor Adrian Perrig (Carnegie Mellon University), Dr. Dusko Pavlovic (Kestrel Institute), Professor John Mitchell (Stanford University), Dr. Diana Smetters (Xerox PARC), Professor Kefei Chen (Shanghai Jiao Tong University), Professor He Jifeng (East China Normal University), Professor Andrew Yao (Tsinghua University), and Professor Gene Tsudik (University of California, Irvine).

I would like to dedicate my work to my parents who have made sacrifices throughout their lives so that I am able to come to Britain to pursue a higher education, and to complete my D.Phil. studies at the University of Oxford.

Contents

1 Introduction **1**
- 1.1 Motivation and Applications . . . 1
- 1.2 Overview . . . 3

2 Notation and basic definitions **8**
- 2.1 Communication links . . . 9
- 2.2 Protocol citations . . . 10
- 2.3 Cryptographic primitives . . . 11
  - 2.3.1 Commitment schemes and Commitment without knowledge . . . 12
  - 2.3.2 Universal hash functions . . . 15
  - 2.3.3 Short hash and digest functions . . . 15
  - 2.3.4 Digital signatures . . . 16
- 2.4 Attack model . . . 17
- 2.5 Cost model . . . 19
  - 2.5.1 Human effort . . . 19
  - 2.5.2 Computation cost of cryptographic primitives . . . 20

3 Pairwise authentication protocols **24**
- 3.1 Multiple empirical short authentication strings . . . 26
- 3.2 Indirect binding . . . 29
  - 3.2.1 Indirect pairwise . . . 29
  - 3.2.2 Hybrid protocol . . . 33
- 3.3 Direct binding pairwise protocols . . . 33
- 3.4 Efficiency of pairwise authentication protocols . . . 39
- 3.5 Conclusions . . . 41

4 Group protocols **42**
- 4.1 Man-in-the-middle attack on a group protocol . . . 44
- 4.2 Asymmetrised group protocols . . . 45
  - 4.2.1 HCBK protocol analysis . . . 47
  - 4.2.2 Modified versions of HCBK . . . 51
- 4.3 Symmetrised group protocols . . . 52
  - 4.3.1 SHCBK protocol analysis . . . 55
  - 4.3.2 Modified version of the protocol SHCBK with proof of security . . . 57
  - 4.3.3 Indirect binding group protocol . . . 57
- 4.4 De-symmetrised SHCBK protocol . . . 59
- 4.5 Hybrid of the HCBK and SHCBK protocols . . . 61
- 4.6 Efficiency of group protocols . . . 63
- 4.7 Conclusions . . . 64

5 Non-interactive and one-way authentication protocols **65**
5.1 Long authentication string protocols . . . 67
5.2 Short authentication string protocols . . . 70
5.3 Improved versions of the (V-)MANA I protocol . . . 73
5.3.1 Direct binding solution . . . 73
5.3.2 Indirect binding solution . . . 75
5.3.3 Solution in Diffie-Hellman style . . . 76
5.4 Security analysis of the Improved (V-)MANA I protocols . . . 76
5.4.1 Security analysis of the direct binding improved (V-)MANA I . . . 78
5.4.2 Security analysis of the indirect binding improved (V-)MANA I . . . 81
5.4.3 Security analysis of Improved V-MANA I in Diffie-Hellman style . . . 81
5.5 Efficiency of one-way authentication protocols . . . 83
5.6 Separation of security concerns . . . 84
5.7 Digital signature without hashing . . . 86
5.8 Flexi-MACs . . . 89
5.9 Conclusions . . . 90

6 Digest functions **92**
6.1 Notation . . . 93
6.2 Background information . . . 95
6.3 Toeplitz matrix product construction of a digest function . . . 96
6.3.1 Efficiency of Toeplitz matrix based digest functions . . . 98
6.3.2 Hardware implementation of digest functions . . . 98
6.4 Comparing Toeplitz matrix and integer long multiplication . . . 100
6.5 Word multiplication based digest functions . . . 101
6.5.1 Efficiency of word multiplication based digest functions . . . 103
6.5.2 Security analysis and usability . . . 104
6.6 Statistical tests of word and Toeplitz matrix multiplications . . . 106
6.6.1 Analysis of statistical test results . . . 108
6.7 Conclusions and future research for short-output functions . . . 110

7 New bounds for almost universal hash functions **124**
7.1 Notations and definitions of universal hash functions . . . 126
7.2 Bounds for almost universal hash functions . . . 127
7.2.1 New AU-bound . . . 128
7.2.2 Error-correcting codes and almost universal hash functions . . . 129
7.3 The significance of the threshold value of ϵ . . . 131
7.3.1 Comparison between the new and other AU-bounds . . . 131
7.3.2 Comparison between the new AU-bound and known ASU-bounds . . . 132
7.3.3 Comparison between the new AU-bound and known AXU-bounds . . . 134
7.4 The optimality of *polynomial hashing* as an AU . . . 135
7.5 Conclusions . . . 136

8 Conclusions and future research **138**
8.1 Conclusions . . . 138
8.2 Short-term public key cryptography . . . 140
8.3 Future research . . . 141
8.3.1 Security proofs of the new family of authentication protocols . . . 141
8.3.2 Digest functions . . . 141
8.3.3 Applications of digest functions . . . 142
8.3.4 Unifying bounds of universal families of hash functions (UHF) . . . 142
8.3.5 Relaxation in human work . . . 143

Chapter 1

Introduction

1.1 Motivation and Applications

Imagine that a group of people come together and agree that they want to transfer data between them securely, meaning that they want it to be secret and of authenticated origin. They all have some pieces of computing hardware (e.g. a mobile phone or a Personal Digital Assistant, PDA). Unfortunately none of them knows the unique name of any of the others' equipment, and in any case no Public Key Infrastructure (PKI) would encompasses them all. How can they achieve their goal in the context that their machines are connected by an insecure network (whether created by WiFi, the Internet, telephony, or a mixture of these)?

In the absence of established cryptographic keys (whether shared secret keys or reliable copies of each other's public keys) this goal is very difficult to achieve. However, a little creative thinking can easily solve the problem: if each person tells the others (using human conversation) a public key for his or her machine, they can then use, for example, the Needham-Schroeder-Lowe protocol [63] (over the insecure network) to establish secure and authenticated communication. (If there are more than two participants then they would either have to adapt that protocol or use it multiple times.) They will have bypassed the intruder for the crucial step of exchanging electronic identities. The problem with this approach is that it requires too much human effort for practical purposes.

The use of public key infrastructures or trusted third parties will often be anywhere from inconvenient to impossible in emerging pervasive computing applications. This is mainly for two reasons: 1. it is expensive to maintain a centralised security infrastructure, such as a PKI, storing information regarding numerous lightweight devices as well as keeping up-to-date with frequent changes, e.g. in

locations or public keys, in pervasive environments; and 2. even when this infrastructure is present, it frequently only identifies nodes by their identification (ID) fields through electronic ID certificates — often completely inappropriate in the world of lightweight and human-driven communication. We are much more likely to identify the intended participants by some aspect of their context, such as "the printer I *see in front* of me" or "the till *sitting in front* of Jim", which needs to be captured by human interactions.

It has been widely realised that it is impossible to bootstrap security from nothing. Nevertheless, as we have discussed above, it is necessary to be able to bootstrap it from minimal assumptions. So what is it reasonable to assume exists prior to an attempt to acquire high-quality security? There have been (at least) two separate approaches to this. One is to assume that a pair of parties, who are seeking to exchange a strong secret key, already share some short or low-entropy secret, such as a password. The other assumes the existence of low-bandwidth *empirical* channels, which are not susceptible to spoofing, although they are liable to be overheard or blocked by the intruder.[1] Based on either assumption, the parties can agree on a strong shared secret from scratch.

The first method, which is based on possessing a shared password, has been studied extensively in the past. Bluetooth itself uses such a protocol. However, Bluetooth has the weakness of only being usable in (usually local) situations where there is a secret shared password, and is subject to severe off-line password guessing attacks [50]. The community has devised a variety of various password based key establishment protocols (see, for example, Bellare, Rogaway et al. [11, 13, 14] and Canetti, Krawczyk et al. [23, 24]) to prevent the off-line dictionary attacks that Bluetooth allows. These ensure that the only way an attacker can find out whether his random guess of the password is correct is by interacting with the player. Typically these reduce the probability of a successful attack to that of guessing the password correctly in a single trial. However, the need to keep the password secret in such protocols precludes their use in scenarios subject to eavesdropping, including most remote ones. For example, without pre-existing secure channels, passwords cannot be transmitted between a pair of parties.

Taking a different approach that makes use of human interactions and human trust to creat a secure and low bandwidth channel, Stajano and Anderson [106] described a method of forming a secure network for a two-party scenario. This approach was refined further by many researchers who frequently worked independently of one another, including Balfanz et al. [10], Creese et al. [28, 29, 30, 31], Gehrmann et al. [36], Hoepman [48, 49], Vaudenay [119], and Čagalj et al. [22], who introduced both pairwise and group authentication protocols using less human interactions. Creese, Roscoe et

[1] Formal specifications and classification of different types of empirical channels can be found in Section 2.

al. [28, 29, 30, 31, 96] refer to these human exchanges as empirical channels, since the recipient has some empirically based knowledge as opposed to cryptographically based knowledge (e.g. via a PKI) about the origin of a message. We therefore use this term throughout this thesis. Examples of these channels in practice are human conversations (i.e. people can recognise each other's voices and reject forged spoken utterances), physical touch (metal against metal), and copying and pasting strings of characters through the use of monitors and keyboards (manual data transfers), i.e. information transmitted over empirical channels cannot be faked, modified or spoofed by the intruder, but may be overheard by anyone.

In the latter approach, to communicate, authenticate and agree on some public data, such as payment, heart rates or cryptographic public keys, electronic devices exchange the data over some insecure medium, e.g. WiFi or the Internet. The devices then compute and display a short digest of the protocol's run that the devices' human owners *verbally* or *visually* compare to ensure that they agree on the same (public) data, i.e. the schemes use human interactions to prevent fraud. In other words, they try to bootstrap security communication without the need for any pre-existing infrastructure, such as a PKI or shared passwords. As a result, this technology can make authentication much more usable and accessible to many applications of low power devices. Apart from group authentication scenarios described at the begin of this section, other examples are: (1) establishing a secure connection between medical sensors and other devices in a hospital or in e-healthcare (telemedicine); and (2) establishing a secure connection in financial transactions, e.g. among a smart card, a security device of a customer, a merchant's computer and the bank over the Internet (secure HTTP) or telephone.

Since this approach could be an interesting alternative to, and has a clear advantage over, PKI and password systems in pervasive environments, we pursue this here. We analyse the security and efficiency in both empirical and computational costs of many known protocols, as well as introducing improved and new schemes in different scenarios.

1.2 Overview

This thesis, particularly as chapters 3–5 covers the differing approaches of several research groups, and since it sits at the boundary of cryptography and protocol design, requires a lot of (new) notation and definitions. The next chapter is devoted to introducing this, i.e. the notation of various types of empirical channels and cryptographic primitives, such as (universal or short) hash functions, commitment schemes and digest functions. We propose cost models for the computation of cryptographic primitives as well as quantifying empirical or human work; these are used to compare the protocols

proposed in this thesis with the prior art. The reader might either choose to study all of this notation in Chapter 2 in advance, or can read it to the end of Section 2.1 and then refer back to section 2.3, 2.4 and 2.5 as needed.

We describe protocols providing mutual, group and one-way authentication in chapters 3, 4 and 5. We will explain the fundamentals of designing this family of authentication protocols in the following order: (1) the introduction of several principles which govern the design and security of pairwise and group authentication protocols; and subsequently (2) the application of the principles as well as new ones to design one-way and non-interactive schemes. It must be admitted that some of the one-way schemes are just simplifications of pairwise and group ones, though the others that are both one-way and non-interactive are not.

In Chapter 3, we discuss mutual authentication protocols using manual data transfers, e.g. human comparison of short authentication strings (SAS) (say 16 bits). We see that it is better, in terms of human work, if only a single SAS needs to be manually compared by humans. This will be further demonstrated when we discuss group authentication protocols in Chapter 4. We demonstrate that the security of these protocols relies on some values (i.e. the SASs) to which everyone is *jointly committed* at the beginning of a run by publishing their individual shares of the commitment. These parties, however, only find out the jointly committed value after all parties publish their shares of the decommitment. The protocols are subsequently categorised into two main groups, depending on whether the information parties try to authenticate is functionally dependent on the jointly committed SASs or not. These strategies are termed *direct* and *indirect* information bindings, and thus leading to different behaviours in terms of computational efficiency.

In order to optimise the human work, it is essential to decrease the number of SASs to a single one, which is crucial when we visit group authentication protocols in Chapter 4. We point out that it is easy to design protocols that are vulnerable to a search or birthday attack, but these attacks can be prevented by using the ideas of *commitment without knowledge* and *joint commitment without knowledge* as mentioned in Chapter 3. These together shape the design strategies of not only group protocols but also pairwise schemes. We point out an interesting trade-off between human and computational costs in different group protocols in this chapter. In addition the security analysis of our group protocols introduces us to a new cryptographic primitive, termed a *keyed digest function*. This is similar to a universal hash function, but has significant differences, such as its short output and collision properties, which will be analysed in greater detail in Chapter 6.

All schemes visited in chapters 3 and 4 are interactive in the sense that all participants have to participate in a protocol run. It is however desirable in many scenarios to design protocols that can

cope with the absence of one or many participants. We refer to these as non-interactive protocols, and they will be discussed in Chapter 5. We will see that non-interactiveness brings with it some constraints which are not present in pairwise and group protocols, and which cannot be solved by using the idea of *joint* commitment without knowledge. For example, to defeat powerful attacks which involve searching[2] we need to use either (1) more empirical bits (i.e. more human work in the protocols of Pasini and Vaudenay [87], Balfanz et al. [10], and Mashatan and Stinson [68, 69]), or (2) stronger empirical channels as in the MANA I protocol [36]. In this chapter, the analysis of some protocols that require a long authentication string leads to the discovery of two different roles of hashing, which relates to a protocol design principle termed *separation of security concerns*. What this means is that protocols should be designed to tackle guess[3] attack and searching attack separately. Using this principle, we develop: (1) several improved versions of the MANA I protocol, and (2) a new family of digital signatures, where we replace the use of cryptographic hashing with more efficient digest functions to increase efficiency by a significant factor.

Many protocols considered in this thesis rely on the use of a new cryptographic primitive, termed a *digest function*, which has similarities to ϵ-balanced and almost universal hash functions. Digest functions, however, typically have a very short output, e.g. 16–64 bits, and hence they are not not required to resist collision and inversion attacks. They also have the potential to be very fast to compute relative to long-output hash functions. In Chapter 6, we introduce two related methods of generating a digest function. The first construction uses Toeplitz matrix multiplication, which is similar to a Toeplitz based universal hashing algorithm of Krawczyk, whose security requirements can be reduced to the underlying ϵ-biased sequences of random variables. The second is based on integer long multiplications which have, perhaps surprisingly, a similar structure to Toeplitz matrix multiplication. However, due to the complication of carry bits, a rigorous mathematical proof of the second construction cannot be provided. We instead exploit the short output of digest functions to carry out statistical analysis, including chi-square tests, quantile-quantile plots and maximum median calculation, of digest collision and distribution test results to argue for the security of the second construction.

In Chapter 7 we introduce a new lower bound on the key length for *ϵ-almost universal* families of hash functions, arising from our security analysis of the MANA I protocol presented in Chapter 5. This result gives us a lower bound on the bitlength of the hash key with respect to the message bitlength, the hash output bitlength and the hash collision probability ϵ. We derive the bound using the pigeon-

[2]Definitions of attack types can be found in Chapter 2.

[3]One in which the intruder makes a single attempt without doing any searching. More information is given in Section 2.4 of Chapter 2.

hole principle, and also show that this bound is tighter than another similar bound derived from the Singleton bound of coding theory. In comparing the bound against other well-studied bounds for this and other universal families of hash functions, we discover a crucial value of the security parameter ϵ that represents an important *threshold* in the behaviour of bounds, and in effect quantifying the *Wegman-Carter* effect [120, 19]. In addition, another new lower bound on the key length for *almost XOR universal* families of hash function is introduced.

We conclude the thesis with a closer look at some directions for future research arising directly from this thesis. The work presented in chapters 3–6 has been reported in a number of papers, published in two workshops [78, 82] (FCS-ARSPA'06 and FCS-ARSPA-WITS'08), two journals [79, 80] (Information and Computation, and Journal of Computer Security), and three international patent applications [97, 98, 99]. We have also incorporated some of the protocols into international standards [84] (ISO/IEC), since they are demonstrably better than the currently standardised techniques of this type.

The protocols in this thesis are organised according to their aims (e.g. group, pair and one-way) and structure (direct versus indirect binding strategies). 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 1.1 shows all relevant protocols both by year of publication, section number where the protocol is described in this thesis 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.

Figure 1.1: Summary of all protocols.

Chapter 2

Notation and basic definitions

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

We will often assume each node A in a group $\mathbf{G}$ of N parties has some information $INFO_A$, representable by K bits, that it wants to have authenticated to other members of the group. The goals of the protocols will always include authenticating pairs $(A, INFO_A)$ as members of the network.[1]

Each $INFO$ might include:

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

Nothing in this information should be secret, since all the protocols we consider will make it public. $INFO_A$ might be attached to A permanently or for the long term; alternately some of it might be

[1] We make the identity A explicit here and in the protocols using it, since the identity is vital to an understanding of who is in the group. In practice, 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.

relevant to this particular run only.

We refer to $INFOS$ as the concatenation in alphabetical order of all the distinct pairs of the form $(A, INFO_A)$ that the parties want to authenticate. If $INFOS$ contains N photographs or similar, it may well be of significant size.

2.1 Communication links

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

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

- $\longrightarrow_N$ represents the normal Dolev-Yao network [34], where all messages transmitted between devices using such a channel can be overheard, deleted or modified by the intruder. Examples of this channel are the Internet, WiFi, or a local network.

- $\longrightarrow_{WE}$, the *weak empirical* channel, cannot be forged, i.e. messages cannot be modified and the receiver knows where messages come from. However, this channel can be blocked, overheard, delayed or replayed. This is the weaker of the two forms of empirical channel described in [87, 119]. Typical examples of such a channel include telephone conversations, voice mails or messages, where messages can be delayed, blocked or replayed, but cannot be forged by the intruder.

- $\longrightarrow^t_{BE}$ is similar to $\longrightarrow_{WE}$ except that messages cannot take more than time t to arrive and cannot be replayed. In other words, no empirical message can be accepted more than t time units after it was sent. Such a channel might be implemented over a reliable medium with a known bound on transmission, or over an unreliable one with the addition of some sort of time-stamp. The latter might make sense if the empirical message is sent by some video means, but otherwise would add significantly to the communication burden. We call this a *bounded delay empirical channel* [80].

- $\longrightarrow_E$ is the empirical channel assumed in [78, 79]. Messages transmitted over such a channel cannot be mistaken or delayed from one to another session. To some extent, this implies that $\longrightarrow_E$ is a special case of $\longrightarrow^t_{BE}$ where t is chosen to be some lower bound on the length of time between one session and a later one. This is the type we use most often. Sometimes the two-way arrow $\longleftrightarrow_E$ is used to indicate (possibly) the same message is transmitted in both directions.

  An example of such a channel is provided by manual data transfers [36, 37, 38], i.e. human users manually copy data from one to another device via their in/output interfaces such as screen, monitor and keyboard. In this case, empirical messages cannot be mistaken or delayed from one to another session because: (1) the humans involved are not away at any time during a protocol run, and (2) each device normally only has one in/output interface for displaying or reading data.

- $\longrightarrow_{SE}$ is similar to a normal empirical channel, but it also provides stall-free transmission. As a result, a message transmitted over the channel cannot be delayed, removed or blocked by the intruder. This implies that $\longrightarrow_{SE}$ is the same as $\longrightarrow^t_{BE}$ where $t \approx 0$. We term this a *strong empirical channel* (or a strong authentication channel). This was also defined in [87, 119]. Face to face human conversation is an example of this channel.

2.2 Protocol citations

Many of the protocols described in this thesis are taken from the literature. However, we will describe minor improvements to some, major improvements to others, as well as completely new protocols. We will use the following notation in protocol descriptions:

- Protocols from the literature, although perhaps described using different notation, are just cited: [].
- Protocols which have been modified in minor ways, often by replacing cryptographic primitives or other data operations, are cited as []*.
- Protocols which are either major modifications to existing ones or are just new are marked *New* and cited.

2.3 Cryptographic primitives

We will be using a variety of cryptographic primitives which are related to hashing. Since bitlengths of hash functions vary widely, for clarity, they will be classified relative to three common security properties of a cryptographic hash function [74], which are inversion-resistant, collision-resistant, and second-preimage-resistant, whose definitions in the complexity-theoretical model are given below.

Definition 1 [74] Suppose $longhash : \{0,1\}^* \rightarrow \{0,1\}^B$. Then $longhash$ is said to be inversion-resistant if, for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x_0 such that $longhash(x_0) = y$ when given any $y \in \{0,1\}^B$ for which a corresponding input is not known.

The notation (ϵ_i, T_i)-inversion-resistant indicates that the success probability of any algorithm or adversary bounded by a time complexity T_i in inverting a hash value, whose corresponding input is not known, is bounded above by ϵ_i.

Definition 2 [74] Suppose $longhash : \{0,1\}^* \rightarrow \{0,1\}^B$. Then $longhash$ is said to be second-preimage-resistant if it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage $x_0 \neq x$ such that $longhash(x) = longhash(x_0)$.

The notation (ϵ_s, T_s)-second-preimage-resistant indicates that the success probability of any algorithm or adversary bounded by a time complexity T_s in finding a second preimage is bounded above by ϵ_s.

Definition 3 [74] Suppose $longhash : \{0,1\}^* \rightarrow \{0,1\}^B$. Then $longhash$ is said to be collision-resistant if it is computationally infeasible to find any two distinct inputs x, x_0 which hash to the same output, i.e., such that $longhash(x) = longhash(x_0)$. (Note that here there is free choice of both inputs.)

The notation (ϵ_c, T_c)-collision-resistant indicates that the success probability of any algorithm or adversary bounded by a time complexity T_c in finding a hash collision is bounded above by ϵ_c.

Following is the list of all hash functions used in this thesis. Sometimes they will be subscripted with the number of output bits.

- $longhash(X)$ and $hash_B(X)$ refer to standard cryptographic hash functions [74] which are assummed to be inversion-resistant, collision-resistant and second-preimage resistant. In common with the literature, we will generally assume that they have at least $B = 160$ bits.

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

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

We will also use a commitment scheme, a universal hash function, a digest function, and a digital signature whose definitions are given below.

2.3.1 Commitment schemes and Commitment without knowledge

Definition 4 A probabilistic commitment scheme,[2] following Vaudenay [119], consists of two mappings:

- *commit*: $\{0,1\}^* \times \{0,1\}^b \rightarrow \{0,1\}^B \times \{0,1\}^B$

  This mapping takes a public and arbitrary-length data $INFO$, a private b-bit random nonce R (e.g. $b = 16$), and then internally generates a $(B - b)$-bit random nonce and finally produces two B-bit strings (e.g. $B = 160$): a commit value c and a decommit value d. This algorithm is nondeterministic since it uses an internally generated random element of $B - b$ bits.

- *open*: $\{0,1\}^* \times \{0,1\}^B \times \{0,1\}^B \rightarrow \{0,1\} \times \{0,1\}^b$

  This mapping takes a public and arbitrary-length data $INFO$, a commit value c and a decommit value d, and produces an error or success signal together with a b-bit random nonce R. This algorithm is deterministic, and has the property that whenever there exists an R such that (c, d) is a possible output for $commit(INFO, R)$, then $open(INFO, c, d)$ yields R.

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

A commitment scheme in the complexity-theoretical model usually has the following two properties:

- (ϵ_h, T_h)-*hiding* [119]: no algorithm or adversary I bounded by a time complexity T_h can win the following game by interacting with a challenger C with a probability higher than ϵ_h.

[2] Please note that to the best of our knowledge there does not seem to be a standard definition of a commitment scheme in the literature, because: (1) the number of inputs (e.g. messages and nonces) varies; (2) the committed input message can be either public or private; and (3) the commitment scheme can be either probabilistic or deterministic.

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

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

- (ϵ_b, T_b)-*binding*:[3] no algorithm or adversary I bounded by a time complexity T_b can win the following game with a probability higher than ϵ_b.

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

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

Note, the above hiding and binding properties also cover the infeasibility that given $(c, INFO)$ the intruder could change the value of $INFO$ to $INFO'$ and then come up with a pair (c', d') such that $open(INFO', c', d') = R$ where c' can be either different from or equal to c; because (1) when $c' = c$, this is at least as hard as the binding game; and (2) when $c' \neq c$, to be successful the intruder must correctly find the value of R which is equivalent to the hiding game.

The bitlength of R will be short, e.g. $b \in [16, 20]$ or even $b = 0$,[4] in all the protocols considered. To prevent brute-force search and birthday attacks, the commitment scheme (i.e. $commit()$) needs to extend R by a randomly chosen secret nonce R' of $B - b$ bits so that the combination $R \parallel R'$ has the same bitlength as the output of a cryptographic hash function. This implies that a commitment scheme is designed to be as secure as a standard cryptographic hash function $hash_{160}()$ or $longhash()$. In practice, a commitment scheme is usually built from a pseudorandom function such as a hash function, and therefore they have the same computational complexity. This issue will be discussed in more detail in Section 2.5.2.

[3] We note that there is a lack of explicitness in the binding property of the commitment scheme defined by Vaudenay in [119], since the security specification there fails to bind it to the input message, as was obviously intended. The definition of $commit(INFO, R)$ there is satisfied by defining the commitment $c = longhash(N, r)$, where N is a long random nonce internally generated by the committer, and the decommitment $d = (N, r)$.

[4] When the random nonce R is null, as in the one-way and non-interactive scheme of Pasini-Vaudenay of Section 5.1, we drop R in the notation for a commitment scheme, i.e. we write $commit(INFO)$.

The above security and efficiency properties of a commitment scheme can be demonstrated by the following construction which was introduced in [89] by Pass.

- **Committing**: to bind some public information $INFO$ and a short random secret key R of b bits together, the algorithm picks another secret random nonce $R' \in_{random} \{0,1\}^{B-b}$. It then sets $d = R \parallel R'$ and $c = longhash(INFO \parallel d)$, i.e. $commit(INFO, R) = c \parallel d$. The committer then publishes the commitment c.[5]

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

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

The first use of this idea is called *commitment without knowledge*, which directly influences the design of HCBK in Section 4.2, and some of the one-way and non-interactive protocols in Chapter 5.

Definition 5 A protocol using a short authentication string (SAS) is said to achieve *commitment without knowledge* if there is a point at which one or more protocol participants are committed to the SAS but without knowing the SAS.

The chief purpose of this idea is to ensure that a powerful attack which might consist of searching and multiple protocol runs does not gain any advantage over a random attack. This argument will be explained in greater detail in Section 2.4 which presents definitions of a variety of attacks carried out by the intruder.

The same properties also apply to a more restrictive use of the idea, called *joint* commitment without knowledge and introduced by us in [80, 82], and which influences the design of nearly every interactive protocol, i.e. pairwise and group authentication schemes in chapters 3 and 4.

Definition 6 A protocol using a SAS is said to achieve *joint commitment without knowledge* if there is a point at which all protocol participants are jointly committed to the SAS but without knowing the SAS.

[5] Instead of using a cryptographic hash function $longhash()$, a universal hash function as defined in Section 2.3.2 can be used, i.e. $c = longhash(INFO \parallel d)$ is replaced with $c = h_d(INFO)$. In such a case a similar change needs to be made to decommitting.

One simple way to achieve *joint commitment without knowledge* is as follows: each node is committed to some value by publishing its commitment, for example, by using the commitment scheme described above or a cryptographic hash function. The jointly committed value (i.e. SAS) is then the output of some function, such as summation, exclusive-or, Diffie-Hellman key agreement or *digest* functions, applying to all of the values to which every node has been committed.

We will see later that there are two different strategies for achieving (joint) commitment without knowledge, i.e. *direct* and *indirect* information binding approaches which are formally introduced in Chapter 3. In addition, the combination of commitment without knowledge and the *direct binding* approach will be refined by the principle **P2** to design several group authentication protocols in Section 4.2.

2.3.2 Universal hash functions

Universal hash functions were introduced by Carter and Wegman [25, 120].

Definition 7 [25] A universal hash function is a set of 2^r hash functions $h_k()$ where $k \in R$ such that $h_k : X \rightarrow Y$; here $R = \{1...2^r\}$, $X = \{1...2^K\}$ and $Y = \{1...2^b\}$. Moreover, for every m, $m' \in X$ $(m \neq m')$, we have:

$$\Pr_{\{k \in R\}}[h_k(m) = h_k(m')] \leq 2^{-b}$$

A number of well-studied families of *universal* hash functions are described in the literature. However, in this thesis, we are mostly interested in ϵ-*almost universal* hash functions (ϵ-*AU*) in which the hash collision probability is parameterised by ϵ as opposed to being fixed at 2^{-b}.

Definition 8 [110] An ϵ-*almost universal* hash function ϵ-*AU* (r, K, b) is a set of 2^r hash functions $h_k()$ where $k \in R$ such that $h_k : X \rightarrow Y$; here $R = \{1...2^r\}$, $X = \{1...2^K\}$ and $Y = \{1...2^b\}$. Moreover, for every m, $m' \in X$ $(m \neq m')$, we have:

$$\Pr_{\{k \in R\}}[h_k(m) = h_k(m')] \leq \epsilon$$

2.3.3 Short hash and digest functions

In this thesis, we will see various uses of new functions that, like cryptographic and universal hash functions, are intended to randomise and convey no useful information about the preimage. However, their outputs are significantly shorter, since they will always produce values of short authentication strings (SASs) transmitted over the empirical channels which can be very limited in bandwidth.

We will see two variants on this idea: the simpler is what we call a short hash: $shorthash()$. This has a single argument, and is intended to be uniformly or near-uniformly distributed over its b-bit range as its argument varies. Since the hash output is too short, it is impossible to have properties such as collision and inversion resistances like in cryptographic hash functions. What is required instead is the *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 [121], i.e. the probability of each output bit flips is independent from each other and is equal to a half.

In some of the protocols we consider, we need to construct a b-bit digest of $INFOS$ and some key k. Similar to $shorthash()$, digests cannot have the usually specified properties of 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 [78, 79].

Definition 9 [78, 79] A keyed *digest* function: $digest : R \times X \rightarrow Y$ where $R = \{1...2^r\}$, $X = \{1...2^K\}$ and $Y = \{1...2^b\}$ are the set of all keys, input messages and digest outputs, and moreover:

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

The rationale for these two specifications, especially the use of '$\oplus\ \theta$' where $\oplus$ refers to the bit-wise exclusive-or operator, will become apparent when we analyse group protocols such as SHCBK in Section 4.3.1. The digest specification has similarities to *universal* hash functions, except the probability of digest collisions relative to different keys, i.e. k and $k \oplus \theta$, is also considered. For example, the way digest keys are agreed between nodes in the SHCBK (or HHCBK) protocol of Chapter 4 can be manipulated such that different nodes' keys may be relatively shifted by a θ known to the intruder. The inclusion of the θ shift is therefore to ensure that this type of activity can never benefit the intruder.

Although both $shorthash()$ and $digest()$ are less secure than cryptographic hash functions, they are potentially faster to compute, thanks to their short outputs. More detail about their comparative speed performance and efficient digest constructions can be found in Section 2.5.2 and Chapter 6.

2.3.4 Digital signatures

The following definition of a digital signature scheme is taken from Smart [105].

A digital signature scheme consists of two transformations:

- a secret signing transform $sign()$,

- a public verification transform *verify*().

In the following discussion, we describe a signature scheme with appendix because the signature is attached to the message before transmission. A, sending a message m, calculates $longhash(m)$ which turns an arbitrarily large message m into a fixed length bit stream of B bits, and then computes:

$$s = sign_A(longhash(m))$$

and then transmits (m, s), where s is the digital signature on the message m. Note, we are not interested in keeping the message secret here, since we are only interested in knowing who it comes from. The receiver of the signature (m, s) applies the public verification transform *verify*() to (m, s). The output is a bit v. The bit v indicates valid or invalid, i.e. whether the signature is good or not. If v is valid the recipient gets a guarantee of three important security properties:

- message integrity: the message has not been altered in transit,
- message origin: the message was sent by A,
- non-repudiation: A cannot claim (s)he did not send the message.

Note, the first two of these properties are also provided by message authentication codes, MACs. However, the last property of non-repudiation is not provided by MACs and has important applications in e-commerce. To see why non-repudiation is so important, consider what would happen if you could sign a cheque and then say you did not sign it.

2.4 Attack model

In Section 2, we defined the intruder's power over data transmitted over all kinds of channels used in the family of protocols. In addition to that, we will also adopt the security model of Bellare and Rogaway [11] where the intruder is allowed to have a full control on which node a new instance of the protocol is launched.

The following are definitions of several types of attack that can be performed by the intruder.

Definition 10 *A powerful attack* makes use of (on- or off-line) searching, e.g. using the birthday paradox to search for hash collisions. This can be either interactive or non-interactive. The attack may consist of *multiple protocol runs*, and so can involve use of previously recorded messages. We also refer to this attack as a *multiple-shot* or *q-shot* attack, where q is the number of protocol runs

involved, i.e. in a pairwise scheme between A and B, a q-shot attack implies that the number of A's and B's instances are $Q_A = Q_B = p$. Of course, it is also possible that $Q_A \neq Q_B$.

Definition 11 *A one-shot attack* is similar to a powerful attack, except that it only involves a single protocol run, i.e. $Q_A = Q_B = 1$. Consequently, this only involves use of searching, but not previously recorded messages.

Definition 12 *A random attack* does not involve any searching. The intruder just *randomly* or *blindly* modifies, deletes or replays information transmitted in a protocol run, i.e. this does not involve use of previously messages and protocol transcripts.

Our aim is to ensure that powerful or multiple-shot attacks give the intruder no advantage over a one-shot attack on this family of protocols.[6] This goal can be 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) which are randomised by random keys and nonces generated by trustworthy parties in each protocol run, then the agents' state of knowledge of the SASs is a uniform distribution, and the SASs of all protocol runs are independent of one other as pointed out by several authors in [61, 82, 87, 119].

Moreover, in the majority of protocols considered in this thesis, SASs are transmitted over (strong) empirical channels ($\longrightarrow_E$ and $\longrightarrow_{SE}$), and therefore cannot be mistaken or delayed from one to another session.[7] Hence for any q we have:

$$\begin{aligned} \mathbf{Pr}(\text{a successful } q\text{-shot attack}) &= 1 - [1 - \mathbf{Pr}(\text{a successful one-shot attack})]^q \\ &\approx q \times \mathbf{Pr}(\text{a successful one-shot attack}) \end{aligned}$$

This model is rather conservative because it is only valid when the intruder launches 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 an implementation is reliably constructed. This means 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 $2^{1-b} = \sum_{l=b}^{\infty} 2^{-l}$, where 2^{-l} is the likelihood of a successful one-shot attack on an optimal implementation of a l-bit

[6] This is similar to the goal of password-based authentication protocols discussed in the first chapter.

[7] Separate security analysis will be provided whenever protocols use weak empirical channels $\longrightarrow_{WE}$ to transmit SASs, which can be stalled and then replayed in other protocol runs.

SAS protocol, i.e. we will formally define optimality in human interaction of this type of protocol in Definition 13 of Section 2.5.1.

Since the protocol design can informally 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.

Since the above analysis only can give informal evidence of security, to make clear which protocols possess proofs of security (presented in papers where they were introduced), and which only have informal evidence of security, we will add this information into box-descriptions of each protocol, i.e. either '(Proof of security)' or '(Informal evidence of security)'. Note, although some protocols have security proofs, they are not really complete since important assumptions have been made, or only certain type of attacks such as one-shot, impersonation and substitution attacks are considered. We therefore will add footnotes pointing out limitations in proofs where we know they exist.

We are also interested in *chosen plain-text attacks* [105] under which the intruder can influence the information trustworthy parties want to authenticate. Even though this attack might seem unrealistic, it is desirable that protocols are immune to it, i.e. it will become useful when we analyse the protocol of Balfanz et al. in Chapter 5. Since the attack relies on searchings, we refer to it as a special case of the powerful 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, everyone has to be trustworthy in a key agreement protocol, whether it is a pairwise or group scheme. More discussion about this issue can be found at the beginning of Chapter 4.

2.5 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 computational processing required at the nodes. The following models for human effort and computation cost are adapted from three of our papers [78, 79, 80].

2.5.1 Human effort

Our main measure of empirical work is the number of bits of the short authentication strings that each user has to compare, handle, remember or manually transfer or copy. 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 is required to compare information and the degree of certainty that human users will *actually* do the work required of them.

At one extreme, we can imagine one of the two parties announcing the final digest and asking the other humans present to put up their hands if the value displayed on their PDAs 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 latter issues are implementation dependent, and orthogonal to the logical structure of the underlying protocol, and thus we will stick to our simple measure. Furthermore, abstracting away from practical implementation of human interaction will allow us to quantify human work in terms of bits, and therefore formally analyse the optimality of human effort relative to the security obtained.

Throughout this thesis, we always attempt to optimise the amount of security one can obtain from a given amount of empirical (human) communication. The following defines what it means for a protocol in this area to optimise human interactions.

Definition 13 [78, 79, 80] A protocol using short authentication strings (SASs) is said to be optimal in human interactions iff there is only a single b-bit SAS that needs to be empirically communicated, and the probability of a successful one-shot attack is upper bounded by 2^{-b}.

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

2.5.2 Computation cost 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 might be limited. Examples of lightweight devices are mobile phones, smart or credit cards, and medical sensors which are quite capable in terms of processing power. In addition, they also can indicate successful or unsuccessful completion of a protocol run and provide means for the input and output of a short string of symbols.

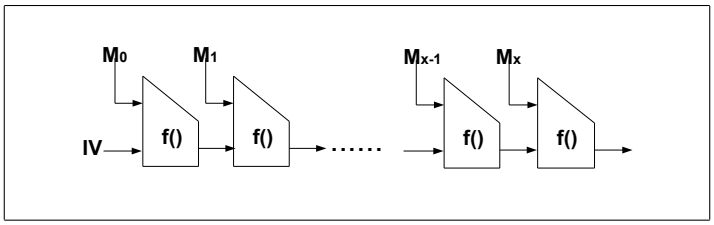


Figure 2.1: The Merkle-Damgård structure of conventional long-output cryptographic hash functions. Here $f()$ is a one-way and collision-resistant compression function, IV is some public initial value, and the input message m is partitioned into multiple fixed-size blocks $M_0, ..., M_x$, which are processed sequentially.

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 $shorthash()$, 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 [36, 79, 119, 123] 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.

For simplicity, we only look at the Merkle-Damgård construction based hash functions [74] (i.e. block cipher based and customised functions such as SHA-1, MD5, Davies-Meyer, Matyas-Meyer-Oseas and Miyaguchi-Preneel) because they are provably secure given that a one-way and collision-resistant compression function exists, and also this family of hash functions is widely used in practice.

It is clear from Figure 2.1 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-1, are *iterative* in nature. They are computed by calling a "compression" function once for every (512-bit) block in sequence.

It also seems clear that the cost will increase at least linearly with the output length b. Considering the type of operation of the Merkle-Damgard construction reveals that it always has an *internal state* whose bitlength is equal to or greater than the output bitlength. The internal state is updated by linear or bitwise operators (e.g. Shifting, AND, OR, XOR and rotation) in each loop of the algorithms to ensure that there is a computation between each message input bit and each bit of the internal state, i.e. the strict avalanche criterion. This implies that the computational cost is proportional to the size of the internal state, and a simple cost model of a b-bit hash function, which we will adopt in

the computation analysis of hash functions, might be:

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

Since well-known hash algorithms tend to be fixed width and vary significantly in their individual costs, it is hard to be too definite about this rule.[8] Although the computational cost model does not take into account the number of clock cycles and implementation-specific issues (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 2.1.

Regarding the cost of computing a digest function $digest(\_,\_)$ which is a family of short hash functions indexed by a key k as defined in Section 2.3.3. Even though the key bitlength might be significantly longer than the digest and hash output in several constructions of universal hash functions invented to date [54, 55], it normally does not play any significant part in the computation. Consequently, key length will not have a big impact on the computation cost.[9] 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 2.3.1 inputs a message $INFO$ 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.[10] 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 of similar length. The latter is true because commitment schemes are normally built from pseudorandom functions such as hash functions [89].

Table 2.1 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

[8] In 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: $HASH(m) = hash(1, m) \parallel hash(2, m) \parallel \ldots \parallel hash(t, m)$. This of course clearly follows our computational cost model.

[9] In fact the longer is the key, the fewer the number of pseudorandom bits we have to generate in our proposed construction of digest functions as well as universal hash constructions of Krawczyk [54, 55], and subsequently the better.

[10] In practice, a commitment scheme, such as the one introduced in Section 2.3.1, 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: $longhash(INFO||R)$. However, it has been noted that $longhash(INFO||R)$ could be replaced with 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.

Cryptographic primitive	Computation cost
$longhash(INFO)$ or $hash_{160}(INFO)$	$WM = 5M$
$longhash(k)$ or $hash_{160}(k)$	$WW = 25$
$hash(INFO)$ or $hash_{80}(INFO)$	$WM/2$
$shorthash(INFO)$ or $hash_b(INFO)$	M
$digest(INFO)$	M
$commit(INFO, R)$	$WM = 5M$
$open(INFO, c, d)$	$WM = 5M$

Table 2.1: 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.

doing is ignoring the multiplicative constant because *all* the computational costs come from the same model. 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.

Every protocol in this thesis, except the first one in Section 4.2, only uses long or short hash functions, 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. At the end of the following three chapters, discussing pairwise, group and one-way authentication protocols, all of the computational costs and human effort will be put into tables summarising the efficiency of each class of protocols.

Chapter 3

Pairwise authentication protocols

A considerable volume of previous work addresses the problem of bootstrapping security from the actions and interaction of humans. However, as we will discuss in this chapter, the majority of the prior art has concentrated on the problem of achieving mutual authentication in a peer to peer (P2P) network. The reason for this might be that P2P applications have been popular and dominant in practice, e.g. secure payments, including CHIP&PIN technology, and online and telephone banking. Another source of many applications is in healthcare informatics, *telemedicine* or *virtual healthcare systems* where sensitive data associated with patients with long-term conditions (e.g. diabetes and asthma) are collected via cellphones from wearable devices on patients. Healthcare applications of this latter type have been developed by T^+ Medical [5].

In this chapter, we look at a variety of pairwise interactive authentication protocols, where parties A and B want to authenticate some public information, such as Diffie-Hellman keys or $INFO$, to each other. Some of this work has previously been published [80]. Our main contributions in this chapter are the introduction of improved versions for protocols due to Hoepman [49], and Wong and Stajano [122, 123], as well as the introduction of two information binding strategies governing the design of all protocols introduced to date.

We show that, in order to optimise (i.e. minimise) the amount of human or empirical work, it is better to handle a single short authentication string (SAS) rather than a multiplicity of such strings, as used in certain other protocols, for example, those of Wong and Stajano, and Hoepman. The work leads us to propose improved versions of both the Wong-Stajano and Hoepman protocols in Sections 3.2.1 and 3.3, which achieve the same level of security with only half the amount of human interactions. 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.

We show that the significance of the idea of *joint commitment without knowledge*, introduced in Section 2.3.1, in providing the same level of security for all protocols presented in this section, i.e. the probability of a successful one-shot attack[1] is upper bounded by 2^{-b}, where b is the bitlength of the SASs. For this reason, in all the pairwise and group protocols (except the Hoepman and Wong-Stajano schemes in Section 3.1, and HCBK in Section 4.2), the value of the unique SAS is jointly committed to by every protocol participant. This therefore leads us to introduce the following two information binding strategies that help us achieve (joint) commitment without knowledge as well as classifying the many protocols considered in this thesis.

We first specify what it means when one bit string is independent of the other bit string, since the two information binding strategies make use of the concept.

Definition 14 A bit string X is independent of a bit string Y if for all random variable X of value x and for all random variable Y of value y: $\mathbf{Pr}[X = x] = \mathbf{Pr}[X = x \mid Y = y]$

Definition 15 [80] A protocol using a SAS is said to achieve *indirect information binding* if the SAS, jointly committed by every node, is *independent* of the information $INFOS$ parties want to authenticate.

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

Definition 16 [80] A protocol using a SAS is said to achieve *direct information binding* if the SAS, jointly committed by every node, is *dependent* on the information $INFOS$ parties want to authenticate.

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

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 the 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

[1] More information about the intruder's power and different types of attack can be found in Section 2.4.

hash function. The computational efficiency pay-off of direct binding strategy will be illustrated each time a protocol with direct binding is introduced, i.e. we will compare its efficiency explicitly with previous related protocols using indirect binding. The advantages will be demonstrated clearly when the cost of all schemes are gathered in a single table at the end of this chapter.

It should be noted that any of the group protocols presented in Chapter 4 can be used to create a group of size 2, and do so as efficiently as the pairwise schemes in Sections 3.2.1 and 3.3. Moreover, one role of these pairwise and group schemes is one-way communication where one party wants to authenticate information to the other node via an asymmetrical empirical channel, as can be illustrated in the protocol of Vaudenay in Section 3.2.1.

3.1 Multiple empirical short authentication strings

We describe two pairwise authentication protocols, the first due to Hoepman [48, 49] and the second to Wong and Stajano [122, 123]. 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 [49] defines SASs as the outputs of a b-bit (short) hash function $shorthash()$, as mentioned in Section 2.3.3. In addition, the Diffie-Hellman tokens $g^{x_{A/B}}$ play the role of both $INFO_{A/B}$ and long fresh random nonces, and so must be unpredictable and fresh at each session.

Hoepman pairwise protocol [49] (Incomplete proof of security)		
1.	A	$\longrightarrow_N B : longhash(g^{x_A})$
1′.	B	$\longrightarrow_N A : longhash(g^{x_B})$
		Where x_Y is a long random nonce of Y
2.	A	$\longrightarrow_E B : shorthash(g^{x_A})$
2′.	B	$\longrightarrow_E A : shorthash(g^{x_B})$
3.	A	$\longrightarrow_N B : g^{x_A}$
3′.	B	$\longrightarrow_N A : g^{x_B}$
		A and B verify the long and short hashes.
		A and B then share the key $k = g^{x_A x_B}$
4.	A	$\longrightarrow_N B : longhash(g^{x_A x_B})$
4′.	B	$\longrightarrow_N A : longhash(g^{x_B x_A})$
Computational cost: $2(WM + M) = 12M$		

This protocol offers good security, i.e. the probability of a successful one-shot attack is bounded by 2^{-b},[2] despite the use of b-bit shorthash functions 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.[3] We will see an example of what goes wrong without *commitment without knowledge* at the start of Section 4.2, which discusses group protocols.

We assume that M is the word-length of the Diffie-Hellman tokens[4] (g^{x_A} and g^{x_B}). Since Messages 4 and 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 2.5.2, the computation cost of the Hoepman scheme is of order $2(WM + M) = 12M$, where W and 1 are the output word-lengths of long and respectively short hash functions.

Taking a different approach, Wong and Stajano [122, 123] 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.

[2] Although a security proof in the Bellare-Rogaway model for this protocol was provided in [49], many formal details supporting claims in the proof are missing. For example, the claim 'success with one instance is independent of success in any other instance' made at the end of the proof (Theorem 3.5 in page 222) which aims to lift the security analysis of a one-shot attack to a multiple-shot attack needs a proof.

[3] In [48], 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 $shorthash(X)$ sent over the empirical channel in the second message. In this circumstance, A only sends out the shorthash iff he receives the 1-bit commitment empirical signal from B at the first place and vice versa. Furthermore, these acknowledgement signals must be transmitted over the empirical channel because they must not be blocked or delayed by the intruder. In this version, A does not need to know the identity of B during Messages 1, so Hoepman refers to it as the *anonymous* case, whereas the protocol above applies to the *non-anonymous* case.

[4] As 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 [122]		
(Informal evidence of security)		
1.	A	$\longrightarrow_N B : g^{x_A}$
1′.	B	$\longrightarrow_N A : g^{x_B}$
2.	A	$\longrightarrow_N B : longhash(A, g^{x_A}, g^{x_B}, R_A, K_A)$
2′.	B	$\longrightarrow_N A : longhash(B, g^{x_B}, g^{x_A}, R_B, K_B)$
		R_Y and K_Y are short and long random nonces of Y
3.	A	$\longrightarrow_E B : R_A$
3′.	B	$\longrightarrow_E A : R_B$
4.	A	$\longrightarrow_N B : K_A$
4′.	B	$\longrightarrow_N A : K_B$
		A and B verify the longhashes.
Computational cost: $2W(2M + W) = 20M + 50$		

The security of this protocol comes from the intruder's inability to invert the longhashes, or to predict the non-determinism introduced by the pair of nonces (R_X, K_X) at the point to which these are committed. As in the Hoepman scheme, 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 the Hoepman scheme ($12M$).

We now make two observations about the structure of this protocol. The high cost of computing *longhash* (due to a long input $(A, g^{x_A}, g^{x_B}, R_A, K_A) : 2M + W$ 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 weakness has been independently found and corrected by the inventors in their revised version of the paper, published in October 2007 [123]. 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 8.

Furthermore, both the Hoepman and Wong-Stajano protocols are *suboptimal* in the amount of

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

We end this section with a crucial observation: the Hoepman scheme chooses to bind Diffie-Hellman tokens directly to the SASs. This is not the case in the Wong-Stajano scheme, which is therefore more expensive in computational cost than Hoepman's scheme. By this we mean that the $INFO$s they are trying to authenticate are used directly in the evaluation of the empirically compared strings in the Hoepman protocol, while those compared in the Wong-Stajano protocol are not. These two different strategies are termed *direct* and *indirect* bindings as were defined in Definitions 16 and 15, 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 (Definition 15), the SASs to which parties are jointly committed are functionally independent of the information they want to authenticate. This is the idea we have briefly seen in the Wong-Stajano scheme, and it appears in many other schemes proposed in the literature [7, 22, 58, 59, 86, 119, 122, 123]. We will analyse these here. What distinguishes all of these from the Wong-Stajano scheme is a single SAS, which is required to be compared over the empirical channel, as opposed to multiple ones.

While the SASs and the data $INFOS$ that parties want to authenticate are independent, the security of the protocols comes from some mechanism binding random nonces, which are instrumental in the computation of SASs, and $INFOS$ together in a secure way. Thus there is a tendency to use commitment schemes (described in Section 2.3.1) 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 [119] and Čagalj et al. [22] 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 [119], 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 [119] (Proof of security)	
1. A	$\longrightarrow_N B : INFO_A, c$
	Where $c \parallel d = commit(INFO_A, R_A)$,
	R_A is a short random nonce of A.
2. B	$\longrightarrow_N A : R_B$
3. A	$\longrightarrow_N B : d$
	B computes $R_A = open(INFO_A, c, d)$
4. A	$\longrightarrow_{WE} B : R_A \oplus R_B$
	B verifies the correctness of $R_A \oplus R_B$
Computational cost: $MW = 5M$	

The protocol delivers the guarantee of authenticity and integrity of $INFO_A$, and even with a single b-bit SAS the probability of a successful one-shot attack is still upper 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. With q concurrent runs of A and B, the number of protocol sessions from the intruder's view will become q^2. Thus, the chance of a successful q-shot attack is upper bounded by $1 - (1 - 2^{-b})^{q^2} \approx 2^{-b}q^2$ as pointed out by Vaudenay [119].[5]

We, however, want to point out that the origin of data transmitted over weak empirical channels cannot be forged, and to take advantage of the replayability of SASs the intruder will have to launch attacks on the *same* pair of parties who are responsible for delivering the SASs, i.e. party A in this

[5] As pointed out in Footnote 3 of Chapter 2 there is a lack of explicitness in the specification of the commitment scheme defined by Vaudenay in [119], since the security specification there fails to bind it to the message input, as was obviously intended. This flaw might well affect the accuracy of Vaudenay's proof of his one-way authentication protocol. Another small problem is that in order to make the proof work in a concurrent setting (Lemma 6 of [119]), Vaudenay implicitly assumed that all protocol instances are independent.

case. The above security analysis is therefore only valid with a small value of q because humans are highly sensitive to delays, i.e. they will quickly become aware that an attack is taking place, and so stop any attempt of running the protocol again as pointed out in Section 2.4. In contrast, if we replace $\longrightarrow_{WE}$ with $\longrightarrow_{E}$ then a SAS cannot be delayed from one to later runs, and so to have a fair chance of a successful attack, the intruder needs to run 2^b concurrent runs of (perhaps different) pairs of parties: a 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.[6]

Although Vaudenay's scheme halves the amount of empirical communication relative to those of Hoepman and Wong-Stajano, it only provides one-way authentication, representing one role of pairwise schemes in this chapter. 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 [119], fortunately makes the protocol structure completely symmetrical. It is essentially the same as another protocol which is termed DH-SC and invented by Čagalj, Čapkun and Hubaux [22].

Čagalj-Čapkun-Hubaux two-way authentication protocol [22] (Informal evidence of security)		
1.	A	$\longrightarrow_N B : INFO_A, c_A$
1′.	B	$\longrightarrow_N A : INFO_B, c_B$
		Where $c_Y \parallel d_Y = commit(Y, INFO_Y, R_Y)$,
		R_Y is a short random nonce of Y.
2.	A	$\longrightarrow_N B : d_A$
2′.	B	$\longrightarrow_N A : d_B$
		Y' computes $R_Y = open(Y, INFO_Y, c_Y, d_Y)$
3.	A	$\longleftrightarrow_E B : R_A \oplus R_B$
Computational cost: $2WM = 10M$		

Both of the above protocols use the *joint commitment without knowledge* idea to precommit two parties to the XOR of some random short secrets or nonces. This is achieved by parties outputting

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

their shares of the commitment of their random short secrets to each other in the first messages.

We note this scheme can be regarded as an upgraded version of the Wong-Stajano protocol (Section 3.1) in two ways. Firstly, the two initial messages in the Wong-Stajano protocol have been successfully eliminated. This is based on the grounds that each node A only needs to commit to its $INFO_A$ at the beginning, so he has not to acquire $INFO_B$ at the time of computing the commitment. Secondly, the order of releasing the SAS and the decommitments has been reversed relative to the Wong-Stajano protocol.[7] As a consequence, parties only need to manually compare a single SAS, which is the XOR of short nonces R_A and R_B implicitly derived from the decommitments.

The same technique can also be used to improve the human and processing cost of the Wong-Stajano scheme.[8]

Regarding computation cost, the two commitments would double the cost of the Vaudenay scheme to an order of $2WM = 10M$. On the other hand, if we quantify the cost relative to the amount of information authenticated then those of Vaudenay and Čagalj-Čapkun-Hubaux will equal each other. The result illustrates the gain in efficiency of these in comparison with those of 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 4.3.3.

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

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

[8] The idea of eliminating the first two messages carrying $g^{x_{A/B}}$, removing the long random nonces as well as reducing the number of different SASs to a single one of b bits in the Wong-Stajano scheme can be demonstrated by our revised scheme. We note: (1) the scheme has a similar form to Čagalj-Čapkun-Hubaux; and (2) the Diffie-Hellman tokens play the dual role of both long random keys (or the nondeterministic and random element of the commitment scheme) and $INFO$s. The improved version therefore achieves the same level of security as the Wong-Stajano protocol of Section 3.1, i.e. the probability of a successful one-shot attack is upper bounded by 2^{-b}.

Improved version of Wong-Stajano *New* **[80]** (Informal evidence of security)			
1.	A	$\longrightarrow_N B$	$: longhash(A, g^{x_A}, R_A)$
1′.	B	$\longrightarrow_N A$	$: longhash(B, g^{x_B}, R_B)$
2.	A	$\longrightarrow_N B$	$: R_A \| \| g^{x_A}$
2′.	B	$\longrightarrow_N A$	$: R_B \| \| g^{x_B}$
3.	A	$\longleftrightarrow_E B$	$: R_A \oplus R_B$
Computational cost: $2W(1+M) = 10M + 10$			

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

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 [86] 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 [86]* (Proof of security)	
1. A	$\longrightarrow_N B : INFO_A, c$ Where $c \parallel d = commit(INFO_A, k_A)$ k_A is a long random nonce of A
2. B	$\longrightarrow_N A : INFO_B, R_B$ Where R_B is a b-bit random nonce of B.
3. A	$\longrightarrow_N B : d$ B computes $k_A = open(INFO_A, c, d)$
4. A	$\longleftrightarrow_E B : R_B \oplus digest(k_A, INFO_B)$
Computational cost: $WM + M = 6M$	

Though the SAS $= R_B \oplus digest(k_A, INFO_B)$ depends functionally on $INFO_B$, it is independent of $INFO_A$. This observation makes the scheme stand as a hybrid of direct- and indirect-binding protocols. Interestingly, the hybrid strategy is also reflected by the differences in the 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 random attacks and so can be short, whereas k_A is not and so has to be long; the two influence the final SAS 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,[9] which is significantly cheaper than Čagalj-Čapkun-Hubaux ($10M$), i.e. the fully indirect binding scheme.

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

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

that we have already studied falls into this category, but is not optimal in the human work. In this section, we will discuss a number of other pairwise protocols which are optimal in this respect.

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 [78, 79].

P1 [78, 79] 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.

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 increasing 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 [119] of 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 searching.[10] On the other hand, the security analysis of the direct binding schemes is similar to indirect binding ones provided the digest function is ideal, as specified in Definition 9 of Section 2.3.3. 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 the Bluetooth whitepaper [7] and whose proof of security is presented by Lindell [61]; here k_A and k_B are long fresh sub-keys generated by A and B.

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

Bluetooth 2 [7, 61]		
(Proof of security)		
1.	A	$\longrightarrow_N B : INFO_A$
1′.	B	$\longrightarrow_N A : INFO_B$
2.	B	$\longrightarrow_N A : longhash(INFOS, k_B)$
3.	A	$\longrightarrow_N B : k_A$
3′.	B	$\longrightarrow_N A : k_B$
		k_Y is a long fresh key of Y
4.	A	$\longleftrightarrow_E B : trunc_b(longhash(f(k_A, k_B, INFOS)))$
Computational cost: $W(2M + W) + 2W(M + W) = 20M + 75$		

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

The inefficiency in using a truncated hash function[11] 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 related protocols using indirect binding (Čagalj-Čapkun-Hubaux: $10M$, and Pasini-Vaudenay: $6M$) and the two following schemes, as it is redundant to bind $INFOS$ and sub-keys together by using both longhash function in Message 2 and in the SAS. Either of them would be sufficient for the obtained security.

Removing this unnecessary binding in Message 2 of Bluetooth 2 can increase its computational efficiency as well as simplicity (eliminating Messages 1 and 1'), since B does not need to know $INFO_A$ (and $INFOS$) at the point when he is committed to k_B. This is what was proposed in [58, 59] by Laur and Nyberg:

[11] $trunc_b()$ takes the first b bits of its input.

Laur-Nyberg pairwise protocol [58, 59]		
(Proof of security for a one-shot attack)		
1.	A	$\longrightarrow_N B : INFO_A, c$
		Where $c \parallel d = commit(k_A)$
2.	B	$\longrightarrow_N A : INFO_B, k_B$
3.	A	$\longrightarrow_N B : d$
		B computes $k_A = open(c, d)$
4.	A	$\longleftrightarrow_E B : h_{k^*}(INFOS)$
		Here $k^* = g(k_A, k_B)$
Computational cost: $2WM = 10M$		

Here $h_{k^*}()$ is a universal hash function [110] of the appropriate length, i.e. b-bit in this case, whose specification is closely related to a digest as discussed in Sections 2.3.2 and 2.3.3. The impact of removing redundancy can be seen in the decline of the computation cost of this protocol: $W^2+2WM = 25 + 10M$.[12]

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

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

After this transformation and a replacement of a commitment scheme with a *longhash*() function, the protocol becomes similar to the following. This was discovered independently by Laur and Nyberg [58, 59], and the author in 2006 when we combined ideas used in our SHCBK protocol (see Section 4.2) and Vaudenay's protocols (see Section 3.2.1).

[12] We choose to ignore the cost of computing $g()$ to combine sub-keys in this calculation, since it is negligible relative to the computation of a universal hash function, which involves applying a cryptographic hash function to the $2M$-word input message $INFOS$ in the first place, as specified by Laur and Nyberg in [58, 59], which results in a cost of $2WM = 10M$.

Pairwise authentication scheme in Vaudenay's style *New* **[58, 59, 80]** (Informal evidence of security)
$1.A \quad \longrightarrow_N B : INFO_A, longhash(k_A)$
$2.B \quad \longrightarrow_N A : INFO_B, k_B$
$3.A \quad \longrightarrow_N B : k_A$
$4.A \quad \longleftrightarrow_E B : digest(k_A \oplus k_B, INFOS)$
Computational cost: $W^2 + 2M = 2M + 25$

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

Improved Hoepman *New* **[80]** (Informal evidence of security)
$1.A \quad \longrightarrow_N B : longhash(A, g^{x_A})$
$2.B \quad \longrightarrow_N A : g^{x_B}$
$3.A \quad \longrightarrow_N B : g^{x_A}$
$4.A \quad \longleftrightarrow_E B : shorthash(g^{x_A} \oplus g^{x_B})$
Computational cost: $WM + M = 6M$

The main difference between this and the previous schemes considered in this chapter 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 or the SAS to be jointly committed without knowledge at the end of Message 2 by both parties, the Diffie-Hellman tokens must be fresh, random and unpredictable at each session.[13] Also because of this, the digest function (2-input function) can be replaced with a single input short hash function $shorthash()$; though the combination of this and the exponentiation of Diffie-Hellman needs to satisfy a specification similar to that of the digest function and the randomising effect of XOR in combining k_As.

In comparison with the Hoepman scheme, this requires a single SAS halving the human work. As in previous protocols, the computation of one longhash and one short hash results in a cost of $WM+M = 6M$: exactly a half of the Hoepman scheme ($12M$) (direct binding, $12M$) and significantly lower than the improved version of the Wong-Stajano scheme (the related protocol using indirect binding, $10M + 10$) of Footnote 8.

[13] It 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) [80] (Informal evidence of security)
1.$A \longrightarrow_N B : longhash(A, g^{x_A})$
2.$B \longrightarrow_N A : g^{x_B}$
3.$A \longrightarrow_N B : g^{x_A}$
4.$A \longrightarrow_E B : shorthash(g^{x_A} \oplus g^{x_B})$
5.$B \longrightarrow_E A$: 1-bit committed signal
Computational cost: $WM + M = 6M$

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

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

Wong and Stajano [123] 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), [123] (Informal evidence of security)
1.$A \longrightarrow_N B : g^{x_A}$
2.$B \longrightarrow_N A : B, g^{x_B}, MAC_{K_B}(B, g^{x_A}, g^{x_B}, R_B)$
R_B and K_B are short and long random nonces of B
3.$A \longrightarrow_E B$: 1-bit committed signal
4.$B \longrightarrow_E A : R_B$
5.$B \longrightarrow_N A : K_B$
6.$A \longrightarrow_E B$: 1-bit committed signal
Computational cost: $W(2M + 1) = 10M + 5$

We note that the order of Messages 4 and 5 can be interchanged. The pair of Messages 4 and 6 results in symmetric agreement on R_B: in fact they are just an implementation of "$A \longleftrightarrow_E B : R_B$".

It seems unlikely that the computation cost of the cheapest of these protocols can be reduced much further. It also seems clear that some sort of cryptographic binding of $INFOS$ to the empirical message is necessary, and our assumed model of the digest function appears to be a lower bound on that

Protocol	Binding	Human work(bit)	Computation cost	Proof of Security
Hoepman	Direct	$2b = 32$	$2(WM+M)=12M$	Incomplete
Improved Hoepman	Direct	b =16	$M + WM = 6M$	No
Improved Hoepman$'$ (one-way empirical)	Direct	$b+1$=17	$M + WM = 6M$	No
Wong-Stajano (one-way empirical)	Direct	$b+1$=17	$2WM = 10M$	No
Wong-Stajano	Indirect	$2b$=32	$2W(2M+W) = 20M + 50$	No
Improved Wong-Stajano	Indirect	b =16	$2W(1 + M) = 10M + 10$	No
Vaudenay ($\rightarrow_{WE}$)	Indirect	b =16	$WM = 5M$	Yes
Čagalj-Čapkun-Hubaux	Indirect	b =16	$2WM = 10M$	No
Pasini-Vaudenay (*longhash*)	Hybrid	b =16	$WM + WM = 10M$	Yes
Pasini-Vaudenay (*digest*)	Hybrid	b =16	$WM + M = 6M$	No
Bluetooth 2 (*longhash*)	Direct	b =16	$W(2M+W)+2W(M + W)=20M+75$	Yes
Bluetooth 2 (*digest*)	Direct	b =16	$W(2M+W)+2M=12M+25$	No
Laur-Nyberg (*longhash*)	Direct	b =16	$W^2 + 2WM = 10M + 25$	Yes
Laur-Nyberg (*digest*)	Direct	b =16	$W^2 + 2M = 2M + 25$	No
Vaudenay-style (*digest*)	Direct	b =16	$W^2 + 2M = 2M + 25$	No

Table 3.1: Interactive pairwise two-way authentication protocols (unless indicated they all use two-way empirical channels: $\longleftrightarrow_E$)

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 SAS and being committed to before any node knows it. This has to be done with strong cryptography, and the hash used in, for example, the Laur-Nyberg scheme appears to be as efficient as possible at doing this.

Another observation we want to make is that 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 powerful and random attacks separately: a manifestation of the principle of *separation of security concerns*, which will be analysed in Chapter 5.

3.4 Efficiency of pairwise authentication protocols

In this section, we tabulate the efficiency of all 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. In this and other tables, we have used the simple cost model

of hash and digest functions described in Section 2.5.2: the cost is proportional to the product of the length of the information being digested and the width of the output.

There is a relationship between how much we assume of the empirical channel and how much work is required over it. Of course, one might want to change the values of any of these parameters for good reason, but we believe that the relative differences between them will not be significant 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 presented in later sections. In those protocols that require the extra confirmation message over the empirical channel (Improved Hoepman, Wong-Stajano etc), we write '$b + 1$' as the amount of empirical effort.

As can be seen from the table, except the multiple-SAS protocols of Hoepman and Wong-Stajano, all other schemes which only require a single b-bit SAS are only susceptible to a one-shot attacker with probability 2^{-b}. These protocols are therefore optimal in human effort.

It is also necessary to point out that, since most direct binding protocols invented by other authors (e.g. Pasini-Vaudenay, Bluetooth 2 and Laur-Nyberg) use a truncated longhash function to compute the SASs instead of a *digest function*, we will therefore illustrate the difference by giving the computation cost of both cases in this and other tables. The truncated longhash or universal hash functions ([110] requiring a longhash to compress large messages into a fixed number of bits initially, see Chapter 6 for more information) will be denoted *longhash*.

In general, direct binding protocols are more efficient than indirect binding ones: up to $B/w = 5$ times more efficient should M get large, thanks to the use of a short output digest function.[14] The larger M (the word length of $INFO$) is, the more accurate this effect.[15] Thus the advantage of direct binding will become crucial in applications which involve authentication of a large amount of data as in telemedicine (e.g. medical data) and teleconference.

The difference in computational cost between direct and indirect bindings is also reflected if we consider the *hybrid binding* protocol of Pasini-Vaudenay whose computation cost ($6M$) is the average of Čagalj-Čapkun-Hubaux (10M) and Laur-Nyberg or Vaudenay-style ($2M + 25$).

Protocols which possess proofs of security (or incomplete proofs) in either the Random Oracle Model or the Bellare-Rogaway (or the standard) Model are marked 'Yes' (or 'Incomplete') in Table 3.1 as well as subsequent tables summarising group and one-way authentication protocols. Although their

[14] In Section 2.5.2 the digest output length is rounded up to 1 word (32 bits). However, if we have a ($w/2 = 16$)-bit digest and a 8- or 16- bit processor, which will often be the case in lightweight devices, then the advantage of digest over hashing, i.e. direct over indirect bindings, potentially grows to 10 times $= \frac{B}{w/2}$ (from 5).

[15] This is not necessarily a clear advantage for direct binding in the case of the Hoepman and Wong-Stajano schemes, because the information, parties want to have authenticated, consists of one or two Diffie-Hellman tokens that are short compared to $INFO_A$.

security proofs have been published in formal proceedings, we note that some have limitations as have been pointed out when we describe the protocols. On the other hand, protocols which do not possess proofs of security are marked 'No' in the table, this however does not mean these schemes are insecure because we have also given informal evidence for the security of all of them here.

3.5 Conclusions

In this chapter we have considered pairwise authentication protocols using short authentication strings. Our first contribution is the introduction of improved versions of both the Hoepman and Wong-Stajano schemes which can halve both human effort and computational cost. These improved versions are believed to be as secure as the original protocols, although we only give informal evidence of security in this chapter.

Secondly, the security of all the protocols considered here has been found to be shown to depend on the idea of *joint commitment without knowledge*, which can be achieved by either direct information binding, indirect information binding or a combination or a hybrid of the first two strategies. We subsequently categorised all protocols relative to the three types of information binding, and discover an interesting difference between the computational cost of these schemes.

Chapter 4

Group protocols

Much of the material in this chapter has previously been published [78, 79]. The main contribution of this chapter is the introduction of the protocol SHCBK (and certain variants) whose security does not rely on the existence of a trustworthy leader to achieve *(joint) commitment without knowledge*, as in the case for the HCBK scheme, introduced by Roscoe [96].

The majority of work done in bootstrapping security in pervasive computing to date has focused on pairwise applications in a peer-to-peer network, as pointed out at the beginning of Chapter 3. However, we believe there is a similar potential for bootstrapping security in larger groups as can be shown by the example given in the introduction of this thesis and quoted again here. 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 designing group authentication schemes in this area was initially done by Creese et al. [28, 29, 30, 31], and more recently by Valkonen et al. [118], resulting in several group protocols that optimise the number of human interactions. The main challenges of bootstrapping group security in pervasive computing are as follows.

There is a slightly grey area for protocols building groups of more than two. 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 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.

One of the first contributions in this area was that of Creese et al. [30] in which they show that the principle **P1** (i.e. *direct* information binding strategy) could be used to construct efficient and secure group protocols. Unfortunately, their first attempt resulted in a scheme that was found by Roscoe [78, 79, 96] to be vulnerable to a *man-in-the-middle* or birthday attack, due to the lack of *commitment without knowledge*. We will, therefore, explain why it is the case in Section 4.1, and then present improved protocols, most notably the protocol HCBK (Roscoe [96]) and our novel scheme SHCBK, in sections 4.2 and 4.3. In spite of the attack, the group protocol invented by Creese et al. introduced implicitly the principle **P1** that is formalised here. The principle, quoted again below for clarity, is important in the development of protocol design in this new area because it contributes to not only the optimisation of empirical work but also the reduction in computational cost.

P1 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.

We have already seen that the direct information binding strategy is significantly more efficient than the indirect one for pairwise protocols (i.e. both of these information binding strategies are instrumental to achieving *commitment without knowledge* as pointed out in Chapter 3). In this chapter, we will see the same is true for group protocols, namely the protocols 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 [119].

Like pairwise authentication schemes, group protocols could be used for one-way authentication, particularly in broadcast of information from a single node to a large number of audiences via an asymmetrical empirical channel. Typical examples of this are the HCBK protocol and the variant

described in sections 4.2 and 4.2.2, and the one-way authentication protocol of Vaudenay described in 3.2.1. The chief difference between these is that the HCBK protocol and its variant uses direct binding, as opposed to the use of indirect binding in the one-way authentication scheme of Vaudenay.

4.1 Man-in-the-middle attack on a group protocol

The following protocol introduced by Creese et al. [30] is probably the very first group authentication protocol in the area of pervasive computing.

Group protocol of Creese et al. [30] (Informal evidence of security)		
1.	$\forall A \longrightarrow_N \forall A'$	: A, Pk_A, T_A
2.	$\forall A \longrightarrow_N \forall A'$	: $\{\text{all Messages } 1, N_A\}_{Pk_{A'}}$
3a.	A displays	: $shorthash(\{\text{all Messages } 2^d\})$, number of processes
3b.	$\forall A \longrightarrow_E \forall A'$	: users compare hashes and check numbers
4.	$\forall A \longrightarrow_N \forall A'$	: $longhash(\{\text{all Messages } 2^d\})$

Here, $\forall A$ means that a message is sent to, or received by, all parties in the group **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, and T_A and N_A are A's fresh nonces. The superscripted expression 'all Messages 2^d' represents the concatenation of all the decrypted content of Messages 2 in alphabetical order, for example. In addition, Messages 4 do not add any extra security to the scheme; instead, their presence aims to provide a confirmation of the shared secret information.

The question with the protocol is whether it is vulnerable to a man-in-the-middle attack. Since the short hash value in Messages 3a is compared manually by humans, its length can only be up to a few digits or characters. It turns out that it is not hard for an intruder to search for a (b-bit) short hash collision that might cause the parties to agree on different (secret) information, as described by Roscoe [78, 79, 96]. The attack is described as follows.

The intruder partitions group **G** of N parties into two subsets S_1 and S_2, and runs two parallel sessions with each of them. During the two parallel runs, the intruder impersonates all parties of subset S_1, by modifying messages sent from subset S_1 with different nonces and public keys for which (s)he knows the corresponding private keys, to talk to parties in the other subset S_2, and vice versa. Conversely, there is no need to alter messages passed within each subset.

$$\forall A \in S_1 : I(A) \longleftrightarrow S_2$$

$$\forall B \in S_2 : I(B) \longleftrightarrow S_1$$

Therefore, everyone in the group **G** is still thinking that (s)he is running the protocol with the other $(N-1)$ parties. To launch a successful attack, the intruder has to search for new nonces, as illustrated in the following toy example.

Suppose that there are 5 parties running the protocol, where $S_1 = \{A, B, C\}$ and $S_2 = \{D, E\}$. Therefore, $\{N_A, N_B, N_C, N_D, N_E\}$ are the nonces created by all parties in Messages 2. The adversary will create shadow copies of these nodes A'–E' , as shown in Figure 4.1, and have to generate the corresponding fake nonces $\{N'_A, N'_B, N'_C, N'_D, N'_E\}$ for their Messages 2. It seeks to choose these values so that

$$shorthash(INFOS, N_A, N_B, N_C, N'_D, N'_E) = shorthash(INFOS', N'_A, N'_B, N'_C, N_D, N_E)$$

Here $INFOS$ and $INFOS'$ are the two different versions of the information that the intruder wants to fool subsets S_1 and S_2 into accepting at the end of the two parallel runs. These should be known to the intruder at the point it performs the following (birthday) search.

The intruder will pick the nonces N'_A, N'_B, N'_D randomly and then search for values of N'_C and N'_E such that the b-bit hashes come out to be the same. This search, shown in the figure, can be expected to take time proportional to $2^{b/2}$, thanks to the birthday paradox.

4.2 Asymmetrised group protocols

To avoid the man-in-the-middle or birthday attack, Roscoe [96] introduced the HCBK protocol, using the principle **P1** and the idea of *commitment without knowledge* to bind information to each party who wants to authenticate it.

We assume that there is one participant L in the protocol whom all agree is trustworthy. This could be because all participants are known to be trustworthy, because L has some special status amongst them, or because L is the only one requiring authentication. L will be called the 'leader', and the other nodes will be called 'slaves' (termed S).

In the following, A represents a typical node (either L or S). $init(L, A)$ is *true* if $L = A$, and *false* otherwise.

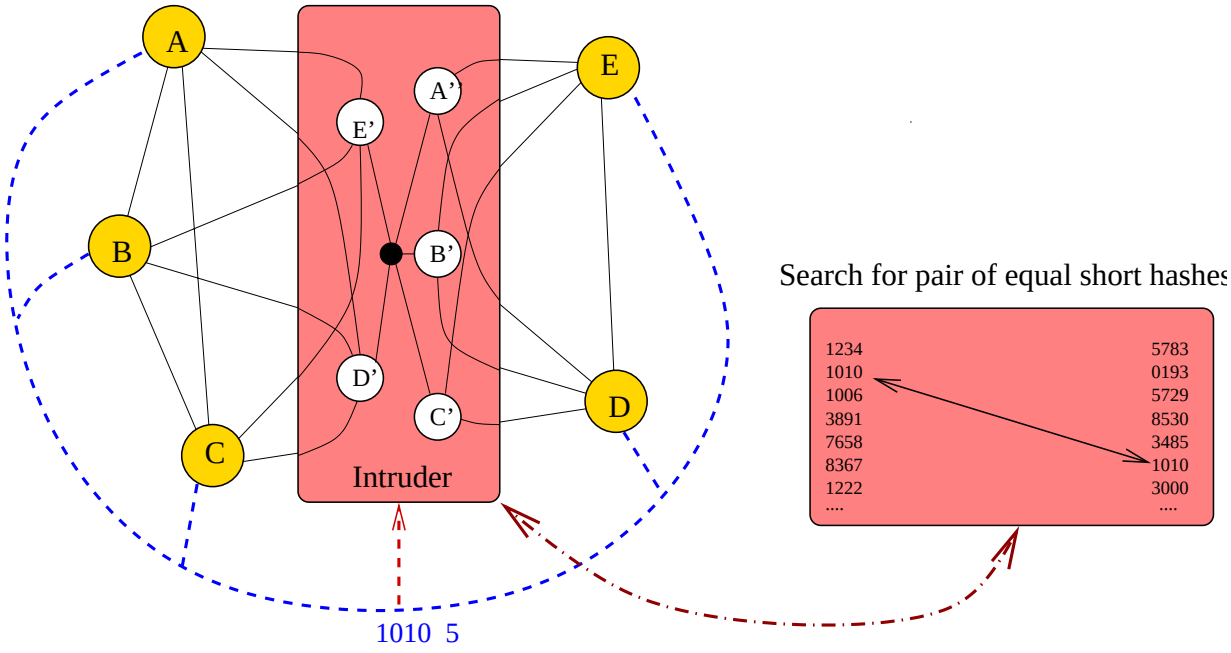


Figure 4.1: Birthday attack on a short hash, where the continuous and dashed lines indicate the Dolev-Yao and empirical channels, respectively. Number 5 besides the digest value '1010' indicates the number of participants that nodes need to agree empirically, as specified in the protocol.

Hash Commitment Before Knowledge, HCBK protocol [96] (Informal evidence of security)			
0.	L	$\longrightarrow_N \forall S$	$: L$
1.	$\forall A$	$\longrightarrow_N \forall A'$	$: (A, INFO_A)$
2a.	L	$\longrightarrow_N \forall S$	$: longhash(k_L)$
2b.	$\forall S$	$\longrightarrow_E L$	: 1-bit committed signal
3.	L	$\longrightarrow_N \forall S$	$: k_L$
4.	$\forall A$	$\longrightarrow_E \forall A'$	$: digest(k_L, INFOS), init(L, A)$
Computational cost: $W^2 + NM = 25 + NM$			

The meanings of these messages are as follows:

- **Message 1** publishes the information that all the nodes want to have attached to them via an insecure channel. Therefore, they do not know upon receiving it that it is accurate.
- **Message 2a** has L devise a key k_L with sufficient entropy that $longhash(k_L)$, which it publishes here, has no more than an infinitesimal likelihood of any searching attack on it succeeding.
- **Message 2b** has all the slaves communicate to L over the unforgeable empirical channel that they have received Message 2a. This can be achieved by a 1-bit signal over the empirical channel. Hence the slaves are *committed* to their final digest value (though none of them know it yet).
- **Message 3** has L publish the key k_L after it has received 1-bit commitments from all members of the group over the empirical channels. All slaves now have the duty to check if the values of Messages 2a and 3 are consistent.
- **Message 4** has them compute and compare the digest: this could be done either through the single point of contact at L or more generally. Once a node knows that all have agreed this value, it has completed the protocol and can enter group mode. It also guarantees that one of the nodes, doing the agreeing, has been playing the leader role thanks to $init(L, A)$ information.

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 $W^2 + NM = 25 + NM$, according to the cost model of Section 2.5.2.

4.2.1 HCBK protocol analysis

The security analysis of the HCBK protocol presented in this subsection builds on Roscoe [96] and jointly between Roscoe and the author [78, 79]. The protocol provides good security because every

slave is committed to the final view of k_L and the digest value before k_L is revealed in Message 3. Thus the idea of *commitment without knowledge* is applied since: (1) leader L must receive the 1-bit empirical signals from all of the slaves before revealing k_L, and (2) it is infeasible for the intruder to invert or constructively use and modify k_L after its longhash is sent out in Message 1. In other words, if an intruder launches a one-shot attack, then the probability of his or her success is upper bounded by 2^{-b}, where b is the digest bitlength.

However, in any real implementation of the protocol, the author would like to stress the significance of correct implementation of the empirical channels as defined in Section 2.1 to avoid false attacks, i.e. information transmitted over this channel must be neither blocked nor delayed from one to another session. Without this property, the protocol will be vulnerable to attacks, such as the following man-in-the-middle attack discovered by Roscoe in [78, 79].

In the kind of attack, the intruder aims to make some (but not all) nodes abandon a protocol run, and bring them to the point in a subsequent run where they are ready to agree the final digest with the rest of the group, whose run was not abandoned.

The attack is going to fail if the intruder abandons the leader's run because (s)he is the final determiner of its own digest value d_L by constructing k_L. So re-starting L would not give greater than 2^{-b} chance of achieving any particular value. Also, the leader expects to get empirical signals in Message 2b from the slaves, and these would not be available from the slaves whose runs were not abandoned.

On the other hand, if a slave S could be re-started after d_L was known, i.e. after k_L is released in Message 3, then the intruder could perform a search for a value k'_L yielding the same digest value d_L with a fake $INFO'_L$ he wants to persuade S. A potential attack discovered by Roscoe is open, provided that (1) the empirical Message 2b from S to L is either blocked or ignored; and (2) the digest sent from L over the empirical channel is delayed from one to a later protocol run.

We shall consider the situation where there are two nodes: the leader L and the non-leader B. $I(L)$ denotes that the intruder impersonates the leader to talk to other nodes, i.e. B in this case. In the first run α, the protocol looks like this:

$0.\alpha.$ $L \longrightarrow_N B$: L

$1.\alpha.$ $L \longrightarrow_N B$: $L, INFO_L$

$B \longrightarrow_N I(L)$: $B, INFO_B$

$I(B) \longrightarrow_N L$: $B, INFO'_B$

$2.a.\alpha.$ $L \longrightarrow_N B$: $longhash(k_L)$

$2.b.\alpha.$ $B \longrightarrow_E L$: B's committed signal

$3.\alpha.$ $L \longrightarrow_N I(B)$: k_L

$I(L) \longrightarrow_N B$: k''_L

As can be seen from run α, $INFO_B$ and k_L have been replaced with some fake $INFO'_B$ and random k''_L in Messages 1 and 3, respectively. This leads the protocol to the following three consequences:

- When B receives the fake key k''_L, B cannot verify the correctness of the longhash sent in Message 2. So B rejects it and aborts the run.
- In the meantime, the intruder obtains the original key k_L, and therefore can determine the final value d_L of the digest in this run.
- Also note that the leader L does not have any idea about the status of the current run, so (s)he just keeps waiting for Message 4 from B.

Meanwhile, the intruder starts a second run with B posing as L:

$0.\beta.$ $I(L) \longrightarrow_N B$: L

$1.\beta.$ $I(L) \longrightarrow_N B$: $L, INFO'_L$

$B \longrightarrow_N I(L)$: $B, INFO_B$

At this point, the intruder knows all the information (i.e. $\{INFO'_L, INFO_B\}$) that B will use in run β to compute the digest, except that run's key k'_L. Therefore, the intruder can search for a k'_L that will digest together with $\{INFO'_L, INFO_B\}$ to d_L. If it succeeds, it continues:

$2.a.\beta.$ $I(L) \longrightarrow_N B$: $longhash(k'_L)$

$2.b.\beta.$ $B \longrightarrow_E L$: The committed signal will be either blocked by I or ignored by L.

$3.\beta.$ $I(L) \longrightarrow_N B$: k'_L

The main feature in this run is that the 1-bit committed signal, sent over the empirical channel from B to L in Message $2.b.\beta$, must be either blocked by the intruder or ignored by the leader. This avoids the possibility that the trustworthy leader might spot this message and realise that an attack is taking place.[1]

After receiving k'_L from Message $3.\beta$, B will be able to check the correctness of the longhash sent in this run. The flaw of the protocol becomes apparent when B, thinking she is in run β, displays and compares the digest of the run β with the digest of L in run α over the empirical channel.

$$4.\beta. \quad B \longrightarrow_E L \quad : \quad digest(k'_L, \{INFO'_L, INFO_B\})$$
$$4.\alpha. \quad L \longrightarrow_E B \quad : \quad digest(k_L, \{INFO_L, INFO'_B\})$$

So in the end, L and B agree the equality of two digest values which have different antecedents. We therefore make the following specification for the implementation of the protocol:

> *The implementation must be designed so that agreement is impossible between final digest values other than those whose commitment has been signalled by the Messages* $2b$ *that* L *received.*

The most obvious way of achieving this is via timing limits: an upper bound on the time between L sending Message 3 and agreeing Message 4, and a lower bound between L receiving a Message $2b$ from S and S sending another Message $2b$.

Another possibility is to force each slave to receive both the key k_L and the digest value from leader L before it recomputes the longhash and the digest to check for equality. This means that a slave cannot abandon a protocol session in the middle of a run, and the digest value transmitted over the empirical channel will not be delayed from one to another session. The restriction does not, however, disallow a slave to have parallel runs with different leaders. However, it would be unfortunate to make a slave wait for a digest event when it would normally be necessary for it to abandon a run because it had received a mismatched k_L and $longhash(k_L)$, for example. One could propose some sort of intermediate solution where a slave always waits when there is a match between k_L and $longhash(k_L)$, and if a mismatch occurs it uses a time delay strategy, i.e. on receipt of a mismatch k_L and $longhash(k_L)$, each slave shall give a failure signal and shall implement a mechanism whereby it will refuse to start a new instance of the protocol for a short time period.

In the case where both a slave and a leader agree to run parallel sessions of the protocol concur-

[1] Since leader L is waiting for a proper digest sent over the empirical channel, (s)he can choose to ignore any one-bit empirical signal from the slave, then it would be unnecessary to block the 1-bit signal from slave B in run β.

rently, it should be reasonable to use run numbers included in empirical communications to distinguish the sessions. (Run numbers concatenated with the SAS or digest are sufficient.) Of course this would add to the empirical effort.

On the assumption that either of the above is achieved, we conclude that the nodes will never seek to agree final digest values to which they were committed later than the issue of Message 3 by L. Therefore, the protocol HCBK achieves its goal of limiting the chance of a successful man-in-the-middle attack to at most 2^{-b}.

4.2.2 Modified versions of HCBK

We will call the above protocol HCBK1, standing for *Hash Commitment Before Knowledge*, the principle on which it works. Recall its goal: to agree a set of information of the form $\{(A, INFO_A) \mid A \in \mathbf{G}\}$ amongst the members of $\mathbf{G}$, and hence authenticate each such $INFO_A$ to the node that is declaring it.

If the nodes are programmed to allow any size of group, there is nothing to stop a dishonest node joining into HCBK1. The result would be that the members of $\mathbf{G}$ have some $INFO$ for nodes who are outside the group defined by the empirical channels. This is fine provided they do not assume that all the nodes, who have participated in the run, are in the intended group.

An alternative, which only makes sense if all the nodes in the group are assumed to be trustworthy, is for each node to check that the number of participants corresponds to the expected number. We will call this protocol HCBK2.

If each $INFO_A$ contains a way of sending A data privately, say a public key (which needs not be certificated or long term) or a Diffie-Hellman token, then we could replace the broadcast Message 3 by some means of propagating k_L securely. This could take the form of a separate message from L to each S, or some tree of propagation amongst the S rooted at L. Upon successful completion of the protocol, the group would then have a shared secret k_L. Since it is vital that a shared secret is not shared with untrustworthy nodes, variants of this form are only useful on the assumptions that (a) all members of $\mathbf{G}$ are trustworthy, and (b) the number of participants is checked as in the protocol HCBK2. Clearly this represents a class of potential variant protocols, but we name them all under the heading of HCBK3.

Recall that these protocols depend crucially on the leader L being trustworthy, otherwise a corrupt leader could use a birthday attack essentially like the one we described in Section 4.2.1.

One situation where this is definitely not an issue is when the slave devices themselves have no need

of security. For example, when the leader is seeking to connect its laptop to some wireless peripheral devices, the leader must be sure that the connection is precisely to those devices which are trusted because of their context, labelling etc. In that case, there is no need for empirical channels *from* the leader *to* the slave devices. All we require is that these devices can signal the leader (probably via some display that the leader's user can see) to convey Message 2b and the digest value in Message 4. This would work for all three of the variants described above: the protocols HCBK1, 2 and 3.

Conversely, when the leader wants to broadcast its authentic information to a large number of slaves, then there is no need for the slaves to trust the leader. In that case, an asymmetric empirical communication of the digest value from the leader to all the slaves will be sufficient. For example, this could be done via a display on the leader's device that all of the slaves' human users can see.

4.3 Symmetrised group protocols

The protocols in the previous section all rely crucially upon leader L being trustworthy: what are we to do if there is no node that is uniformly trusted or it is hard to select one, but we still want a *local PKI* authenticating the $INFO_A$s of all trustworthy nodes? What we would like to achieve instead is that a *successful run* of the protocol correctly authenticates all the trustworthy parties to each other, irrespective of what the others may have done.

In order to do this, we identify the following second principle, derived from the design of the HCBK protocol and the *direct binding* strategy of *joint commitment without knowledge*, introduced by us in [78, 79]:

P2 [78, 79] A node A is safe from effective manipulations of its final hash or digest d to equal others in a hash collision or anomaly provided that there is a point at which the following things are both true.

(a) A is committed to its final value $d = digest(k^*, INFOS)$, 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.

Note that this is true of the leader L and the value k_L in the protocol HCBK1 at the point where L has just sent Message 2a. In that protocol, L is completely committed to its digest at the point $longhash(k_L)$ is revealed, so no other node can have used $longhash(k_L)$ in a message. The purpose

of clause (ii) above will become apparent later when more than one node is responsible for the value of the digest key.

Also, we need to be precise about the idea that k_A randomises the calculation of k^*. We will assume that A calculates k^* by some formula from k_A and perhaps some values, which are communicated to it by other nodes and are independent of k_A, thanks to assumption (b)(ii) of **P2**. What we mean by 'randomises k^*' is that if k_A varies uniformly and randomly across its range, then k^* also varies uniformly and randomly across its own for other values being fixed.

The most straightforward way of achieving randomisation is for A to calculate k^* as the XOR of the set of k_As it wants to construct it from.

The fact that this definition is symmetric is an advantage in group protocols, because it does not matter what order each node records the same group in. From here on, we will assume that this XOR method is used, and in fact we have already taken account of this in the definition of a digest function in Section 2.3.3: it lies behind the '$\oplus\ \theta$' in the second part.

In the HCBK protocol, **P2** applies to the leader relative to k_L, which is the only contributor to the final digest key. That protocol relies on much more subtle reasoning in respect of the slave nodes, as shown by the reasoning in the previous section and the principle of Messages 2 and 4 being aligned that we had to adopt. If the slave nodes had been able to follow **P2**, there would have been no need for this.

Our new sort of protocol is designed so that all nodes can rely independently on **P2**. Therefore, each node will now need some value (a sub-key) made up especially for this purpose, and which is fresh and unpredictable in each session. Let us call this value k_A for party A. The protocol is now:

Symmetrised HCBK protocol *New* **[78, 79]** (Informal evidence of security)		
1.	$\forall A \longrightarrow_N \forall A'$:	$A, INFO_A, longhash(A, k_A)$
2.	$\forall A \longrightarrow_N \forall A'$:	k_A
3.	$\forall A \longrightarrow_E \forall A'$:	$digest(k^*, INFOS)$ where k^* is the XOR of all the k_As for $A \in \mathbf{G}$
Computational cost: $NW^2 + NM = 25N + NM$		

The following notes explain these messages.

- **Message 1**: Each node A introduces the information $INFO_A$ it wants to authenticate and a long hash of its sub-key k_A. The identity A is included in this longhash to ensure that the intruder posing as $C \neq A$ cannot simply copy A's sub-key k_A by copying its longhash value to

negate A's randomising effect on k^* (i.e. a reflection attack).[2] After this message each node B should have all the information it requires about the other nodes except for the values k_As, and furthermore should be committed to each of these values in the sense that when told the k_As it will be able to check each one.

At the point when the sending and receiving of this message is complete, it follows that every node A is jointly committed to some final digest value without knowing it. The distinction between being *jointly committed* to a value and *knowing* it is immensely important here. From this we know that **P2** applies.

- **Message boundary**: There has to be some moment at which a node decides it has finished inputting new Message 1s. This might be determined by some timeout, or some message sent from one of the nodes either empirically or over the Dolev-Yao network. It is clearly in nodes' interests that they all make correct decisions on this, for otherwise they will not agree. One can imagine them attempting to synchronise by agreeing on a hash of all the Message 1s they know about over the Dolev-Yao channel. This might well serve a useful purpose, since it would guard against involving humans in empirical communication when there is no point.[3]

  Whatever mechanism they choose does not matter provided it does not involve them revealing the sub-keys k_Xs to each other. For it is absolutely vital that none of them accepts any further Message 1 after its own sub-key k_X is revealed.

- **Message 2**: Each node broadcasts its unguessable sub-key to all other nodes once it is committed to its final digest value. Having received all these sub-keys, each node can check the correctness of all the long hashes received from Messages 1. If there is anything unmatched regarding the long hash values, the node will abort and presumably tell the rest of the group that this has happened.

- **Message 3**: has the members of **G** display and compare the value of digests through the

[2] We could build a check into our protocol by saying that no node A accepts its own value of $longhash(k_A)$ from another user, however, putting the name in makes it clearer. The same reflection attack was also reported in the papers of Čagalj et al. [22], and that is why their pairwise protocol concatenates a single bit (0 and 1) in front of each $INFO$. The same thing is done with the two-way authentication scheme of Vaudenay [119], but he did not make it clear why. Fortunately, the reflection attack does not work against the HCBK scheme, as there is only a single cryptographic hash generated by the leader: $longhash(k_L)$.

[3] Since each device knows how many protocol participants, e.g. this information can be passed down from each human user to his or her own device prior to a protocol run, each device can individually determine when it has received *enough* number of Messages 1 and therefore can reveal its subkey in Message 2. The devices therefore do not need to synchronise on Messages 2, i.e. avoiding the *Byzantine Generals' Problem* [8] in the literature.

empirical channel. Notice that, like the previous protocols we have considered, this digest follows **P1** and includes the whole data of the protocol that parties want to authenticate.

4.3.1 SHCBK protocol analysis

We shall call this the SHCBK protocol, for *Symmetrised Hash Commitment Before Knowledge*. The final result is that the members of **G** are authenticated to each other as the owner of the information they have introduced.

The following theorem will provide supporting evidence for the security of the protocol SHCBK, i.e. the SHCBK protocol achieves its goal of authenticating the $(A, INFO_A)$s.

Theorem 1 [79] Suppose that a group of nodes run of a session of the SHCBK protocol which calls for the agreement on the digest value $d = digest(k^*, INFOS)$. Assume that A and B are two trustworthy parties among the group of nodes then the likelihood that $digest(k_A^*, INFOS_A) = digest(k_B^*, INFOS_B)$ and $(k_A^*, INFOS_A) \neq (k_B^*, INFOS_B)$ is smaller than or equal to 2^{-b}.

Here each node A makes a contribution, sub-key k_A, towards its own calculation of final digest key k_A^* via XOR. And $INFOS_A$ and $INFOS_B$ are A's and B's versions of $INFOS$.

The following proof only provides supporting evidence for the security of the protocol SHCBK because we have assumed that given $INFOS$ and $longhash(A, k_A)$, $longhash(B, k_B)$ ..., it is infeasible for an intruder to gain any advantage in predicting the value of $digest(k^*, INFOS)$, i.e. the digest value (SAS) is a uniform distribution in the intruder's view.

Proof As we have argued above, whatever other (dishonest) nodes pick for their sub-key k_Cs, these values and their hash values are independent of final digest keys k_A^* and k_B^*, thanks to (1) the use of XOR to combine subkeys; and (2) the identities included in $longhash(A, k_A)$ and $longhash(B, k_B)$ in Messages 1 of the SHCBK scheme to avoid a reflexive attack, i.e. part (b)(ii) of **P2** is satisfied.[4] As a result, the actual values of both k_A^* and k_B^* are uniform random variables, whose values no node knew at the point where they agreed to finish inputting the first messages.

Clearly the intruder cannot prevent A having sub-key k_A in its XOR, nor can it prevent B from having sub-key k_B. It can, however, prevent one or other of A and B from having the other's sub-key in the set it XORs. In this case, the values of final digest keys k_A^* and k_B^* are themselves independent uniform random variables as k_A and k_B vary. By the first part of the specification of a

[4] In fact the intruder can use any function derived from A's or B's own longhash to compute k_C. However, since it has no way of relating these longhashes back to k_A and k_B, such values are no better than independent for cryptographic purposes.

digest function, we know that the digest function is uniformly distributed with respect to keys, and therefore the probability of a digest collision in this case should be less than or equal to 2^{-b}, whether $INFOS_A$ and $INFOS_B$ are equal or not.

We now concentrate on the case where each gets to see the other's sub-key. In that case, there is no need for the intruder to allow A and B to see the same set of other sub-keys k_Cs. If all other nodes are under the control of the intruder, we will have final digest keys: $k_A^* = k_A \oplus k_B \oplus \phi$ and $k_B^* = k_A \oplus k_B \oplus \psi$, for values ϕ and ψ controlled by the intruder. So there will be a value $\theta = \phi \oplus \psi$, independent of k_A and k_B and possibly picked by the intruder, such that $k_A^* = k_B^* \oplus \theta$ as k_A and k_B vary randomly.

The probability that $digest(k_A^*, INFOS_A) = digest(k_B^*, INFOS_B)$ when $INFOS_A \neq INFOS_B$ is then smaller than or equal to 2^{-b}, by the second part of the digest specification (Definition 9 of Chapter 2). ∎

Note that this result establishes that B should be in a position to associate $INFO_A$ with the owner of device A confidently, even if nodes other than the two of them are not trustworthy. Thus the protocol SHCBK does indeed authenticate the $INFO_A$s of trustworthy nodes to each other even if corrupt nodes are in $\mathbf{G}$.

It is important to note that our protocols do not supply evidence that nodes *are* trustworthy: mutual trust has to be brought into the protocol from outside, or possibly be established subsequent to the protocol run based on nodes' later communications.

Calling the basic protocol SHCBK1, it can be extended by a count of nodes to create the protocol SHCBK2 for the case where all nodes are assumed to be trustworthy. Asymmetric versions are also possible; they use more computational effort than the asymmetric versions of the HCBK scheme, but avoid the *commit* signals required there.

The robust security achieved here comes at the expense of increased computation compared to the protocol 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 the HCBK scheme. The computation cost of the protocol 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.

4.3.2 Modified version of the protocol SHCBK with proof of security

After the publication of the SHCBK protocol, Laur and Pasini [60] presented a modified scheme where *longhash*() function is replaced by a commitment scheme. The advantage of the scheme is that it is possible to construct a proof of security in the Bellare-Rogaway Model thanks to security properties of a commitment scheme. Laur and Pasini use an almost universal hash function to compute the SAS that we have improved to a digest function in the following protocol.

Modified version of the protocol SHCBK [60]* (Proof of security)	
1. $\forall A$	$\longrightarrow_N \forall A' : A, INFO_A, c_A$ Where $c_A \parallel d_A = commit(A, k_A)$, k_A is a long subkey randomly picked by A.
2. $\forall A$	$\longrightarrow_N \forall A' : d_A$
3. $\forall A$	$\longrightarrow_E \forall A' : digest(k^*, INFOS)$ where k^* is the XOR of all the k_As for $A \in \mathbf{G}$
Computational cost: $NW^2 + NM = 25N + NM$	

Since a commitment scheme has the same computational complexity as a *longhash*() function, the above protocol and the SHCBK protocol are equal in terms of computational cost at $25N + NM$.

4.3.3 Indirect binding group protocol

We have claimed that the direct binding approach (the HCBK and SHCBK protocols) remains more efficient in group protocols than indirect binding as it was in pairwise ones. The argument is true because it is more efficient to have the SAS created from the presumed large $INFOS$ rather than it is to have each $INFO_A$ bounded to random nonces by full-power cryptography to resist combinatorial search attack. In order to illustrate this advantage, we will generalise the (symmetrical) indirect binding pairwise scheme of Čagalj, Čapkun and Hubaux [22] in Section 3.2.1 into a group protocol. The level of security achieved by this scheme is the same as the protocol SHCBK: (1) tolerant of corrupt parties; and (2) the probability of successful one-shot attack is upper bounded by 2^{-b}, where b is still the bitlength of the single SAS transmitted over the empirical channel.

Indirect-binding group protocol *New* [**80**] (Informal evidence of security)	
1. $\forall A$	$\longrightarrow_N \forall A' : INFO_A, c_A$ Where $c_A \parallel d_A = commit(A, INFO_A, R_A)$, R_A is randomly picked by A.
2. $\forall A$	$\longrightarrow_N \forall A' : d_A$ A' computes $R_A = open(A, INFO_A, c_A, d_A)$
3. $\forall A$	$\longrightarrow_E \forall A' : \bigoplus_{A \in \mathbf{G}} R_A$
Computational cost: $NWM = 5NM$	

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

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, i.e. being tolerant of corrupt

parties. 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 chapter.

4.4 De-symmetrised SHCBK protocol

The difference in efficiency between the SHCBK and HCBK protocols raises the question of whether it is possible to reduce the amount of computation processing in the protocol 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, and compute and send its longhash over the normal network. The single node left is the unique slave S, who transmits its fresh sub-key k_S to other nodes only after receiving longhashes from every leader. Below, A is a typical node which is either S or L.

De-symmetrised SHCBK protocol [118]* (Informal evidence of security)			
0.	S	$\longrightarrow_N \forall L$	$: S$
1.	$\forall L$	$\longrightarrow_N \forall A$	$: INFO_L, longhash(L, k_L)$
2.	S	$\longrightarrow_N \forall L$	$: INFO_S, k_S$
3.	$\forall L$	$\longrightarrow_N \forall A$	$: k_L$
4.	$\forall A$	$\longrightarrow_E \forall A'$	$: digest(k^*, INFOS)$ where k^* is the XOR of all the k_As for $A \in \mathbf{G}$
Computational cost: $(N-1)W^2 + NM = 25N + NM - 25$			

We discovered this shortly after the SHCBK scheme. It was also independently invented by Valkonen, Asokan and Nyberg [118], who were not aware of the SHCBK scheme 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 the protocol 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 the protocol 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 without compromising the security. 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 as follows.

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.

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

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

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

This should not take a long time as the 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

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

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

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

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

Note that not only does the above attack work when we XOR digest subkeys as in the SHCBK protocol, but also with any other ways to combine subkeys. This is because the intruder will always be able to predetermine the final digest value before any slave is committed to the digest. Moreover 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.

4.5 Hybrid of the HCBK and SHCBK protocols

We can, however, reduce computational cost if we are prepared to weaken our corruption tolerance requirement towards that of the protocol HCBK. With the addition of 1-bit empirical commitment signals like those in the HCBK protocol and allowing the number of leaders l to vary between 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 the protocol 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 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 everyone in Message 0 by a single node T (either a slave or the leader), who knows this information.

Hybrid HCBK *New* [**80**]		
(Informal evidence of security)		
0.	T	$\longrightarrow_N \forall A : SL$
1.	$\forall A$	$\longrightarrow_N \forall A' : A, INFO_A$
2*a*.	$\forall L$	$\longrightarrow_N \forall A : longhash(L, k_L)$
2*b*.	$\forall S$	$\longrightarrow_E \forall L$:1-bit committed signal
3.	$\forall L$	$\longrightarrow_N \forall A : k_L$
4.	$\forall A$	$\longrightarrow_E \forall A' : digest(k^*, INFOS), leader(SL, A)$
		Where k^* is the XOR of all the k_Ls for $L \in SL$
Computational cost: $lW^2 + NM = 25l + NM$		

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

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

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.[5] Post hoc this establishes agreement on who the leaders are in a very strong way, but with a lot of leaders it could be expensive.

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

[5]In order to illustrate that if the information $leader(SL, A)$ 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 nodes A and B. The intruder invents $INFO'_X$ for each of them in which it says X is a leader, i.e. A will receives $INFO'_B$ saying that B is the leader and vice versa. In fact neither A nor B act as a leader, and the intruder is able to send hash keys to A and B such that the final digests 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 1-bit committed signal 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 2*b*, which is specified in this protocol to save the amount of human work, however, practical implementations might vary significantly from our specification.

about by the intruder and the weaker use of the *leader* information.

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

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

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

4.6 Efficiency of group protocols

Similar to the efficiency analysis of pairwise authentication protocols in Section 3.4, we use the simple cost model of both human work and computing hash and digest functions in sections 2.5.1 and 2.5.2. Even though all protocols in this section are labelled group protocols, we can turn them into pairwise schemes by setting $N = 2$, where N is the number of participants in the group.

From Table 4.1, we can see the same difference in computational cost between indirect and direct information bindings in group protocols as in pairwise authentication schemes of Chapter 3, i.e. direct binding protocols (HCBK, SHCBK, De-symmetrised SHCBK and Hybrid HCBK) are about $W = B/w = 5$ times more efficient than the indirect binding protocol should M get large.

Among the three direct binding protocols, HCBK, Hybrid HCBK and SHCBK, we recall a trade-off between computational cost, the level of corruption resistance, and empirical work arising from 1-bit empirical committed signals in the HCBK and Hybrid HCBK protocols. For example, the SHCBK scheme (and its desymmetrised version) neither relies on the trustworthiness of any party nor requires 1-bit empirical signal, but the SHCBK scheme is slightly more expensive than both HCBK and Hybrid HCBK protocols by $25(N-1)$ and $25(N-l)$ units respectively. Since human or empirical work is

Protocol	Binding	Human work(bit)	Computation cost	Proof of Security
Indirect binding	Indirect	16	$WNM = 5NM$	No
HCBK	Direct	$b + 1$=17	$NM + W^2 = NM + 25$	No
SHCBK	Direct	b =16	$NM + NW^2 = NM + 25N$	No
Laur-Pasini(*longhash*)	Direc	b =16	$WNM + NW^2 = 5NM + 25N$	Yes
Laur-Pasini(*digest*)	Direct	b =16	$NM + NW^2 = NM + 25N$	Yes
De-symmetrised SHCBK(*longhash*)	Direct	b =16	WNM+W^2(N-1)=5NM+25(N-1)	No
De-symmetrised SHCBK (*digest*)	Direct	b =16	NM+W^2(N-1)=NM+25(N-1)	No
Hybrid HCBK (l leaders: $l \in \{1...N\}$)	Direct	$b + 1$ =17	$NM + W^2 l = NM + 25l$	No

Table 4.1: Group authentication protocols (they all use empirical channels: $\longrightarrow_E$)

expensive and does not scale well to multiple-party scenarios, the protocol SHCBK should be much more usable than the HCBK protocol or the Hybrid HCBK protocol in practice, e.g. creating a secure connection between many sensor devices in hospitals or battlefields.

4.7 Conclusions

In this chapter we have explained why it is not easy to design group authentication protocols in this area. We then consider a number of both direct and indirect binding schemes whose supporting evidence of security is derived from the idea of (joint) commitment without knowledge. We note that a direct binding strategy used to achieved commitment without knowledge can be refined further by two design principles **P1** and **P2** introduced in this chapter.

As shown in Table 4.1, the only group authentication protocol considered here that possesses a formal proof of security is the Laur-Pasini scheme. This reflects the difficulty in providing a security proof for group protocols which can involve many parties. This is therefore another challenge in designing group protocol, which remains a topic for future research.

Chapter 5

Non-interactive and one-way authentication protocols

Much of the materials in this chapter has previously been published [80, 82]. The main contributions of this chapter are a security analysis and certain improved versions of the one-way authentication scheme MANA I, which was standardised in 2005 [36, 38]. We also study the advantages of the general protocol design *separation of security concerns* in the context of one-way authentication protocols using human interactions, which subsequently underlies the structure of an alternative and new family of digital signature.

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. Initially, we set out to do this with just one-way communication and short authentication strings (SAS) without the presence of any preliminary security infrastructure.

Although every pairwise and group protocol of chapters 3 and 4 can be adapted to provide one-way authentication, the chief difference between those and the protocols considered here is that one-way protocols can be non-interactive, and thus the absence of one or more protocol participants in a run is still accepted. For this reason, the *joint commitment without knowledge* idea, which involves the participation of every node, does not underlie the security and design of non-interactive protocols as it influentially does in pairwise and group schemes. What we discover instead is that to guarantee that a powerful or multiple-shot attack does not have any advantage over random and one-shot attacks, non-interactive protocols require the use of either more empirical bits or stronger empirical channels. We therefore do not regard one-way and non-interactive schemes as more fundamental than pairwise

and group protocols.

To set this work in context, recall the classic (non-interactive) *digital signature* of Section 2.3.4, whose use is typically supported by a PKI. Here, a message $INFO_A$ of the sender A is accompanied by the digital signature $sign_A(longhash(INFO_A))$. The receiver knows $INFO_A$ really is from A, since he can form the cryptographic hash of $INFO_A$ and verify if it really was A who signed $INFO_A$. 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 chapter than there might appear to be. As 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 digital signature as the combination of a large message $INFO_A$ over an insecure channel with the smaller one $longhash(INFO_A)$ over an authenticated one.

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

We begin this chapter with a number of non-interactive and one-way authentication schemes using empirical channels in different ways. Although non-interactiveness is a useful property allowing the absence of some parties in a protocol run, it brings in constraints which are not present in interactive schemes. For example, to protect the protocols of Pasini-Vaudenay [87], Mashatan-Stinson I/II [68, 69] and Balfanz et al. [10] agaist searching requires the transmission of more empirical bits than desirable, i.e. [80,160] compared to $b \in [16, 20]$ bits in pairwise and group protocols.

Subsequently, we observe the development of the MANA I protocol by Gehrmann, Mitchell and Nyberg [36, 37, 38] that can reduce the number of empirical bits significantly. This was one of the first examples in the literature where any useful search is quite effectively prevented. However, we will see that the scheme neither optimises human effort nor offers as much security as has previously been believed. By applying *direct* and *indirect* information binding strategies (see Definitions 16 and 15 of Chapter 3), we offer a number of improved versions for the MANA I scheme. These provide more security for half the empirical work of the MANA I protocol. To some extent, they can be regarded as simplifications of existing pairwise and group schemes in terms of protocol structure. However, at the expense of non-interactiveness, these require the use of a more general or stronger empirical channel to transmit the SASs to obtain the same level of security as in interactive (e.g. pairwise and group) protocols, i.e. the same probability of a successful attack. The optimality of human work in these schemes can be demonstrated in their security analysis.

Having done this work in the context of empirical channels, we are able to formulate a general principle from it: the principle of *separation of security concerns*, under which protocols should be

designed to resist a random attack and a powerful attack separately, as well as the idea of *factorisation of cryptographic hashing* which can reduce processing cost by a significant factor in these protocols. Although the security analysis and computational efficiency of these schemes have been studied by many researchers [49, 58, 69, 87] to date, as far as we are aware the use and benefits of separation of security concerns have not been analysed in this context before.

These principles in turns allow us to devise a new class of digital signatures that works in the context of a PKI or similar. These schemes often have efficiency advantages over a conventional digital signature. These come from the use of an efficient digest function in place of a cryptographic hash function. Digest functions have a very short output (e.g. 16-32 bits), and hence should be fast to compute, but at the same time are not required to meet some of the strong specifications of cryptographic hash functions. Thus the new signature schemes must be designed in such a way to avoid any search for the value of digest before it is revealed, i.e. we will see how the idea of factorisation of cryptographic hashing can be applied to tackle this challenge. We end this chapter with a new mechanism called *Flexi-MAC* which can provide not only data origin and data integrity but also non-repudiation. The main feature of Flexi-MAC is its flexibility within a single certificate in providing various levels of security in terms of non-repudiation and verification at the cost of extra processing. We therefore hope it will find use in a wide range of applications such as digital certificate and digital rights management.

5.1 Long authentication string protocols

The above analysis of the use of digital signatures shows they are closely analogous to the following one-way authentication protocol, devised by Balfanz et al. [10] and subsequently proved to be secure by Pasini and Vaudenay [87]. In this scheme, A wants to authenticate its information $INFO_A$ to B.

Balfanz et al. non-interactive protocol, [10] (Proof of security [87])
1. $A \longrightarrow_N B : A, INFO_A$ 2. $A \longrightarrow_{WE} B : longhash(A, INFO_A)$ B verifies the longhash.
Computational cost: $WM = 5M$

Under the assumption of a chosen plain-text attack in Section 2.4, the information $INFO_A$ is under the control of the intruder. Hence, it might carry out an off-line search to find a different pair

$(INFO_A, INFO'_A)$ both hashing to the same value.[1]

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

However, this is something which is 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, thanks to Definition 13 of Section 2.5.1.

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 of Section 2.5.2.

In order to improve the number of authenticated bits, Pasini and Vaudenay [87] make use of a 80-bit probabilistic commitment scheme[2] 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 [87] (Proof of security)
1. $A \longrightarrow_N B : c \parallel d = commit_{80}(A, INFO_A)$ B computes $A \parallel INFO_A = open(c, d)$ 2. $A \longrightarrow_{WE} B : hash(c)$ B verifies the hash.
Computational cost: $MW/2 + W^2/4 = 2.5M + 6.25$

Here the hash function is required to be 2nd-preimage-resistant (i.e. an intruder cannot find a second value v' such that $hash(v) = hash(v')$ for fixed v) as opposed to collision resistance required for the protocol of Balfanz et al. where both v and v' are allowed to vary.

In [87], 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

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

[2] A commitment scheme was defined in Section 2.3.1. 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 *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.

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.

More recently, Mashatan and Stinson [68, 69] introduced another two schemes which achieve the same level of security with the same number of empirical bits as the Pasini-Vaudenay scheme but does not require the use of a commitment scheme. The first scheme is non-interactive, the second one is interactive, and both of them are described here.

In the following non-interactive scheme, termed as the Mashatan-Stinson I protocol, the 80-bit hash function $hash()$ is required to be hybrid-collision resistant as defined in [69]. K_A is a long random key of $B/2 = 80$ bits which is generated by A in Message 1.

Mashatan-Stinson I: non-interactive protocol [69] (Proof of security)
1. $A \longrightarrow_N B : INFO_A \parallel K_A$
2. $A \longrightarrow_{WE} B : hash(INFO_A \parallel K_A)$
Computational cost: $(M + W/2)W/2 = 2.5M + 6.25$

Note, the random key k plays the same role as the non-deterministic component (also of $B/2$=80 bits) injected by A into the commitment scheme used in the Pasini-Vaudenay protocol. This therefore implies that both of these obtain the same level of security. The computational cost of the Mashatan-Stinson I protocol is $(M+W/2)W/2 = 2.5M+6.25$, which is the same as the Pasini-Vaudenay scheme, but of course this should be simpler to implement in practice since the Mashatan-Stinson I protocol only requires the use of a 80-bit hash function as opposed to both a hash function and a commitment scheme in the Pasini-Vaudenay scheme.

In the following interactive scheme, termed as the Mashatan-Stinson II protocol, K_A and K_B are long random keys of $B/2 = 80$ bits which are generated by A and respectively B in Messages 1 and 2. The 80-bit hash function $hash()$ is required to be interactive-collision resistant as defined in [68].

Mashatan-Stinson II: interactive protocol [68] (Proof of security)
1. $A \longrightarrow_N B : INFO_A \parallel K_A$
2. $B \longrightarrow_N A : K_B$
3. $A \longrightarrow_{WE} B : hash(INFO_A \parallel K_A \parallel K_B)$
Computational cost: $(M + W)W/2 = 2.5M + 12.5$

The inclusion of both long random keys generated by two parties as input to the hash function slightly increases the computational cost of the Mashatan-Stinson II protocol to $(M+W)W/2 = 2.5M+12.5$, which is 6.25 units higher than those of Pasini-Vaudenay and Mashatan-Stinson I.

It seems fair to remark that since even 80 bits (20 hexadecimal digits) in those of Pasini-Vaudenay and Mashatan-Stinson I/II will seem tedious for most humans to compare carefully or copy from one to another device, these one-way non-interactive protocols are not likely to find widespread use unless the humans have a high level of commitment and possibly a well-designed user interface to ensure user compliance. This is a particularly difficult task when the two 20-hexadecimal-digit numbers which need to be compared by elderly people attempting to set up a secure channels between some wearable sensors and laptops to upload medical data only differ in one or two digits.

5.2 Short authentication string protocols

Gehrmann, Mitchell and Nyberg [36] took a different approach to preventing any useful search. They use empirical channels to transmit the b-bit output of a check function $\mathrm{MAC}_k()$[3] together with a b-bit key that has been instrumental in its computation.

MANA I (Gehrmann, Mitchell and Nyberg) [36, 37, 38] (Informal evidence of security)		
1a.	A	$\longrightarrow_N B : A, INFO_A$
1b.	B	$\longrightarrow_E A$: 1-bit committed signal
		A picks a b-bit random number k
2.	A	$\longrightarrow_E B : k, \mathrm{MAC}_k(A \parallel INFO_A)$
Computational cost: $WM = 5M$		

To eliminate 1-bit empirical signals in the MANA I protocol,[4] Vaudenay proposes to use a strong empirical channel (stall-free or instant delivery, and is denoted $\longrightarrow_{SE}$) to send the key and the check-value.[5] 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.

[3] As suggested in [36, 37], a check function $\mathrm{MAC}_k()$ can be implemented by either CBC-MAC or universal hash functions based on error-correcting codes, which are potentially less efficient than digest functions as discussed in Section 6.2

[4] In the original description of the protocol 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.

[5] We can replace the strong empirical channel with a bounded delay empirical one ($\longrightarrow^t_{BE}$), provided B checks that he has received Message 1 before Message 2 could have been sent.

V-MANA I, [87, 119]*
(Informal evidence of security)
1. $A \longrightarrow_N B : A, INFO_A$
A picks a b-bit random number k
2. $A \longrightarrow_{SE} B : k, digest(k, A \parallel INFO_A)$
B verifies the digest.
Computational cost: M

Binding $INFO_A$ (M words) directly to the SAS makes the protocol efficient because each node computes a single digest at a cost of M: much cheaper than a long output hash function in the Balfanz et al. scheme and a commitment scheme in the Pasini-Vaudenay scheme.[6]

The protocol demonstrates that the use of the strong empirical channel, providing stall-free transmission, will lead to significantly fewer 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 search.

We note, however, that the protocol is suboptimal in human work relative to the level of security obtained. Anyone can modify $INFO_A$ to $INFO'_A$ blindly in the first message and hope that $digest(k, A \parallel INFO_A)$ and $digest(k, A \parallel INFO'_A)$ happen to be the same in Message 2. This will occur with probability 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 [36, 38, 87] 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 function, a MAC or a check-value function such that the probability of any one-shot or collision attack 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 $\{k_1, \cdots, k_c\}$. Using an off-line brute

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

force search at the cost of $2^{bc/2}$ computation steps he can expect to find two different $INFO_A$ and $INFO'_A$,[7] such that:

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

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

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

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.

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

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

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

The above vulnerability indicates we need to increase the 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 —$\mathbf{Pr}_r(c)$ — to converge to 2^{-b}. This is, however, not feasible in this protocol, since the key must be sent with the digest value over the strong empirical channel that is severely limited in bandwidth.

An interesting question arises as we want to know how large the 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. This question is addressed in Chapter 7.[8]

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

[8] There is a known theoretical bound of Stinson [110] on the bitlength of the key that can guarantee the digest meets its specification: bitlength$(k) \geq$ 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 Chapter 7.

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

5.3 Improved versions of the (V-)MANA I protocol

Given the weaknesses of the (V-)MANA-I protocol discussed in the previous section, we propose three different improved versions for the V-MANA I protocol optimising empirical or human work, and these also apply to the MANA I protocol. In other words, human comparison or handling of a b-bit SAS corresponds to a probability $2^{-b} + \epsilon$ of a successful one-shot attack; here ϵ is a negligible value relative to 2^{-b}. Whilst this can only be done at the expense of another (third) message sent over the Dolev-Yao channel, we argue that this is not a bad trade-off because the priority is to optimise empirical work.

In all of these protocols, a node A always sends a commitment of its private key (i.e. using a hash function or a commitment scheme) to B at the beginning of a run. However, different from interactive schemes such as the HCBK or SHCBK protocol, these protocols require the use of timing constraints to ensure that B is committed without knowledge to A's private key. Therefore, with the aid of either strong or bounded delay empirical channels, the protocols achieve *commitment without knowledge* with respect to node B.

These solutions make use of different information binding strategies to fence off any useful search, thanks to the idea of *separation of security concerns* (see Section 5.6). Supporting evidences for security of these protocols will be given separately in the next section.

5.3.1 Direct binding solution

In contrast to the (V-)MANA I protocol, the key k, generated by A in the following protocol, can be as long as we want to ensure that the digest function meets the specification of Section 2.3.3. Furthermore, it *must* have sufficient entropy that it is infeasible for an attacker to deduce k from $longhash(k)$. While the empirical transmission in the V-MANA I protocol is instantaneous to ensure that Message 1 is definitely received before Message 2 is sent, we can weaken the assumption to being of bounded delay as follows.

However, it always has to be significantly longer than b in practice.

Improved version of V-MANA I (direct binding) *New* **[80, 82]** (Informal evidence of security)	
1.	$A \longrightarrow_N \quad B : INFO_A, longhash(k)$
2.	$A \longrightarrow^t_{BE} \quad B : digest(k, INFO_A)$
3.	$A \longrightarrow_N \quad B : k$
Computational cost: $M + W^2 = M + 25$	

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 chapter. 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 which involve searching.

Interestingly, we can replace the bounded delay empirical channel and the need to wait by a simple acknowledgement from B to A. The resulting protocol turns out to be the pairwise (one-way authentication) version of the HCBK protocol of Chapter 4 and [96].

Improved version of MANA I (direct binding) [78, 79, 96] (Informal evidence of security)	
1a.	$A \longrightarrow_N \quad B : INFO_A, longhash(k)$
1b.	$B \longrightarrow_E \quad A$: 1-bit committed signal
2.	$A \longrightarrow_E \quad B : digest(k, INFO_A)$
3.	$A \longrightarrow_N \quad B : k$
Computational cost: $M + W^2 = M + 25$	

We note that the digest and the key (Messages 2 and 3) can be released in any order as long as A has received the commitment signal from B in Message 1.[9] 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

[9]It is worth emphasising that any real life implementation of this protocol must ensure that the 1-bit signal and the digest value are neither blocked nor delayed from one to another session, as specified in the definition of empirical channels in Section 2.1. As pointed out in Section 4.2.1, the assumption on empirical channels can be achieved by one of the following: (1) timing limits on the arrival of Messages 1a, 2 and 3; (2) making sure that B only finishes a run with A after it has received both the digest value and the key; or (3) a combination of (1) and (2), i.e. using a time delay mechanism. If A and B want to have multiple sessions in parallel, then it would be better (in both security and usability) should they include run numbers along side SAS or digest to distinguish the runs.

design, i.e. the same technique can be applied to two other improved versions of the V-MANA I protocol discussed in sections 5.3.2 and 5.3.3.

Since the SASs in these schemes are functionally dependent on the authentic information $INFO_A$, we term these the direct binding version of the Improved (V-)MANA I protocol. However, the computation cost is slightly increased to $W^2 + M = 25 + M$, due to the extra longhash required in the first message.

5.3.2 Indirect binding solution

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

Improved version of V-MANA I (indirect binding) *New* **[82]** (Informal evidence of security)	
1. $A \longrightarrow_N$	$B : INFO_A, c$ $(c, d) = \text{commit}(INFO_A, R)$
2. $A \longrightarrow^t_{BE}$	$B : R$
3. $A \longrightarrow_N$	$B : d$
Computational cost: $W(M + W) = 5M + 25$	

The order and time constraints of messages' arrival in this scheme must be the same as in the direct binding version of Improved V-MANA I. 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.

It is interesting to note that this protocol might be regarded as the non-interactive version of the pairwise (indirect binding) protocol of Vaudenay [119] in Chapter 3.

As in 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) *New* [**82**] (Informal evidence of security)		
1a.	$A \longrightarrow_N$	$B : INFO_A, c$ $(c, d) = \text{commit}(INFO_A, R)$
1b.	$B \longrightarrow_E$	A : 1-bit committed signal
2.	$A \longrightarrow_E$	$B : R$
3.	$A \longrightarrow_N$	$B : d$
Computational cost: $W(M + W) = 5M + 25$		

5.3.3 Solution in Diffie-Hellman style

We propose another improved scheme, whose protocol structure is derived from the pairwise (direct binding) authentication protocol of Hoepman [48, 49] in Chapter 3.

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

Improved version of V-MANA I (Diffie-Hellman style) *New* [**82**] (Informal evidence of security)		
1.	$A \longrightarrow_N$	$B : longhash(g^k)$
2.	$A \longrightarrow^t_{BE}$	$B : shorthash(g^k)$
3.	$A \longrightarrow_N$	$B : g^k$
Computational cost: $WM + M = 6M$		

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

5.4 Security analysis of the Improved (V-)MANA I protocols

As pointed out in Section 2.4, we adapt the Bellare-Rogaway security model where an intruder can control on which node a new protocol instance is launched.

Another important assumption we make is that although the intruder can launch new instances of any party or device, the intruder will not be able to influence the fresh, random or private data which

are generated by the party and are instrumental in the computation of the SAS compared by human owners of the devices, i.e. for otherwise, the intruder could easily fool B into accepting a fake $INFO'_A$ by searching for a digest or short hash collision. Examples are long key k in the direct binding version of Improved (V-)MANA I, and short nonce R and commitment value c in the indirect binding ones.

We are going to give supporting evidence that the Improved (V-)MANA I protocols are secure against a one-shot attack in the first step, and then use Theorem 2 stated below to lift the one-shot attack's model to a powerful attack's model.

The following theorem is the combined result of Lemma 6 of Vaudenay [119] and Theorem 5 of Pasini and Vaudenay [87].

Theorem 2 [82, 87, 119] We consider a powerful attack such that the number of instances of A (respectively B) is at most $\mathcal{Q}_A$ (respectively $\mathcal{Q}_B$).

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

The following proof is not complete since we only consider the case when the intruder cannot influence random keys which are generated by A's instances (possibly launched by the intruder) and are instrumental in the computation of SASs. Note, we believe that the same assumption has also been made by Vaudenay in his proof of this theorem (i.e. Lemma 6 of [119]).

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

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

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

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

Note, the SASs' values of all compatible pairs of instances are uniformly distributed and independent[10] from one another because the SASs are randomised by either random keys (k in direct binding), random nonces (R in indirect binding), or random Diffie-Hellman tokens (g^k in the Diffie-Hellman style version). All of these random elements, which are instrumental in the computation of SASs, are unknown to the intruder at the point when they were generated by A's instances thanks to the above assumption. (This argument remains true even when data $INFO_A$s are controlled by the intruder in the direct binding version, thanks to the use of digest functions.)

We know that the probability of a successful attack on each compatible pair of instances is limited to p (i.e. A and B agree on the same digest of different preimage data $INFO_A$s). We therefore have that the powerful adversary is successful with probability $P \leq p \cdot \mathcal{Q}_A$. ∎

5.4.1 Security analysis of the direct binding improved (V-)MANA I

Theorem 3 [82] Given that $longhash()$ is (ϵ_c, T_c)-collision-resistant and (ϵ_i, T_i)-inversion-resistant, a powerful attack with number of A's (respectively B's) instances bounded by $\mathcal{Q}_A$ (respectively $\mathcal{Q}_B$) is successful against the direct binding versions of Improved (V-)MANA with probability $2^{-b}\mathcal{Q}_A(1 + \epsilon_i + \epsilon_c)$ in a time $Q_A(T_i + T_c)$.

The following proof applies to the direct binding version of Improved V-MANA I, but it can be slightly modified to cope with the direct binding version of Improved MANA I. However, the proof is not complete since:

- We have only taken into account three common (but not all) properties of a cryptographic hash function: collision resistance, 2nd-preimage resistance and inversion resistance.
- As in our supporting evidence for the security of the SHCBK protocol (Theorem 1 of Section 4.3.1), we have to assume that given any $INFO_A$ and $longhash(k)$, it is infeasible to gain any advantage in predicting the value of $digest(k, INFO_A)$, i.e. the digest value should be uniformly distributed even in the presence of m and $longhash(k)$. This problem arises since we are not able to formalise the relation or independence between hash and digest functions, i.e. if someone computes $digest(k, INFO_A)$ as $digest'(longhash(k), INFO_A)$ then the protocol is not secure even though $digest'(longhash(k), INFO_A)$ deems to satisfy all requirements of a digest function.

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

[10] See Definition 14 for what independence means.

A one-shot intruder has no advantage of sending fake $INFO'_A$ and $longhash(k')$ to B (masquerading as A) after the digest is released in Message 2. Therefore, after $INFO_A$ and $longhash(k)$ are sent in Message 1 where k is a private, fresh and long (160-bit) key generated by A in each session and is unknown to any one including the intruder, there are three possibilities that can happen: (1) with probability ϵ_c the intruder can find a hash collision in a time T_c; (2) with probability ϵ_i the intruder can invert the hash value in a time T_i; and (3) with probability $(1 - \epsilon_c - \epsilon_i)$ neither can the intruder find a hash collision nor invert the hash value. Note, there is no need to consider the 2nd-preimage resistance property of a hash function since the intruder does not know key k generated by the honest party A in Message 1.

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

Game against the improved V-MANA I (direct binding)– hash collision				
1.	A	$\longrightarrow_N$	$I(B)$	$: INFO_A, longhash(k)$
	$I(A)$	$\longrightarrow_N$	B	$: INFO'_A, longhash(k')$
2.	A	$\longrightarrow_{SE}$	B	$: digest(k, INFO_A)$
3.	A	$\longrightarrow_N$	$I(B)$	$: k$
Winning condition: $digest(k, INFO_A) = digest(k', INFO'_A)$ or $digest(k, INFO_A) = digest(k'', INFO'_A)$				

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

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

Game against the improved V-MANA I (direct binding)– hash inversion				
1.	A	$\longrightarrow_N$	$I(B)$	$: INFO_A, longhash(k)$
	$I(A)$	$\longrightarrow_N$	B	$: INFO'_A, longhash(k)$
2.	A	$\longrightarrow_{SE}$	B	$: digest(k, INFO_A)$
3.	A	$\longrightarrow_N$	$I(B)$	$: k$
Winning condition: $digest(k, INFO_A) = digest(k, INFO'_A)$ or $digest(k, INFO_A) = digest(k', INFO'_A)$				

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

3. On the other hand, with probability $(1 - \epsilon_i - \epsilon_c)$ in a time $(T_i + T_c)$ neither can the adversary search for a hash collision or invert the hash value. Thus the adversary has to select a random pair $(k', INFO'_A)$ where $INFO_A \neq INFO'_A$.

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

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

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

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

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

5.4.2 Security analysis of the indirect binding improved (V-)MANA I

Theorem 4 [82] Given that a commitment scheme is (ϵ_h, T_h)-hiding and (ϵ_b, T_b)-binding, a powerful attack with number of A's (respectively B's) instances bounded by $\mathcal{Q}_A$ (respectively $\mathcal{Q}_B$) is successful against the indirect binding versions of Improved (V-)MANA with probability $(\epsilon_h + \epsilon_b)\mathcal{Q}_A$ in a time $Q_A(T_b + T_h)$.

The following proof gives supporting evidence for the security of the indirect binding version of Improved (V-)MANA I.

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

- Leaving c unchanged, the intruder sends $INFO'_A$ and c to B (masquerading as A) where $INFO'_A \neq INFO_A$. With probability ϵ_b in a time T_b, the intruder can come up with a d' (which can be either the same as or different from d revealed in Message 3) such that $open(INFO'_A, c, d') = R$ thanks to the binding property of a commitment scheme.
- With probability ϵ_h in a time T_h, the intruder can guess the value of R from $INFO_A$ and c, and then compute (c', d') such that $open(INFO'_A, c', d') = R$ thanks to the hiding property of a commitment scheme.[11]

We can apply Theorem 2 to deduce that any powerful intruder has a success probability of $\mathcal{Q}_A(\epsilon_b + \epsilon_h)$ in a time $Q_A(T_h + T_b)$. ∎

5.4.3 Security analysis of Improved V-MANA I in Diffie-Hellman style

Theorem 5 [82] Given that $longhash()$ is (ϵ_c, T_c)-collision-resistant and (ϵ_i, T_i)-inversion-resistant, a powerful attack with number of A's (respectively B's) instances bounded by $\mathcal{Q}_A$ (respectively $\mathcal{Q}_B$) is successful against the Improved V-MANA I protocol in Diffie-Hellman (D-H) style with probability $2^{-b}\mathcal{Q}_A(1 + \epsilon_c)$ in a time $Q_A(T_c + T_i)$.

The proof for this theorem is similar to the proof of Theorem 3, and so it is not complete due to similar reasons stated above: (1) we only look at three main properties of a cryptographic hash function; and (2) we assume that given $longhash(g^k)$ it is infeasible for the intruder to gain any advantage in predicting the value of $shorthash(g^k)$.

[11] Since $INFO_A \neq INFO'_A$, it is very unlikely that $c = c'$.

Proof As in the proof of Theorem 3, there are three possibilities which can happen after A releases Message 1:

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

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

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

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

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

3. On the other hand, with probability $(1 - \epsilon_i - \epsilon_c)$ in a time $T_i + T_c$ neither can the adversary search for a hash collision or invert the hash value. Thus the adversary has to select a random D-H token $g^{k'}$ and send $longhash(g^{k'})$ to B in Message 1 (masquerading as A).

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

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

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

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

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

5.5 Efficiency of one-way authentication protocols

Unlike other tables in sections 3.4 and 4.6, the levels of security are not identical in the protocols listed in Table 5.1. Here 160 and 80 are examples of the numbers of bits required to make a hash function preimage resistant and second preimage resistant as defined in Section 2.3. We assume that 2^{-b} likelihood of a one-shot attacker success is sufficiently small in all cases (except the first two protocols), but for reasons discussed earlier the (V-)MANA I protocol does not attain this.

As in pairwise and group protocols of chapters 3 and 4, the same difference ($W = B/w = 5$-fold) in computational cost occurs between direct and indirect binding strategies, namely between various improved schemes of the (V-)MANA I protocol. This observation does not however apply to those schemes of Balfanz et al., Pasini-Vaudenay and Mashatan-Stinson I/II because their SASs are not really short, i.e. we need to apply a long output ($\geq$ 80-bit) hash function to the input message in these protocols which are more expensive than a digest function.

Although the computational cost of each of the Improved MANA I protocols is equal to the corresponding version of Improved V-MANA I, their human costs increase slightly to $b + 1 = 17$ bits, thanks to the 1-bit committed signal. However, these bits only need to be transmitted over the *normal* empirical channel versus either *strong* or *bounded delay* empirical channels as in the improved

Protocol	Binding	Human work(bit)	Computation cost	Proof of Security
Balfanz et al.	Direct	B=160 ($\rightarrow_{WE}$)	$WM = 5M$	Yes
Pasini-Vaudenay	Indirect	$B/2$=80($\rightarrow_{WE}$)	$MW/2+W^2/4$=$2.5M$+6.25	Yes
Mashatan-Stinson I	Direct	$B/2$=80 ($\rightarrow_{WE}$)	$(M + W/2)W/2 = 2.5M$+6.25	Yes
Mashatan-Stinson II	Direct	$B/2$=80 ($\rightarrow_{WE}$)	$(M + W)W/2 = 2.5M$+12.5	Yes
MANA I (CBC-MAC)	Direct	$2b$+1=33 ($\rightarrow_{E}$)	$WM = 5M$	No
V-MANA I (*digest*)	Direct	$2b = 32$ ($\rightarrow_{SE}$)	M	No
Improved MANA I	Direct	$b + 1 = 17$ ($\rightarrow^{t}_{E}$)	$M + W^2 = M + 25$	No
Improved MANA I	Indirect	$b + 1 = 17$ ($\rightarrow^{t}_{E}$)	$W(M + W) = 5M + 25$	No
Improved MANA I	Direct(D-H)	$b + 1 = 17$ ($\rightarrow^{t}_{E}$)	$WM + M = 6M$	No
Improved V-MANA I	Direct	$b = 16$ ($\rightarrow^{t}_{BE}$)	$M + W^2 = M + 25$	No
Improved V-MANA I	Indirect	$b = 16$ ($\rightarrow^{t}_{BE}$)	$W(M + W) = 5M + 25$	No
Improved V-MANA I	Direct(D-H)	$b = 16$ ($\rightarrow^{t}_{BE}$)	$WM + M = 6M$	No

Table 5.1: One-way authentication protocols

versions of the V-MANA I protocol.

5.6 Separation of security concerns

When designing an authentication protocol, particularly one based on hash functions, such as the protocols of Balfanz et al., Pasini-Vaudenay or any other protocols presented to date, we typically need, inter alia, to meet the following pair of objectives:[12]

A Powerful attacks, involving searching for hash collisions etc., are made too difficult to carry out (on- or off-line) with any reasonable hope of success, namely that they do not meaningfully improve the attackers' chance of success.

B Whatever guess-work, random or one-shot attack or strategy the attacker can carry out, perhaps involving objective **A**, his or her chances of success are sufficiently low. For example, if humans need to compare a b-bit SAS to complete a protocol run then the success probability of a random attack should be bounded by 2^{-b}.

These two are inextricably linked in traditional uses of hash functions, and indeed we would normally characterise the required strength of a hash function as being what is required to overcome both of these simultaneously. This can be clearly seen in those protocols of Balfanz et al., Pasini-Vaudenay, and Mashatan-Stinson I of Section 5.1.

[12]This observation also applies to password-based key distribution schemes since short (b-bit) passwords are always subject to guess or dictionary attacks. These schemes are designed so that the only way an attacker can find out whether his or her random guess of the underlying password is correct is by interacting with the players. Normally, the success probability of such a random guess attack is bounded by 2^{-b} where b is the password bitlength.

On the other hand, protocols such as our improved versions of the protocol MANA I and the protocol HCBK only work because it has been possible to separate the two above concerns or objectives **A** (powerful attacks) and **B** (random attack). Specifically, these protocols avoid a powerful attack by pre-committing participants to nondeterministic or hidden values (key k), and keep the probability of random attacks working low by choosing a good digest method and a short string of sufficient length.

It was *necessary* to separate these concerns in the protocols, because it was unreasonable to expect humans to transmit or compare a value as complex as a normal cryptographic hash value accurately (or in good temper!), as illustrated in protocols of Balfanz et al., Pasini-Vaudenay, and Mashatan-Stinson I of Section 5.1. It is interesting to note, however, that it brings a quite unexpected benefit. Of the various calculations performed by the participants in the direct binding version of Improved MANA I or (S/H)HCBK, only the calculation of the short string or digest actually depends on the message $INFO_A$ or $INFOS$. It is reasonable to expect that it can be done cheaply as a short output function of $INFO_A$ because the objective of this calculation is only to meet goal **B**, rather than both this and what will almost always be the harder one **A**. According to the cost model of Section 2.5.2, the processing cost of this calculation should be much cheaper than the protocol Balfanz et al., Mashatan-Stinson, and Pasini-Vaudenay when $INFO_A$ is significantly bigger than key k, i.e. from 2.5 to 5 times as seen in Table 5.1. We therefore come to the following striking conclusion: *when $INFO_A$ is large, protocols based on the computation of a short digest can be more efficient than a traditional digital signature scheme based on a standard cryptographic hash of the whole of $INFO_A$.*

This leads us to propose the following principle:

- **Separation of Security Concerns**: where a single cryptographic primitive is being used to satisfy several different security goals, one should consider if efficiency gains can be made by meeting these goals separately. This particularly applies if the primitive is being applied to a large block of data.

A particular consequence of the above principle, derived from protocols such as our Improved (V-)MANA I protocol, is the following:

- **Factorisation of cryptographic hashing**: where a cryptographic hash function is being applied to a substantial item of data, analyse whether its security goals can be achieved more cheaply via a combination of a digest function to limit the chances of random attacks and some constant-time supplementary operations that limit the chances of an attacker to a single try.

5.7 Digital signature without hashing

We have seen the benefits of separating two roles of hashing in manual authentication protocols, in this section we will see another separation but between message transmission and authentication on conventional digital signature schemes which works where there is a PKI: $A \longrightarrow B : (INFO_A, sign_A(longhash(INFO_A)))$. Here, a message $INFO_A$ of the sender A is accompanied by the signature: $sign_A(longhash(INFO_A))$, which provides data integrity, data origin and non-repudiation. However, for largish m (including CDs, DVDs, videos, pictures and medical data), the time for hashing overtakes the time taken for the signature, and will dominate for much larger messages than this.

In order to reduce the time spent on hashing large messages in digital signatures, we will present a number of protocols as alternatives to the conventional method of signing messages. These arise from the idea of factorisation of cryptographic hashing mentioned in the previous section. In the following digital signature schemes, hash functions are replaced by (short-output) digest functions to potentially increase efficiency by a significant factor.

All protocols, given in this and the next sections, were proposed by Roscoe, except that Digital Signature without hashing III and IV were jointly invented by Roscoe and the author.

In the first, A can compute $digest(k, INFO_A)$ simultaneously with sending Message 1. However, A only sends this digest to B once the latter has signalled that it is completely committed to the value $INFO_A$ by sending a nonce.

Digital signature without hashing I (interactive) *New* [**82**] (Informal evidence of security)
1. $A \longrightarrow_N B : INFO_A$
2. $B \longrightarrow_N A : N_B$
3. $A \longrightarrow_N B : sign_A(k, digest(k, INFO_A), N_B)$

Provided that B has not sent Message 2 until it knows (and is therefore committed to) $INFO_A$, it knows that A has not revealed the hash key k to anyone before that point because Message 3 depends on N_B. Note that N_B, communicated over Dolev-Yao channel, is playing the same role of the 1-bit acknowledgement over empirical channel in our Improved MANA I protocol.

In our second protocol, the role of the nonce N_B is replaced by a time stamp ts, whose role is to prove that k was not revealed until B was committed to $INFO_A$. A must therefore wait a sufficient amount of time between completing Message 1 and sending Message 2.

Digital signature without hashing II (non-interactive) *New* [**82**] (Informal evidence of security)
1. $A \longrightarrow_N \quad B : INFO_A$
2. $A \longrightarrow_N \quad B : sign_A(k, digest(k, INFO_A), ts)$

In this scheme, B cannot accept the protocol run unless receipt of $INFO_A$ was complete by time ts, which resembles to the requirement of the bounded delay empirical channel in the Improved V-MANA I scheme. Notice that this version is suitable for broadcasting a message to many Bs simultaneously, but cannot (unlike the traditional digital signature) be used over and over again at different times. This is because the use of the same digest key at different times will allow an intruder and even the sender to search for a second $INFO'_A$, such that $digest(k, INFO_A) = digest(k', INFO'_A)$, and then deploy this against later recipients of the signature. This also implies that these schemes do not provide non-repudiation, and therefore we refer to this family of protocols as *one-time digital signature without non-repudiation*.

A further disadvantage of the above protocols is that they do not permit the recipient to begin digesting until after the key k has been received. We believe, however, that they give both parties a significant reduction in processing time over an ordinary cryptographic hash function. Given the signing operator takes a constant time and message $INFO_A$ can be arbitrarily large, the speed-up factor will approach the ratio between bitlengths of hash and digest functions, e.g. 5 to 10 fold, according to the cost model of Section 2.5.2.

Below we offer several mechanisms that can overcome the latter difficulty. And in the next section, we provide another one termed Flexi-MAC that overcomes all of them at additional processing cost.

A can allow B (or B can allow A) the chance to begin digesting immediately by using a confidential mechanism for the agreement of keys, as can be shown in the following two similar protocols.

Digital signature without hashing III (non-interactive) *New* [**82**] (Informal evidence of security)
$1\alpha.\quad A \longrightarrow_N \quad B : \{k\}_{pk(B)}$
$2.\quad A \longrightarrow_N \quad B : INFO_A$
$3.\quad A \longrightarrow_N \quad B : sign_A(B, longhash(k), digest(k, INFO_A))$

Digital signature without hashing III (interactive) *New* [**82**] (Informal evidence of security)
$1\beta.\quad B \longrightarrow_N \quad A : \{k\}_{pk(A)}$
$2.\quad A \longrightarrow_N \quad B : INFO_A$
$3.\quad A \longrightarrow_N \quad B : sign_A(B, longhash(k), digest(k, INFO_A))$

Notice that B's name and $longhash(k)$ in Message 3 prove to B that Message $1\alpha/\beta$ had k encrypted for only B. This longhash represents proof to B that the key k is unknown to anyone except A and B. For this reason, we can remove the time stamp, timing constraints and the restriction on the order of sending or receiving these messages. The order above is advantageous because it allows both parties to compute $digest(k, INFO_A)$ without delay. Clearly any other way of transmitting secret information from A to B can be used in place of the initial public key encryption.

If there is no need for both parties to digest simultaneously, the three messages can be combined into a single one, and indeed the secret transmission of k can be moved inside the final signed package as in:

Digital signature without hashing IV (α version) *New* [**82**] (Informal evidence of security)
$A \longrightarrow_N B \quad : \quad INFO_A, sign_A(B, \{k\}_{pk(B)}, digest(k, INFO_A))$

One disadvantage of Digital signature without hashing IV is that unless the bitlength of $pk(B)$ is appropriately shorter than $pk(A)$, say 512 and 1024 bits respectively, we have to hash $(B, \{k\}_{pk(B)}, digest(k, INFO_A))$ before signing. This is because the bitlength of the combination of these items is bigger than the bitlength of $sk(A)$. However, we can avoid computing a cryptographic hash if the digest value is moved out of the signature as follows:

Digital signature without hashing IV (β version) *New* (Informal evidence of security)
$1.\quad A \longrightarrow_N \quad B : INFO_A, digest(k, INFO_A)$
$2.\quad A \longrightarrow_N \quad B : sign_A(\{B, k\}_{pk(B)})$

We note that all of the above are tailored to a single recipient or verifier, and we shall refer to them as a *personal* signature. The weakness will be overcome by Flexi-MAC, presented in the next section.

5.8 Flexi-MACs

None of the protocols above are relevant to the important practical problem of allowing one user to publish a piece of data together with some form of MAC that *any* recipient can check at *any* time. In methods I and II the key k is only valid for a short period, whereas in III and IV it is designed only for a single recipient.

Roscoe [82] offers the following concept as a partial solution to this problem. It actually requires *more* effort on the part of the sender than the conventional approach, but of course we hope that this will be more than counterbalanced by two advantages: (1) non-repudiation and (2) the large number of recipients who can check it quickly at any time.

The main idea of a Flexi-MAC is to make A compute a large number of digests of $INFO_A$ under different and randomly chosen keys (set K), which she includes in a single signed block as her 'Flexi-MAC' of the message $INFO_A$. The verifier B then picks a random subset S of K. The cardinality of S determines the security level. The check succeeds if the computed digest of $INFO_A$ under each member or key of S agrees with the corresponding one in the Flexi-MAC. The mechanism is summarised as follows.

Flexi-MAC (non-interactive) *New* **[82]** (Informal evidence of security)
To sign: A devises randomly a set K of distinct keys, and posts $\text{Sign}_A(\{(k, digest(k, INFO_A))\|k \in K\})$
To verify: B chooses a number of k_is $\in K$ at *random* and computes $digest(k_i, INFO_A))$ and checks for equality.

Provided that the verification of each of these individual digests is much easier to compute than a single cryptographic hash, this should still be achievable more quickly than verifying a standard signature. It will also have the advantage that a single signature or certificate can be checked to different degrees, depending on the perceived security threat, the time and computing power available (i.e. the number of keys selected from set K to verify the signature).

For example, suppose that our Flexi-MAC consists of 20 signed key and digest combinations, and we believe it is inconceivable that an attacker can have found a digest collision over more than three of these keys in a short time. Then if the recipient chooses 1, 2 or 3 of the keys at random, it follows

that the chances of an attack succeeding are respectively bounded by 15%, 1.5% or 0.07%. We can always increase the number of signed key and digest combinations to make it harder for the attacker.

By choosing many distinct keys k randomly, A will not be able to change his or her mind about the value of $INFO_A$ later because it would be infeasible to search for a different $INFO'_A$ which digests to the same value under every key in set K. In other words, Flex-MAC provides non-repudiation.

In all of the proposed digital signature schemes with digest functions, we do not have the same human-derived limit on the width of the digest function as in the non-standard authentication protocols based on human interactions. It seems probable that the digests in Flexi-MACs can be wider, say 64 bits. What we still require, of course, is that they are efficient to calculate relative to cryptographic hashing. In addition, Flexi-MACs also allow outdated but well understood hash functions such as MD5 or SHA-1 to be used in the place of digest functions.

5.9 Conclusions

We have studied a number of one-way and non-interactive authentication protocols in this chapter. Since they can be non-interactive, the idea of joint commitment without knowledge does not underlie the security of these protocols. Instead we need to use either longer SASs or stronger empirical channels to make these protocols secure. This work fortunately led to the discovery of the design principle of separation of security concerns, which underlies the security and the computational efficiency of the three improved versions of the MANA I protocol introduced in this chapter.

In the second half of the chapter, a new alternative to conventional digital signatures has been introduced. The new schemes have the advantage of reduced computational cost by comparison with traditional schemes, because they replace cryptographic hash functions with short and more efficient digest functions. However, we have only studied the simplified version of digital signatures where padding (e.g. OAEP)[13] is not considered. It would be interesting to know whether we can use digest functions within padding operators to increase computational efficiency even further. One weakness of the new signing algorithms is that they do not provide non-repudiation, except in the case of Flexi-MAC of Section 5.8.

We hope that the new digital signature schemes with digest functions and Flexi-MAC introduced here will find many interesting applications in the area of Digital Rights Management (DRM) where the integrity and origin of an extremely large volume of data (such as images, DVDs or CDs in the media industry) might need to be verified at the customers' end in a limited amount of time. Another

[13]Modern digital signatures involve padding to ensure that the attacker cannot manipulate the plaintext to exploit the mathematical structure of the primitive.

possible application of the new schemes is in (secure) teleconferencing, in which data needs to be signed and to be transmitted to the receiver simultaneously in real time, i.e. computational efficiency becomes an important design factor.

Chapter 6

Digest functions

Much of the material presented in this chapter has previously been published [79, 83]. The main contributions are two digest constructions. The first is based on Toeplitz matrix multiplication and ϵ-biased sequences of random variables. The second uses integer multiplications and a structural similarity between Toeplitz matrix multiplication and the "school book" algorithm for integer long multiplication.

Digest functions have similarities to cryptographic hash and universal hash functions, the second of which was extensively studied in the early nineties by many researchers [25, 51, 55, 110, 120]. The main difference between digest functions and nearly all other families of hash functions is that digest functions typically have a very short output in the range between 16 and 64 bits. For this reason, it is not possible to generate digest functions which can resist collision and inversion attacks in the same way as cryptographic hash functions. This feature however opens the way for efficiently computed digest functions arising from both their short output and the potential of parallel computation. We observe that parallelism is unfortunately not available with many current cryptographic hash functions, such as SHA-I and MD5, whose design strictly follows the *sequential* Merkle-Damgård construction [32].[1] The main goal of this chapter is therefore to study efficient digest constructions which are provably secure regarding their security properties, including distribution and some other imposed restrictions in a digest function, e.g. the likelihood of digest collision with respect to distinct keys are taken into

[1] There have been a number of newly proposed cryptographic hash functions, such as MD6 [95] and SANDstorm [4] (both of which were entered into the hash competition [3] organised by NIST recently), which make use of the hash-tree structure introduced by Merkle [75] to benefit from parallel computation. We note that there are two weaknesses of a hash-tree construction: (1) the speed-up factor of parallel computation is bounded above by the depth of the hash tree, i.e. $\log n$ where n is the length of input messages, relative to the sequential Merkle-Damgård construction; and (2) to the best of our knowledge, there might be security weaknesses due to the variable depth of the tree arising from variable length input message, and thus this structure has not been widely adopted to construct efficient hash functions for commercial purposes.

consideration.

Although short-output digest functions can be computed by just truncating the first small number of bits of a cryptographic or universal hash function, as discussed in Section 6.2, we will point out that this strategy does not exploit the short output of digest and parallel computation to increase computational efficiency. We subsequently prove that a well-studied universal hashing algorithm based on Toeplitz matrix multiplication and ϵ-biased sequences of random variables can generate digests with the properties we require in Definition 18.

The second main contribution of this chapter is the discovery of a structural similarity between Toeplitz matrix multiplication and the "school book" algorithm for integer long multiplication. Even though these are not exactly the same due to the effect of carry bits in integer multiplications, this work suggests another method using word multiplications which can be computed efficiently on any processor. We note that carry bits in integer multiplication make it not feasible to give a mathematical proof of digest properties, we will instead exploit the short output of digest functions to facilitate our digest collision and distribution tests with statistical analysis. This approach includes computing the p-values in chi-square tests, quantile-quantile plots and the maximum median calculation of collision and distribution test results on word multiplication based digest algorithm. This will suggests carry bits in integer long multiplication do not introduce much bias into the digest computation regime relative to the Toeplitz based method. To the best of our knowledge, this is the first time when statistical approach has been used to analyse the security properties of hash, universal hash and digest functions. Without carry bits our word multiplication based digest construction is identical to the Toeplitz one, and indeed such a carry-less instruction for word multiplication has been introduced into the next generation of Intel processor [2].

Both our digest constructions in this chapter can benefit from parallel computation should they be computed in multi-core microprocessors, such as a 48-core CPU recently announced by Intel [1].

6.1 Notation

Every operator used in this chapter is bitwise, which is equivalent to working over the finite field $\mathbb{F}_2$. Therefore, we will replace bitwise exclusive-OR and bitwise AND with addition (or summation) and multiplication in $\mathbb{F}_2$, respectively. We note that addition is equivalent to subtraction in $\mathbb{F}_2$, and so we will stick to the use of addition to simplify the notation. Also, to avoid confusion, '$\times$' and '*' are used to indicate finite field multiplication and respectively integer multiplication. We refer to m as the input message (i.e. a bit vector) of digest functions from now on.

In this chapter, we only consider non-zero messages representable by K bits, and hence the set of all messages is denoted $X = \{1...(2^K - 1)\}$. r, K and b are the bitlengths of digest or hash key, input message, and digest or hash output.

Definition 17 [55] An ϵ-balanced universal hash function is a set of 2^r hash functions $h_k()$ where $k \in R$ such that $h_k : X \rightarrow Y$; here $R = \{0...(2^r - 1)\}$, $X = \{1...(2^K - 1)\}$ and $Y = \{0...(2^b - 1)\}$. Moreover, for every non-zero message m and every hash output $y \in Y$, we have:

$$\Pr_{\{k \in R\}}[digest(k, m) = y] \leq \epsilon$$

Extending the definition of an ϵ-balanced universal hash function, we define an ϵ-balanced digest function as follows.

Definition 18 [79] A ϵ-balanced *digest* function: $digest : R \times X \rightarrow Y$ where $R = \{0...(2^r - 1)\}$, $X = \{1...(2^K - 1)\}$ and $Y = \{0...(2^b - 1)\}$ are the set of all keys, input messages and digest outputs, and moreover:

- for every $m \in X$ and $y \in Y$, $\Pr_{\{k \in R\}}[digest(k, m) = y] \leq \epsilon$
- for every $m, m' \in X$ $(m \neq m')$ and any $\theta \in R$: $\Pr_{\{k \in R\}}[digest(k, m) = digest(k + \theta, m')] \leq \epsilon$

Note that the above definition of a digest function is the extended version of Definition 9 of Section 2.3.3 in two ways. First the ditribution and digest collision probabilities are parameterised by ϵ as opposed to being fixed at 2^{-b}. Secondly we have replaced $\oplus\theta$ in the second requirement with $+\theta$, because the bitwise exclusive-or operator is equivalent to addition over the finite field $\mathbb{F}_2$. Finally, any function satisfying Definition 18 also satisfies both Definition 17 and the requirements of an ϵ-almost universal hash function of Section 2.3.2.

Definition 19 [9, 55] Let S be a distribution of sequences of length l. Let (α, s) denote the scalar (or inner) product modulo 2 of $\alpha \in \{0,1\}^l$ and $s \in \{0,1\}^l$.

- S is said to pass the linear test $\alpha \in \{0,1\}^l$ with bias ϵ if $|\Pr_{\{s \in S\}}[(\alpha, s) = 1] - 1/2| \leq \epsilon$.
- S is said to be an ϵ-biased distribution if it passes all linear tests $\alpha \neq 0$ with bias ϵ.

Definition 20 A Toeplitz or diagonal-constant matrix A is a (not necessary square) matrix where each descending diagonal from left to right is constant, i.e. for all pairs of indexes (i, j): $A_{i,j} = A_{i+1,j+1}$.

6.2 Background information

It would be a great mistake to compute $digest(k, m)$ as some function $f()$ of k and some similar length $digest'(m)$, i.e. $digest'()$ is not collision-resistant or the lengths of $digest'$ and $digest$ are similar, which it might be tempting to do.

$$digest(k, m) = f(k, digest'(m))$$

This is because an intruder could search for a set of different input messages all digesting to the same value $digest'(m)$ irrespective of the value of k, i.e. any pair m and m' of these would violate the second requirement of a digest function. We therefore need to embed the key k fundamentally into the calculation of $digest(k, m)$: it cannot be an afterthought.

Although digest output is short, it is challenging to construct digest algorithms that have both efficiency and provable security relative to the digest's requirements. Thus the majority of short-output digest constructions invented to date [7, 36, 86] make use of cryptographic hash functions, as seen in the following examples.

Pasini and Vaudenay [86], Gehrmann et al. [36], and the Bluetooth white-paper [7] suggested the following scheme:

$$digest(k, m) = trunc_b\left(hash(k \parallel m)\right)$$

where $trunc_b()$ function truncates to the leading b bits. We make two observations about this.

1. The definition of a cryptographic hash function does not normally specify the distribution of individual groups of bits, and so whether or not this is a good idea will very much depend on the properties we require of the hash function and the properties we want from the digest function. It follows that a specific analysis would be required for any particular standard cryptographic hash function proposed.

2. Computing a longhash operating on long words as in the sequential Merkle-Damgård construction can be potentially expensive, i.e. neither does the Merkle-Damgård structure permit parallelism nor exploit the short output of a digest function to increase efficiency, as can be seen in Figure 2.1 of Chapter 2.

Taking a different approach, Gehrmann and Nyberg [36] proposed using error-correcting codes, such as the extended Reed-Solomon codes, to construct the digest function. This has the advantage of having a coherent mathematical structure. On the other hand, to have unconditional security, the algorithm limits the bitlength of the input message to some fixed number, such as 128 or 256, which is the length

of a dataword input by the algorithm. To digest any significant amount of data, the algorithm first compresses the input message into that number of bits by using a cryptographic hash which can be inefficient as discussed above. This feature also makes the scheme be potentially vulnerable to attacks should the intruder find (off line) a collision on the cryptographic hash, i.e. this is equivalent to finding a pair of distinct messages which violate the second requirement of a digest function. The reason for this weakness is because the input message is not entirely linked to the key in digest computation, as discussed at the beginning of this section.

6.3 Toeplitz matrix product construction of a digest function

We now describe a universal hash function construction using Toeplitz matrixes whose binary elements are drawn from ϵ-biased sequences of (pseudo)random bits. This construction was proved by Krawczyk in [55] to satisfy the requirements of an ϵ-balanced universal hash function. We will then show this algorithm can be used to generate a digest function with the properties we require.

Suppose we want to compute a b-bit universal hash of a (non-zero) K-bit message m, then the key k must be drawn from an ϵ-biased distribution R of length $r = K + b - 1$. Using the key k, we can generate a Toeplitz matrix $A(k)$ of K rows and b columns. Using matrix product, we define:

$$h_k(m) \quad = \quad m \times A(k) \tag{6.1}$$

The symbol '$\times$' represents the product of the vector m and the matrix $A(k)$ in $\mathbb{F}_2$.

Theorem 6 [55] $h_k(m)$ defined in Equation (6.1) above satisfies the definition of an ϵ-balanced universal hash function.

Readers who are interested in the proof of this theorem can find it in the paper of Krawczyk [55] (Theorem 9, Lemma 10 and Theorem 5). The proof makes use of a known relation between Discrete Fourier Transform of matrix multiplication and ϵ-biased distributions.

If we replace a universal hash function $h_k(m)$ with a digest function $digest_T(k, m)$, where index T indicates a Toeplitz based digest construction, we have:

$$digest_T(k, m) \quad = \quad m \times A(k) \tag{6.2}$$

Theorem 7 $digest_T(k, m)$ defined in Equation (6.2) above satisfies the definition of an ϵ-balanced digest function.

Proof The first requirement of an ϵ-balanced digest function is satisfied thanks to Theorem 6.

An observation on the digest construction using Toeplitz matrix multiplication is that this construction is linear in terms of both input message m and key k. The latter is true because for any k and θ of length $r = K + b - 1$ bits, we always have $A(k + \theta) = A(k) + A(\theta)$ and hence $digest_T(k + \theta, m) = digest_T(k, m) + digest_T(\theta, m)$.

For any pair of distinct messages (m, m') and any θ of length r bit, a digest collision $digest_T(k, m) = digest_T(k + \theta, m')$ is equivalent to:

$$\begin{aligned} digest_T(k, m) &= digest_T(k + \theta, m') \\ m \times A(k) &= m' \times A(k + \theta) \\ m \times A(k) &= m' \times A(k) + m' \times A(\theta) \\ (m - m') \times A(k) &= m' \times A(\theta) \\ digest_T(k, m - m') &= m' \times A(\theta) \end{aligned}$$

Since we have fixed m, m' and θ, let us denote $m'' = m - m'$ and $\alpha = m' \times A(\theta)$. And the second requirement of a digest function is satisfied because:

$$\Pr_{\{k \in R\}}[digest_T(k, m) = digest_T(k + \theta, m')] = \Pr_{\{k \in R\}}[digest_T(k, m'') = \alpha] \leq \epsilon$$

The inequality in the above equation is true thanks to the first requirement of a digest function (or Theorem 6). ∎

In the above Toeplitz based digest computation, the key k is assumed to have a bitlength of $r = K+b-1$ which can be long if the input message is large. In practice, however, a key is normally of the size of a typical cryptographic hash function, say $l = 160$ or 256 bits, and hence we need to derive the required number r of bits pseudorandomly from the initial l bits.

One possible way is to use a Lineal Feedback Shift Register (LFSR) as suggested by Alon et al. [9] (Proposition 1 of Section 3). In this LFSR construction, the first $l/2$ bits of key k will be used to select the linear structure of the underlying $l/2$-bit LFSR used, i.e. this is equivalent to selecting an irreducible polynomial of degree $l/2$. This LFSR is then seeded by the other $l/2$ bits of key k. This construction has been shown by Alon et al. [9] to produce an ϵ-biased distribution on length $r = K + b - 1$ with $\epsilon = \frac{r}{2^{l/2}}$, where each sequence is generated out of l initial bits.

There are a number of other well-studied constructions of ϵ-biased sequences due to Alon et al. [9],

including Legendre symbol construction and scalar product construction, both of which can be found in their paper.

6.3.1 Efficiency of Toeplitz matrix based digest functions

Given an ϵ-biased sequence of length $r = K + b - 1$ bits,[2] it is straightforward to design a customised hardware implementation of $digest_T(k, m)$, which can also benefit from parallelism.

We first look at the complexity of a sequential implementation of this algorithm. The cost of multiplying a K-bit vector by a $K \times b$ matrix is $2Kb$ bit operations, which are either AND or XOR in this case. On a w-bit microprocessor, we can expect this to take at least $2Kb/w$ bitwise word operations. This figure clearly shows that the cost of computing digest function is at least proportional to its output length. To simplify the calculation, we assume that w equals b, and so this is equivalent to $2K$ word operations. Since the latency of most logical instructions (AND, OR, NOT, and XOR) of modern microprocessors is 1 clock cycle, executing $2K$ bitwise word operations takes $2K$ arithmetic clock cycles.

As regards parallel computation, we observe that a matrix multiplication permits parallelism in two distinct ways. First, each digest output bit can be computed on its own, and hence the complexity can be improved by a factor of b. Secondly, a $K \times b$ matrix multiplication can be split into $t = K/w$ parallel $w \times b$ matrices multiplications with their resulting b-bit vectors being XORed to produce the digest value. The second method is thus $t = K/w$ times more efficient than the sequential computation.

6.3.2 Hardware implementation of digest functions

In this section, we present a modified algorithm which is an approximate hardware implementation of the digest construction based on Toeplitz matrix multiplications. This uses operations which can be executed efficiently in standard microprocessors.

Suppose we are given input data m with bitlength K, and $r = s * b$ is some multiple of the desired output length b. We first need to generate $(K/s) + b$ pseudorandom numbers each of length r, seeded by k, in a register R on designated clock cycles. One possibility is to use one or more linear feedback shift registers (LFSRs) to produce pseudorandom numbers, each seeded with the whole of k or some linear function of it, as illustrated in Figure 6.1 and discussed in [54, 55, 40]. We initialise a shift register S and an accumulator register A both with length r to standard values, possibly 0. The random number generators are designed to produce r bits on each cycle in a register R.

[2] The r-bit ϵ-biased sequence can be derived (off-line) pseudorandomly from a shorter key and stored in both devices for subsequent digest computation.

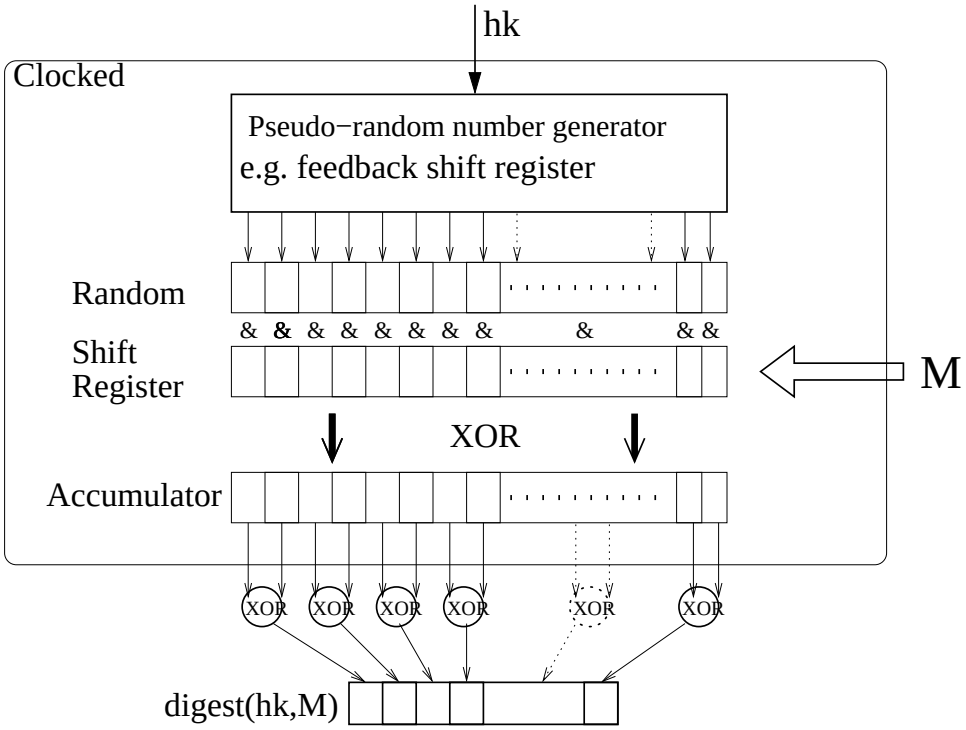


Figure 6.1: Hardware implementation of a digest function.

On each designated clock cycle, the register S is shifted by s bits introducing the next s bits of m or some standard values, possibly 0 if m is already complete. By this means the whole of m will have passed entirely though S after $(K/s) + b$ cycles. Also on each designated clock cycle, we enable the replacement of the accumulator register A by the previous value *bitwise-exclusive-or-ed* (addition in $\mathbb{F}_2$) with the *bitwise-and* (multiplication in $\mathbb{F}_2$) of the registers R and S.

$$A = A \oplus (R \wedge S)$$

After the $(K/s) + b$ cycles are completed, we partition the register A into b groups of s consecutive bits. The bits of each of the said b groups are *exclusive-or-ed* (addition in $\mathbb{F}_2$) to a single bit. Finally, the juxtaposition of the said b bits will form the digest value. That way each input bit has contributed once to each bit of the digest as in the software implementation (see the previous section).

The efficiency of the implementation can be improved if we switch to using a Toeplitz matrix of random values, rather than the completely random one discussed above. What this means is that instead of producing r fresh random bits from a PRNG each cycle, we only need to shift the register R by s bits to the left and fill the s most least significant bits by new s random bits per each cycle.

6.4 Comparing Toeplitz matrix and integer long multiplication

In this section, we will show a structural similarity between the "school book" algorithm for integer long multiplication and Toeplitz matrix multiplication. This similarity, in the next section, will lead to a more efficient algorithm for digest function.

Suppose as in the Toeplitz-based method of Section 6.3, key k is drawn from an ϵ-biased distribution R of length $r = K + b - 1$. Consider a "school book" algorithm for integer long multiplication of two numbers k and m represented in binary, i.e. $k = (k_1 \dots k_r)$ and $m = (m_1 \dots m_K)$. The following is, however, only a part of the table that is used to carry out the integer long multiplication.

	k_1	k_2	$\cdots$	k_{b-1}	k_b	$\cdots$	k_r
					m_1	$\cdots$	m_K
	$m_K \times k_1$	$m_K \times k_2$	$\cdots$	$m_K \times k_{b-1}$	$m_K \times k_b$	$\cdots$	$\cdots$
$\cdots$	$m_{K-1} \times k_2$	$m_{K-1} \times k_3$	$\cdots$	$m_{K-1} \times k_b$	$m_{K-1} \times k_{b+1}$	$\cdots$	
$\cdots$	$m_{K-2} \times k_3$	$m_{K-2} \times k_4$	$\cdots$	$m_{K-2} \times k_{b+1}$	$m_{K-2} \times k_{b+2}$	$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\cdots$	$m_1 \times k_K$	$m_1 \times k_{K+1}$	$\cdots$	$m_1 \times k_{r-1}$	$m_1 \times k_r$		
$\cdots$	d_1	d_2	$\cdots$	d_{b-1}	d_b	$\cdots$	$\cdots$

We are only interested in the overlap of the expanded multiplication, where each bit of the resulting product $(d_1 \dots d_b)$ is influenced *directly* by every bit of the message m. In other words, each carry bit c_i which is instrumental in the computation of d_i is considered to have an *indirect* impact.

We note that the overlapping part of the expanded multiplication can be interpreted as a matrix of b columns and K rows. Moreover, the bits k_is in each row of this matrix are shifted by 1 position to the right hand side as we move from one to the upper row. This means that if we reverse the order of these rows in the expanded multiplication, we will have the property of a Toeplitz matrix. As a result, the b-bit digest value is equivalent to:

$$digest_M(k, m) \quad = \quad m \times A(k) + \quad (c_1 \dots c_b) \tag{6.3}$$

Note that the index "M" in $digest_M(k, m)$ indicates an integer long multiplication based digest construction. Here $A(k)$ is a Toeplitz matrix of b columns and K rows, which is constructed by the same $r = K + b - 1$ bits of (or derived from) key k, and $(c_1 \dots c_b)$ is a bit vector whose elements are carry bits which are instrumental in the computation of the corresponding output or digest bits.

The above description demonstrates a structural similarity between integer long multiplication and Toeplitz matrix multiplication, however they are not equivalent due to the impact of carry bits which are accumulated from d_b all the way to d_1 in an integer long multiplication. Nevertheless, it opens the way for devising new digest constructions using word multiplications which can be computed fast on any microprocessor. This will be addressed in the next section.

6.5 Word multiplication based digest functions

Although the Toeplitz matrix based digest function can be computed efficiently on customised hardware, many applications of digest functions will need to carry out this operation in software on

standard and sometimes basic microprocessors. In this section, we will show that another digest function $digest_{WM}(k, m)$ with strong structural similarities to both $digest_T(k, m)$ and $digest_M(k, m)$ can be calculated using standard integer multiplication for half or whole word blocks that are implemented efficiently in just about all microprocessors. This method is inspired by the relation between the "school book" algorithm for integer long multiplication and Toeplitz matrix multiplication, as demonstrated in Section 6.4.

Let us divide message m into b-bit blocks $[m_1, \ldots, m_{t=\frac{K}{b}}]$. We generate $t+1$ pseudorandom b-bit blocks k_i derived from key k.[3] In the following equation, the index "WM" of $digest_{WM}(k, m)$ indicates a b-bit word multiplication based digest function.

$$digest_{WM}(k, m) \quad = \quad \sum_{i=1}^{t} [\mathrm{Low}(m_i * k_i) + \mathrm{Up}(m_i * k_{i+1})] \tag{6.4}$$

Here, * refers to a word multiplication of two b-bit blocks which produces a $2b$-bit output. The output of each word multiplication can be partitioned into the lower and upper (b-bit) halves by applying Low() and Up() functions respectively. Both '+' and $\sum$ used in Equation (6.4) above are additions in $\mathbb{F}_2$.

To see why $digest_{WM}(k, m)$ resembles $digest_M(k, m)$, we give a simple example when the bitlength of the input message is $K = tb = 3b$ in Figure 6.2.

As was pointed out in Section 6.4, there may be some asymmetry due to carry bits in integer or word multiplications. However, the effect of carry bits in this construction is reduced significantly relative to an integer long multiplication of $digest_M(k, m)$ as described in Section 6.4, where carry bits are accumulated from the least to the most significant bits of the product. In contrast, by partitioning an integer long multiplication into multiple (b-bit) word multiplications, carry bits are only accumulated locally, i.e. up to $2b - 1$ bits to the left hand side of each position as opposed to $K(K + b - 1)$ bits in an integer long multiplication of Section 6.4.

The above construction produces a b-bit digest, in practice, we might want to construct longer digests whose output is a multiple of b, i.e. xb-bit digest function. Using the same idea, it is not hard to generalise the above algorithm to multiple-word digest construction as follows.

We still divide m into b-bit blocks $[m_1, \ldots, m_{t=\frac{K}{b}}]$. However, we will need to generate $t + x$ pseudorandom b-bit blocks k_i derived from key k to compute a xb-bit digest. For all $j \in \{1...x\}$, we

[3] Any pseudorandom number generator (PRNG) which produces ϵ-biased sequences can be used here. A possible candidate is a LFSR whose linear structure and initial seed are derived from key k as described in Section 6.3.

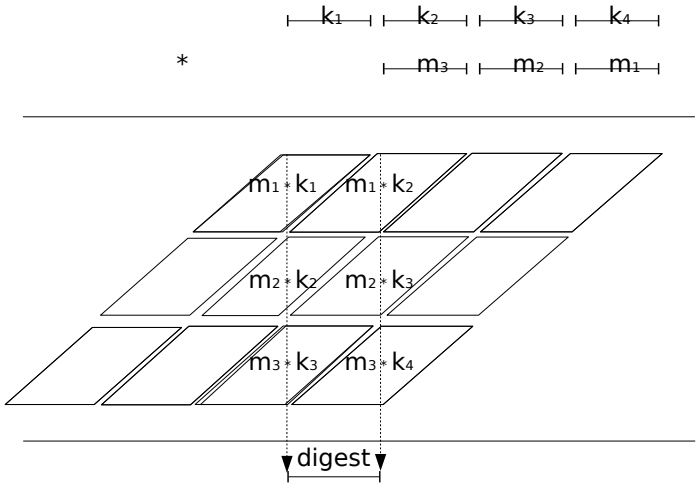


Figure 6.2: Word multiplication model $digest_{WM}(k, m)$. Each parallelogram equals the expansion of a word multiplication between a b-bit key block and a b-bit message block.

then define:

$$d_j = \sum_{i=1}^{t} [\text{Low}(m_i * k_{i+j-1}) + \text{Up}(m_i * k_{i+j})]$$

And

$$digest_{WM}(k, m) = (d_1 \cdots \ d_x)$$

Note that the negative and biased impact of carry bits in integer multiplications can be eliminated entirely if we make use of the carry-less multiplication instruction, termed PCLMULQDQ in the next generation of Intel processors [2], that computes the carry-less multiplication of two 64-bit operands without the generation or propagation of carry bits.[4] This will make $digest_{WM}(k, m)$ identical to $digest_T(k, m)$ as defined in Equation (6.2) of Section 6.3.

6.5.1 Efficiency of word multiplication based digest functions

Given an ϵ-biased sequence of length $r = K + b - 1$, the computation of a $(b = 64)$-bit $digest_{WM}(k, m)$ as defined in Equation (6.4) consists of $2t = 2K/b$ word multiplications and $2t = 2K/b$ bitwise XOR

[4]The primary purpose of this new instruction is for computing the Galois Hash, which is the underlying computation of the Galois Counter Mode (GCM) and AES-GCM.

operations.[5] In the worst case, each word multiplication takes $w = b$ clock cycles to complete, but it is always significantly faster in many modern microprocessors which have RISC pipeline instructions. For example, AMD (K8 and K10) and Intel CPUs [17] can dispatch a $64 \times 64 = 128$-bit MUL once every 2-4 cycles with a latency of 4-7 cycles. To simplify the calculation, we only consider non-pipeline execution, and thus the number of *arithmetic* clock cycles of computing a (b=64)-bit $digest_{WM}(k, m)$ in an AMD CPU is $\frac{2K}{64}4 + \frac{2K}{64} = 5K/32$ compared to $2K$ arithmetic clock cycles of computing $digest_T(k, m)$ of Section 6.3.1.

Another observation we want to make is that the computation of $digest_T(k, m)$ is done via many ($2K$) logical instructions (AND, OR and XOR) as opposed to a significantly smaller number of MUL and logical instructions in $digest_{WM}(k, m)$. As a consequence, the number of *control* clock cycles (e.g. loop indexing, fetching and writing) involved in $digest_{WM}(k, m)$ is potentially reduced by a factor of $\frac{2K}{2K/b+2K/b} = b/2$ relative to $digest_T(k, m)$.

The word multiplication based method also permits a high level of parallelism in multi-core processors [1] thanks to its modular structure, i.e. the digest computation can be easily split into many groups of equal or different sizes of independent word multiplications.

6.5.2 Security analysis and usability

Part of the security of the word multiplication based digest function can be based on the structural similarity between the school book algorithm for integer long multiplication and the constant diagonal property of a Toeplitz matrix. Of course, there is some effect of carry bits, and hence we have conducted some statistical tests described in later section and referred below to show that carry bits do not seem to introduce much bias into the computation regime.

Those statistical tests of the word multiplication based construction are carried over 10240000 keys and 2^{16} pairs of distinct messages, and there are two important points that can be deduced from the results.

- First we consider a number of exceptional cases that an attacker could focus on, including clustered messages whose statistical properties do not deviate from a random function, i.e. the collision probability $\epsilon \approx 2^{-b}$. There is however a minor bias regarding very dense or very sparse messages where $\epsilon \approx 2^{1-b}$, but this does not break our construction as will discussed later in this subsection.

[5] There is no cost of applying Low() and Up() functions to the product of 2 word operands because in a 64/32-bit processor the multiplication result of two 64/32-bit words is typically stored in two 64/32-bit registers which are the upper and lower halves of the product.

- Secondly, our statistical tests are designed to catch a certain bad construction of digest functions. For example, suppose $digest(k, m) = f(k, digest'(m))$, where $digest'()$ is a random function and of similar length as $digest(k, m)$. Since function $digest'()$ has a short output, e.g. $b = 16$, and our collision tests always involve 2^{16} pairs of distinct messages, then with a very high probability there exists a pair of messages that produces a collision on $digest'()$ regardless of what $digest'()$ is, and this pair of messages would collide on $digest()$ for all keys. Thus this type of bad construction will be immediately caught by our tests.

On one hand carry bits make it difficult to provide a security proof, on the other hand they introduce non-linearity which gives an advantage over the Toeplitz matrix method in terms of *collision multiplication attack*, i.e. this is the ability, given a collision (m_1, m_2), to derive an arbitrarily number of collisions. The Toepliz matrix based method suffers from this attack[6] because given m_1, m_2, k, θ such that $digest_T(k, m_1) = digest_T(k \oplus \theta, m_2)$ then for any pair m' and k' of equal length we have: $digest_T(k \parallel k', m_1 \parallel m') = digest_T((k \oplus \theta) \parallel k', m_2 \parallel m')$, i.e. this can be seen as a special case of length extension attack. In contrast, we believe that such an attack does not work agaist the word multiplication based construction due to the non-linearity of carry bits influenced by both key and message. More generally, we found that it is very diffcult to control the impact of carry bits without the knowledge of keys (which are private in applications of digest functions), and to come up with a clever strategy (e.g. collision on pairs of variable-length messages) to give the attacker an advantage in searching for collisions.

We note that digest functions, to the best of our knowledge, only need to resist collision attacks when being used in the non-standard authentication technology as well as a new family of digital signature where cryptographic hash functions can be replaced with this new primitive to increase computational efficiency. Thus anything faster than an exhaustive search does not automatically implies a cryptographic breaks, i.e. $\epsilon = 2^{1-b}$ as seen above, even when $b = 16$, is secure enough for a collision attack on this type of function. What we look for is the right trade-off between security and efficiency, which is why we parameterise the collision probability as ϵ in our work.

[6] Please note that defeating collision multiplication attack is not required in many current cryptographic applications of digest function.

6.6 Statistical tests of word and Toeplitz matrix multiplications

Since if digest keys are derived from a standard pseudorandom number generator (PRNG), which produces ϵ-biased sequences of pseudorandom bits, clearly we should expect good statistical results on Toeplitz based digest construction $digest_T(k, m)$ thanks to its mathematical structure (Theorem 7). In this section, we concentrate on verifying the quality of word multiplication based digest function $digest_{WM}(k, m)$ with respect to its distribution and collision properties.

In our digest collision tests, we consider N pairs of distinct messages in combination with a large set $\mathcal{K}$ of keys. For each ith pair of messages (m_i, m_i'), x_i denotes the number of distinct keys from set $\mathcal{K}$ of all keys which result in digest collisions. What we want to verify is the following null hypothesis H_0:

> H_0: The observed numbers of collision keys of all N trials $\{x_1, \ldots, x_N\}$ should fall into a binomial distribution: $x_i \sim Binom(\|\mathcal{K}\|, \epsilon)$, where ϵ is the pairwise-collision probability of a digest function and $\|\mathcal{K}\|$ is the cardinality of set $\mathcal{K}$.

The motivation behind this null hypothesis comes from the fact that in an ideal digest function the chance ϵ that a pair of distinct messages digesting to the same value under key k_1 is the same as and independent from under another randomly chosen (different) key k_2. To check the accuracy of this hypothesis, we first carry out the *chi-square test* to compute the corresponding p-value [94]:

$$p\text{-value} = \Pr(\mathcal{X}_V^2 > \text{Observed } X^2)$$

Secondly we compare the distribution of $\{x_1 \ldots x_N\}$ against an ideal binomial distribution $Binom(\|\mathcal{K}\|, \epsilon)$ by plotting their quantiles against each other. This is called the *Quantile-Quantile* or Q-Q plot [94]. If the two distributions being compared are similar, then the points in the Q-Q plot will approximately lie on the line $y = x$, which is also drawn in each Q-Q plot to illustrate any difference between the two.

To calculate the collision probability of a digest construction, we will look at the maximum x' of the set $\{x_1 \cdots x_N\}$. And suppose that the null hypothesis H_0 is correct then the value of x' should be very near to the maximum median x at which all N trials from the binomial distribution $Binom(\|\mathcal{K}\|, \epsilon)$

are 50% likely to be less than x. This means that

$$(\Pr\{x_i \leq x\})^N = \left(\sum_{t=0}^{x} \binom{\|\mathcal{K}\|}{t} \epsilon^t (1-\epsilon)^{\|\mathcal{K}\|-t}\right)^N \approx \left(\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \epsilon^{-\frac{(t-\mu)^2}{2\sigma^2}} dt\right)^N = 1/2$$

Here $\Pr\{x_i \leq x\}$ is the probability that the ith trial (the number of collision keys for the ith pair of messages) is smaller than x, and $\mu = \|\mathcal{K}\|\epsilon$ and $\sigma^2 = \|\mathcal{K}\|\epsilon(1-\epsilon)$. Since all N trials are themselves independent, we need to raise $\Pr\{x_i \leq x\}$ to the power of N. The approximation between the cumulative binomial distribution function and the cumulative normal distribution function is due to the DeMoive-Laplace limit theorem [100]. The point x, which is used to check the accuracy of x' (the observed maximum number of collision keys across N pairs of messages), can then be found by using tabulations of the probability mass functions of the normal distribution [6]. For example, when $N = 2^{16}$, $\epsilon = 2^{-10}$ and $\|\mathcal{K}\| = 102400, 1024000$ and 10240000, then the expected values for observed x's are respectively 142, 1134 and 10424, as given in Tables 6.2, 6.4, 6.12 and 6.6.

Regarding a digest distribution test, the main difference between this and collision one is that we will fix N distinct pairs of message and digest output, and then for each ith pair (m_i, d_i) we denote y_i the number of keys from set $\mathcal{K}$ satisfying the equation $digest_{WM}(k, m_i) = d_i$. Since for any pair (m_i, d_i), the probability that key k_1 satisfying $digest(k_1, m_i) = d_i$ is the same as and independent from another randomly chosen key k_2, the set $\{y_1 \ldots y_N\}$ should also fall into a binomial distribution, i.e. $y_i \sim Binom(\|\mathcal{K}\|, \epsilon)$. As a result, the same methods of analysing collision tests (chi-square test, Q-Q plots and max median) described above apply to distribution tests.

In our tests, we generate 32-bit digest functions, and so it is sufficient to consider 32-bit input messages. The resulting length for digest keys that are pseudorandomly generated in our test is therefore 64 bits. Since digest keys are usually not influenced by an attacker at the point when they are invented in many applications of digest functions [36, 58, 79, 82, 86], we use the Mersenne Twister pseudorandom number generator [70] to derive large sets $\mathcal{K}$ of 64-bit keys.

A problem in carrying out collision (and distribution) tests is that it is very expensive when we want to acquire meaningful and reliable results about collision (and distribution) on the whole of a 32-bit digest output.[7] For this reason, we will only consider collisions on groups of 10 adjacent bits of a digest output, which perhaps unexpectedly gives us evidence on any difference in distribution

[7]For example, to have around $1024 = 2^{10}$ 32-bit digest collisions for each pair of messages, we need to consider or generate pseudorandomly $2^{32} \times 2^{10} = 2^{42}$ distinct keys. Since there are 2^{16} pairs of messages in our tests, the number of digest computations is $2 \times 2^{16} \times 2^{42} = 2^{59}$ or $2^{59} \times 2^{11} = 2^{70}$ bit operations, i.e. each 32-bit digest computation of a 64-bit key and 32-bit messages requires two 32-bit word multiplication adding up to $2 \times 2^5 \times 2^5 = 2^{11}$ bit operations. This therefore goes beyond the computation capability of our computers, and it will be much more than this for longer digest functions and longer messages.

between these group of bits potentially caused by the impact of carry bits in integer multiplication. This also makes the expected probability of a pairwise 10-bit digest collision be $\epsilon = 2^{-10}$ in all of our tests. In particular, we will look at the following three groups of bits: 10LSB, 10MIB and 10MSB which denote the 10 least, the 10 middle and respectively the 10 most significant bits of a 32-bit digest output of $digest_{WM}(k, m)$.

As opposed to digest keys, input messages are public and can be under the control of an attacker, we therefore consider the following four types of input messages:

- Series 1: Pseudorandom messages derived from the Mersenne Twister [70];
- Series 2: Sequential (or cluster) messages where the initial message or pair of messages is derived from the Mersenne Twister;
- Series 3: Sparse and cluster messages where the initial pair of messages in collision tests is $(1,2^8+1)$; and
- Series 4: Dense and cluster messages where the initial pair of messages in collision tests is $(2^{32}-1,2^{32}-257)$.

6.6.1 Analysis of statistical test results

We have implemented our tests relative to three different numbers of keys, i.e. capitals A, B and C denote 102400, 1024000 and 10240000 which are the cardinalities of set $\mathcal{K}$. However, due to lack of space, we will only give p-values of chi-square tests and their Q-Q plots for the case of 10240000 keys as presented in Tables 6.1, 6.3, 6.9, 6.11, 6.7 and 6.5. We note that to ensure uniformity in our computation of p-values in chi-square tests for collision and distribution, we always use the same set of bins (or intervals) to count the observed occurrences in each interval.[8]

Since a large number of tables reporting the statistical results of our collision and distribution tests considered here do not always make it easy to see their purposes and importance, we summarise the contents of all of these tables in the following figure. Another digest construction termed $digest_{XWM}(k, m)$ will be described at the end of this section.

[8]The sets $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_N\}$ are counted into the following 16 intervals: [0,9650), [9650,9700), [9700,9750), [9750,9800), [9800,9850), [9850,9900), [9900,9950), [9950,10000), [10000,10050),[10050,10100), [10100,10150), [10150,10200), [10200,10250), [10250,10300), [10300,10350), [10350,$+\infty$), and therefore the degree of freedom in all chi-square tests is 15. Here $10000 = 10240000 * 2^{-10} = N\epsilon$ is the expected value of mean of number of keys across N trials in our chi-square tests.

Table	Digest method	Digest property	$\|\mathcal{K}\|$	N	Tests
6.1	$digest_{WM}()$	distribution	C	2^{18}	Chi-square and Q-Q plot
6.2	$digest_{WM}()$	distribution	A,B,C	2^{18}	mean, max (no of keys) and variance
6.3	$digest_{WM}()$	collision	C	2^{16}	Chi-square and Q-Q plot
6.4	$digest_{WM}()$	collision	A,B,C	2^{16}	mean, max (no of keys) and variance
6.5	$digest_{XWM}()$	collision	C	2^{16}	Chi-square and Q-Q plot
6.6	$digest_{XWM}()$	collision	A,B,C	2^{16}	mean, max (no of keys) and variance
6.7	$digest_{XWM}()$	distribution	C	2^{18}	Chi-square and Q-Q plot
6.8	$digest_{XWM}()$	distribution	A,B,C	2^{18}	mean, max (no of keys) and variance
6.9	$digest_{T}()$	distribution	C	2^{17}	Chi-square and Q-Q plot
6.10	$digest_{T}()$	distribution	A,B,C	2^{17}	mean, max (no of keys) and variance
6.11	$digest_{T}()$	collision	C	2^{16}	Chi-square and Q-Q plot
6.12	$digest_{T}()$	collision	A,B,C	2^{16}	mean, max (no of keys) and variance

It is clear from Tables 6.1 that the distribution property of $digest_{WM}(k, m)$ approaches optimality (i.e. as in an ideal binomial distribution) because of the right range of the observed p-values and nearly all the points in the Q-Q plots approximately lie on the line $y = x$ in Series 1-4. This observation is strengthened by results reported in Table 6.2 which indicates that the ratio between max and mean numbers of keys converges to 1 as the cardinality of set $\mathcal{K}$ increases. Please note that to exploit the 10-bit digest output and to discover if any correlation exits among all distinct digest values with respect to the same message, in our distribution tests we always pair each message with each of every possible digest output from 0 to $1023 = 2^{10} - 1$, i.e. in Table 6.2, $N = 2^{18}$ pairs are made up from 2^8 distinct messages and 2^{10} digest values. As a result, the mean values of numbers of keys in all distribution tests equal $\|\mathcal{K}\|\epsilon$ exactly as seen in Tables 6.2, 6.8 and 6.10. What we are interested in are the distribution properties of $\{y_1 \dots y_N\}$ represented by p-value, Q-Q plot, max and variance.[9]

As regards collision property, Table 6.3 suggests when input messages are pseudorandomly generated (Series 1-2), the numbers of collision keys in $digest_{WM}(k, m)$ fall into a binomial distribution (i.e. the null hypothesis H_0 is accurate), because both p-values are in the right range and nearly all points in the Q-Q plots lie approximately on the line $y = x$. Moreover, the maximum number of observed collision keys across all N pair of distinct messages, their mean and variance values (Series 1-2 of Table 6.4) are all near to the expected values in an ideal scenarios. As in distribution tests,

[9] When each message is not paired up with each of every possible digest output, it is highly unlikely that the observed value of mean equals $\|\mathcal{K}\|\epsilon$ exactly. This has been tested and yielded similar statistical results as the above case.

the ratio between the maximum number of observed collision keys and Mean converges to 1 with each additional order of magnitude in the size of the set $\mathcal{K}$, i.e. 10-time increase in the number of keys.

We also obtain similar results regarding 10MSB, 10 MIB and 10MSB in both distribution and collision tests of $digest_{WM}(k, m)$ when messages are pseudorandomly generated as seen in Series 1–2 of Tables 6.3–6.4, and Series 1–4 of Table 6.1–6.2. This suggests that the digest output bits are equivalent in their quality, and thus carry bits in integer multiplication does not introduce much bias into the digest computation regime.

However, when it comes to either sparse or dense (and cluster) messages as seen in Series 3-4 of Table 6.3, we detect unsatisfactory behaviours. For example, there are many points in Q-Q plots moving far away (both above and below) from the line $y = x$, and their p-values are extremely small compared to Series 1–2. To confirm the negative impacts of dense and sparse messages, we look at the ratio between Max and Mean numbers of collision keys in Series 3–4 of Table 6.4, and realise that it is around 2 and does not seem to improve even when we increase the number of keys. We therefore conclude that the null hypothesis H_0 stated above is not correct in this case.

To fix this weakness, we need to destroy known patterns within input messages. In other words, we want to randomise input messages in an unpredictable, but also deterministic, way prior to them being processed by our digest schemes. One possible and efficient way is to exclusive-or the message with the output of some function $f()$ applying to the key k, resulting in the following improved digest construction:

$$digest_{XWM}(k, m) = digest_{WM}(k, m \oplus f(k))$$

Since k is random and fresh in each protocol session, and so is the modified version of m. In particular, when we try $f(k) = k_1 \oplus k_2$, where k_1 and k_2 are the two (32-bit) halves of key k, then we seem to be able to get around the negative impact caused by very dense or sparse messages (except a bias in the 10MIB relative to 10MSB and 10LSB of a 32-bit digest output) as reported in Series 3–4 of Tables 6.5–6.6. Devising good ways to randomise input messages is therefore a subject for future research.

6.7 Conclusions and future research for short-output functions

There are two main contributions about a new short-output digest function presented in this chapter. First, we prove that Toeplitz matrix multiplication coupled with ϵ-biased sequences of (pseudo)random

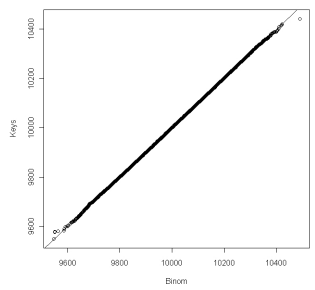
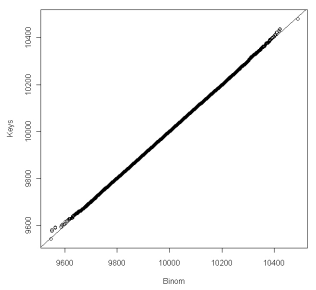
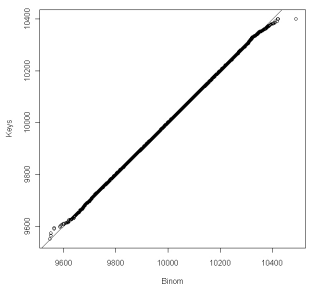
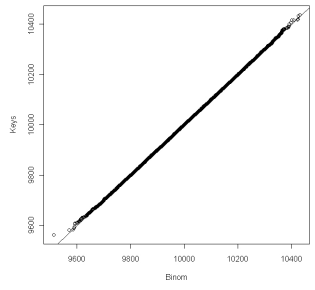
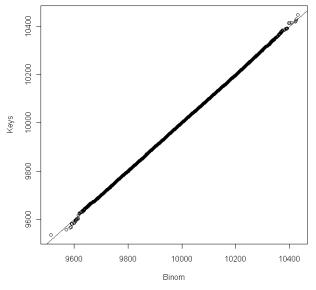
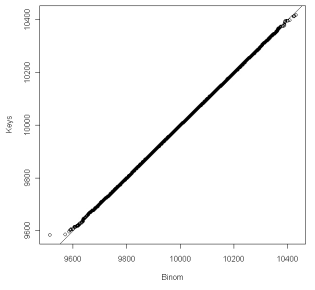
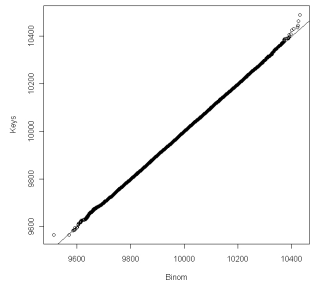
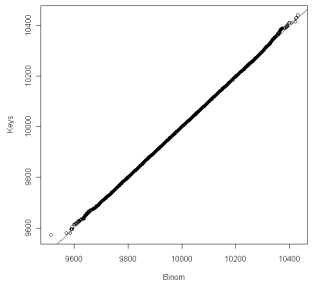
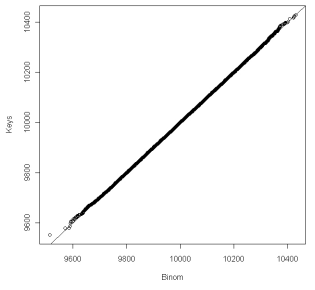
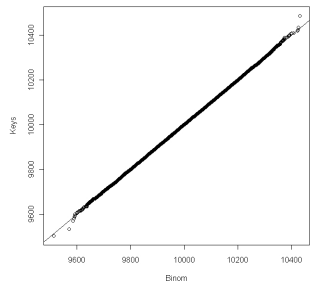
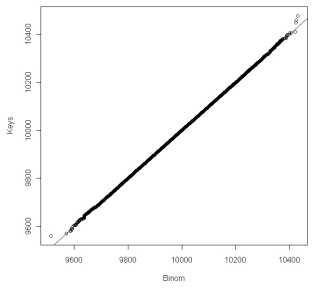
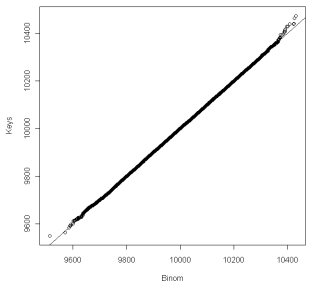
	10LSB	10MIB	10MSB
Pseudorandom 32-bit messages.			
Series 1C	p-value = 0.2316	p-value = 0.1379	p-value = 0.1986
			
Cluster 32-bit messages: the initial message is derived pseudorandomly.			
Series 2C	p-value = 0.5744	p-value = 0.958	p-value = 0.4005
			
Sparse and Cluster 32-bit messages: the initial message is 1.			
Series 3C	p-value = 0.2192	p-value = 0.1688	p-value = 0.2438
			
Dense and Cluster 32-bit messages: the initial message is 2^{32}-1.			
Series 4C	p-value = 0.5379	p-value = 0.905	p-value = 0.6854
			

Table 6.1: Chi-square and Q-Q plots of distribution test results of 32-bit $digest_{WM}(k, m)$: $N = 2^{18}$ pairs of message and digest output (m, d) over $\|\mathcal{K}\| = 10240000$ keys. In each Q-Q plot, the vertical (Keys) and horizontal (Binom) axes denote quantiles on observed keys and respectively binomial distribution.

Test Series	No of keys $\|\mathcal{K}\|$	Map range	Expected Mean & Max numbers of keys, and Variance	Observed Mean μ'	Observed Max y'	Observed Variance σ'^2
			$\mu = \|\mathcal{K}\|\epsilon$ $\sigma^2 = \|\mathcal{K}\|\epsilon(1-\epsilon)$ $\int_{-\infty}^{y} \epsilon^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = 2^{-1/N}$	$\Sigma_{i=1}^{N} y_i/N$	max of $\{y_1 \dots y_N\}$	$\Sigma_{i=1}^{N} \frac{(y_i-\mu')^2}{N}$
Pseudorandom 32-bit messages.						
1A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	148 149 148	100.45 99.86 99.80
1B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1139 1149 1146	1001.43 998.39 1002.97
1C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10442 10481 10401	10049.08 9941.39 10027.96
Cluster 32-bit messages: the initial message is derived pseudorandomly.						
2A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	148 149 152	99.63 99.94 99.99
2B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1143 1141 1149	998.92 1002.83 1001.46
2C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10437 10449 10419	10036.47 9999.81 9944.26
Sparse and Cluster 32-bit messages: the initial message is 1.						
3A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	155 149 150	99.38 99.10 99.90
3B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1147 1142 1139	996.47 991.83 987.38
3C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10490 10443 10430	10022.99 9914.02 10063.45
Dense and Cluster 32-bit messages: the initial message is $2^{32} - 1$.						
4A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	155 148 152	99.54 100.31 100.08
4B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1146 1137 1143	1002.80 1001.23 998.89
4C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10486 10477 10473	9998.82 9986.94 10032.88

Table 6.2: Comparing the mean, max (no of keys) and variance of distribution test results of 32-bit $digest_{WM}(k, m)$ against their expected values in an ideal binomial distribution: $N = 2^{18}$ distinct pairs of message and digest output (m, d). The set $\{y_1 \dots y_N\}$ denotes the number of keys observed and counted for all N pairs (m, d).

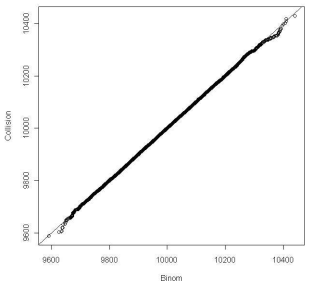
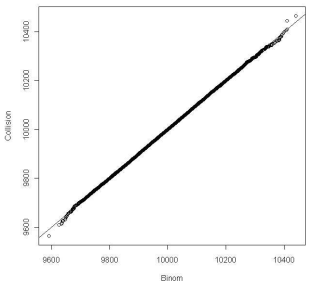
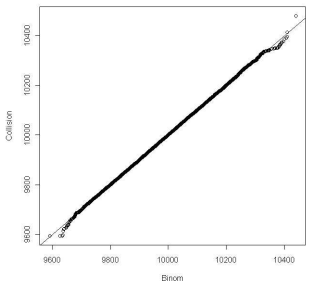
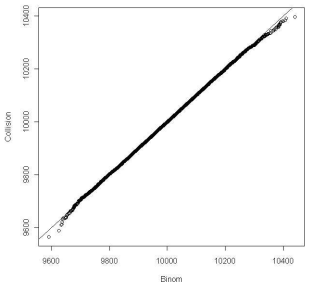
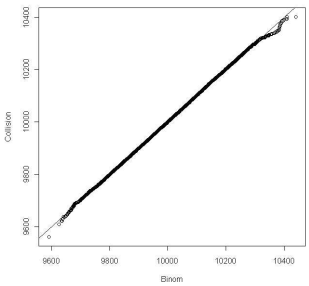
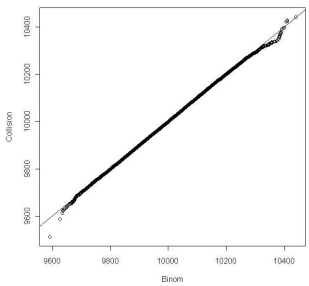
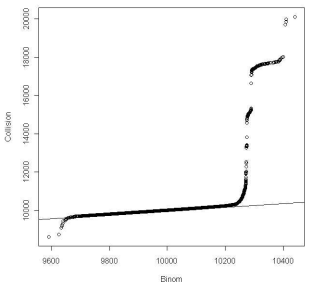
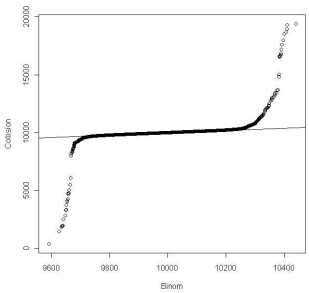
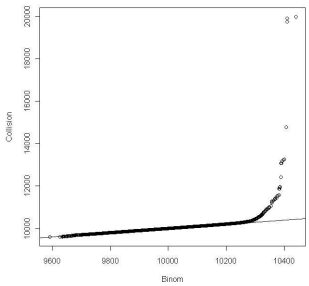
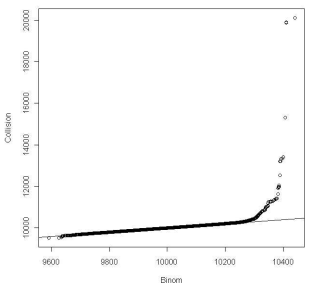
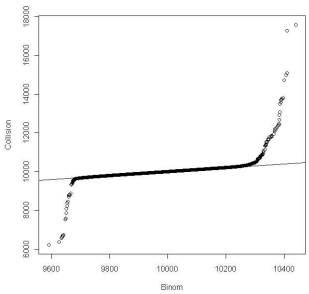
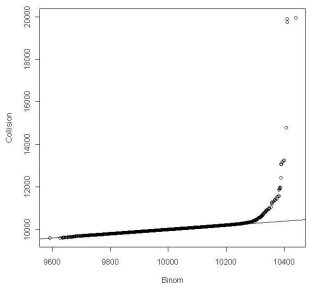
	10LSB	10MIB	10MSB
Pseudorandom 32-bit messages.			
Series 1C	p-value = 0.6868	p-value = 0.4816	p-value = 0.4816
			
Cluster 32-bit messages: the initial pair of messages is derived pseudorandomly.			
Series 2C	p-value = 0.6366	p-value = 0.5683	p-value = 0.3670
			
Sparse and Cluster 32-bit messages: the initial pair of messages is $(1,2^8+1)$.			
Series 3C	p-value = 2.2e-16	p-value = 2.2e-16	p-value = 2.2e-16
			
Dense and Cluster 32-bit messages: the initial pair of messages is $(2^{32}-1, 2^{32}-257)$.			
Series 4C	p-value = 2.2e-16	p-value = 2.2e-16	p-value = 2.2e-16
			

Table 6.3: Chi-square and Q-Q plots of collision test results of 32-bit $digest_{WM}(k,m)$: $N = 2^{16}$ pairs of 32-bit messages over $\|\mathcal{K}\| = 10240000$ keys. In each Q-Q plot, the vertical (Collision) and horizontal (Binom) axes denote quantiles on observed collision keys and respectively binomial distribution.

Test Series	No of keys $\|\mathcal{K}\|$	Map range	Expected Mean & Max numbers of collision keys, and Variance	Observed Mean μ'	Observed Max x'	Observed Variance σ'^2
			$\mu = \|\mathcal{K}\|\epsilon$ $\sigma^2 = \|\mathcal{K}\|\epsilon(1-\epsilon)$ $\int_{-\infty}^{x} \epsilon^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = 2^{-1/N}$	$\Sigma_{i=1}^{N} x_i/N$	max of $\{x_1 \ldots x_N\}$	$\Sigma_{i=1}^{N} \frac{(x_i-\mu')^2}{N}$
Pseudorandom 32-bit messages.						
1A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	100.10 99.61 99.99	156 162 140	99.5 98.1 99.8
1B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	999.91 999.98 999.97	1153 1128 1134	993.9 994.3 1000.5
1C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	10000.73 10000.36 9999.99	10412 10433 10404	10039.8 10010.3 10102.2
Cluster 32-bit messages: the initial pair of messages is derived pseudorandomly.						
2A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	100.05 99.97 100.04	153 143 145	99.0 100.4 99.1
2B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	1000.07 1000.03 999.68	1135 1133 1146	1004.7 1000.6 1004.4
2C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	9999.74 10000.10 10000.41	10406 10431 10415	10037.5 10026.4 9998.5
Sparse and Cluster 32-bit messages: the initial pair of messages is $(1, 2^8+1)$.						
3A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	100.32 100.02 100.12	214 195 224	114.2 101.8 99.99
3B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	1002.80 1000.27 1000.01	2059 1976 1998	2434.8 1355.1 1061.6
3C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	10027.32 10003.16 10005.45	20123 19348 20047	155437.4 44738.7 17346.7
Dense and Cluster 32-bit messages: the initial pair of messages is $(2^{32}-1, 2^{32}-257)$.						
4A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	100.06 100.02 100.18	202 163 224	99.8 100.3 100.2
4B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	1000.29 1000.45 999.97	2059 1692 2003	1069.7 1077.6 1062.7
4C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	10003.03 10003.72 10005.41	19905 17478 20090	17230.5 19184.4 17413.0

Table 6.4: Comparing the mean, max (no of keys) and variance of collision test results of 32-bit $digest_{WM}(k, m)$ against their expected values in an ideal binomial distribution: $N = 2^{16}$ pairs of 32-bit messages. The set $\{x_1 \ldots x_N\}$ denotes the number of collision keys observed and counted for all N trials.

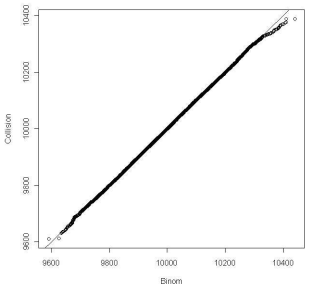
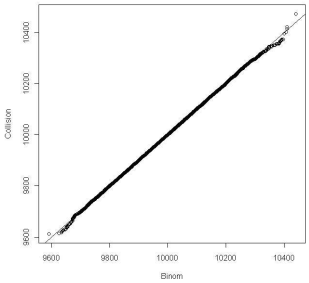
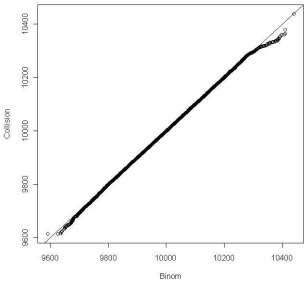
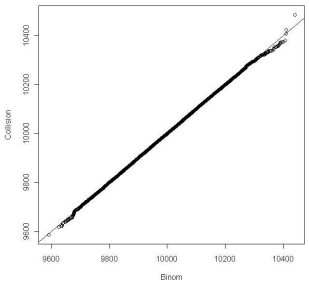
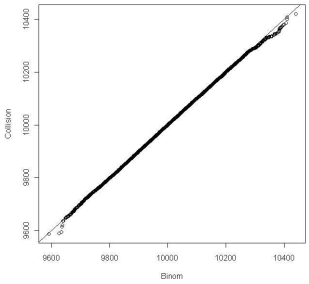
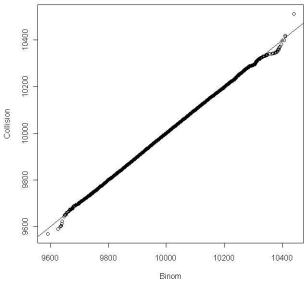
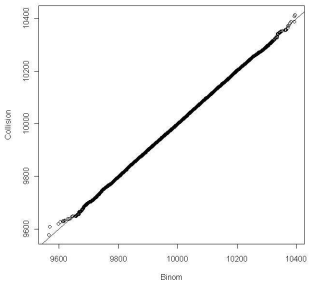
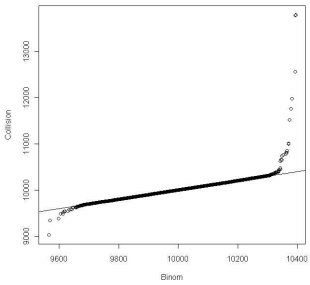
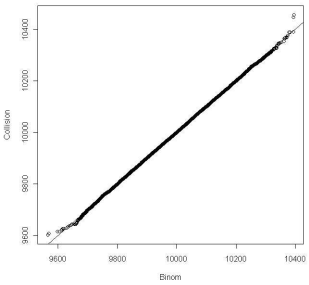
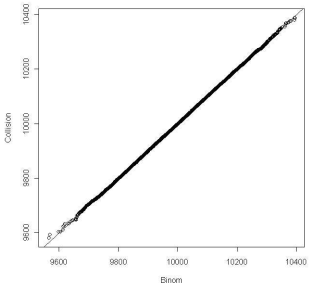
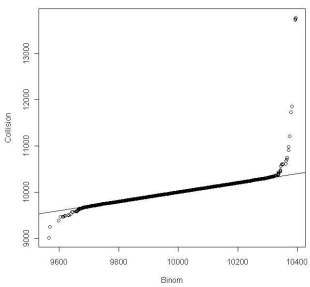
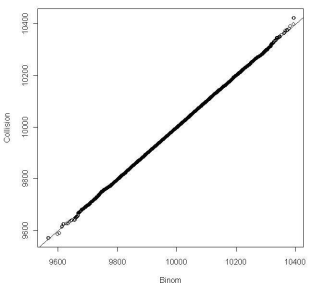
	10LSB	10MIB	10MSB
Pseudorandom 32-bit messages.			
Series 1C	p-value = 0.1139	p-value = 0.9219	p-value = 0.090
			
Cluster 32-bit messages: the initial pair of messages is derived pseudorandomly.			
Series 2C	p-value = 0.9015	p-value = 0.2727	p-value = 0.5111
			
Sparse and Cluster 32-bit messages: the initial pair of messages is $(1,2^8+1)$.			
Series 3C	p-value = 0.0730	p-value = 4.155e-10	p-value = 0.4952
			
Dense and Cluster 32-bit messages: the initial pair of messages is $(2^{32}-1, 2^{32}-257)$.			
Series 4C	p-value = 0.6578	p-value = 6.616e-12	p-value = 0.4497
			

Table 6.5: Chi-square and Q-Q plots of collision test results of 32-bit $digest_{XWM}(k,m)$ (input messages are XORed with two 32-bit halves of keys prior to multiplication): $N = 2^{16}$ pairs of 32-bit messages over $\|\mathcal{K}\| = 10240000$ distint keys. In each Q-Q plot, the vertical (Collision) and horizontal (Binom) axes denote quantiles on observed collision keys and respectively binomial distribution.

Test Series	No of keys $\|\mathcal{K}\|$	Map range	Expected Mean & Max numbers of collision keys, and Variance	Observed Mean μ'	Observed Max x'	Observed Variance σ'^2
			$\mu = \|\mathcal{K}\|\epsilon$ $\sigma^2 = \|\mathcal{K}\|\epsilon(1-\epsilon)$ $\int_{-\infty}^{x} \epsilon^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = 2^{-1/N}$	$\Sigma_{i=1}^{N} x_i/N$	max of $\{x_1 \ldots x_N\}$	$\Sigma_{i=1}^{N} \frac{(x_i-\mu')^2}{N}$
Pseudorandom 32-bit messages.						
1A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	99.96 100.00 100.10	141 147 143	99.5 98.1 100.1
1B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	1000.01 999.89 1000.05	1136 1148 1143	997.2 1010.0 996.6
1C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	9999.82 10000.52 9999.73	10389 10474 10439	10007.3 10033.5 9996.1
Cluster 32-bit messages: the initial pair of messages is derived pseudorandomly.						
2A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	99.98 100.01 100.02	150 144 148	100.1 99.7 99.1
2B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	999.91 999.98 1000.00	1159 1132 1142	997.6 999.6 994.5
2C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	9999.89 10000.17 9999.82	10485 10417 10430	9937.5 10037.8 10073.5
Sparse and Cluster 32-bit messages: the initial pair of messages is $(1,2^8+1)$.						
3A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	100.01 100.02 100.14	147 143 147	99.6 98.4 99.2
3B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	999.84 1000.04 999.79	1129 1359 1133	1002.1 1010.3 1002.1
3C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	9999.58 10000.69 10002.48	10427 13734 10425	10058.8 10834.1 9905.4
Dense and Cluster 32-bit messages: the initial pair of messages is $(2^{32}-1, 2^{32}-257)$.						
4A	102400	10LSB 10MIB 10MSB	100 - 142 - 99.90	99.99 99.95 100.13	145 144 145	98.7 98.8 99.4
4B	1024000	10LSB 10MIB 10MSB	1000 - 1134 - 999.0	999.70 1000.01 999.95	1142 1374 1143	990.1 1004.7 994.4
4C	10240000	10LSB 10MIB 10MSB	10000 - 10424 - 9990	9999.48 9999.25 10001.97	10398 13707 10503	10044.7 11166.1 9926.0

Table 6.6: Comparing the mean, max (no of keys) and variance of collision test results of 32-bit $digest_{XWM}(k,m)$ (input messages are XORed with two 32-bit halves of the key prior to multiplication) against their expected values in an ideal binomial distribution: $N = 2^{16}$ pairs of 32-bit messages. The set $\{x_1 \ldots x_N\}$ denotes the number of collision keys observed and counted for all N trials.

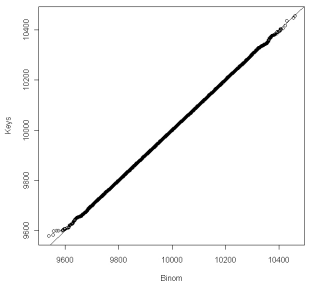
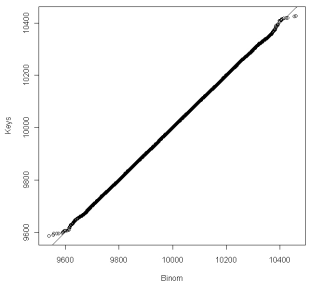
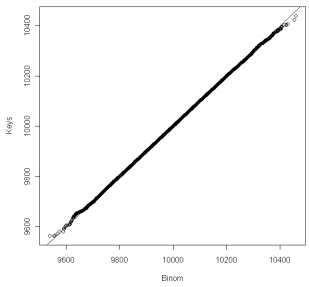
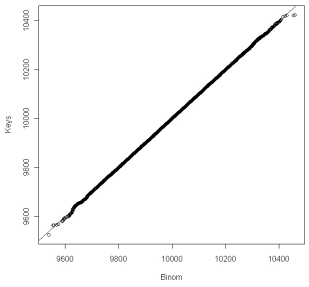
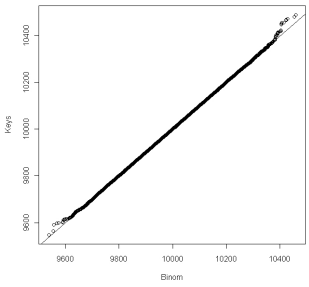
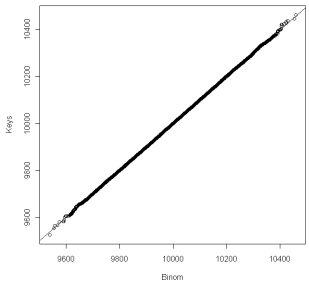
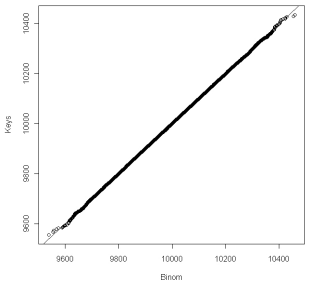
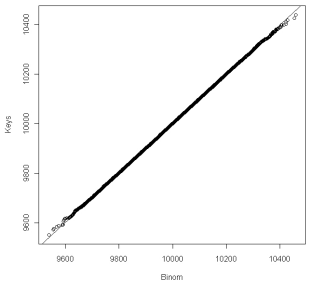
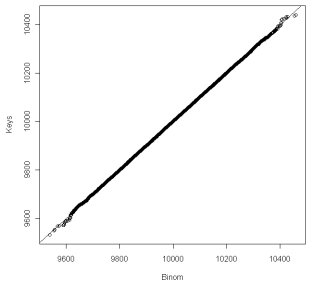
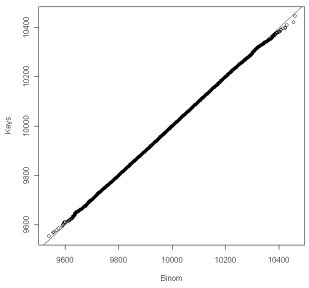
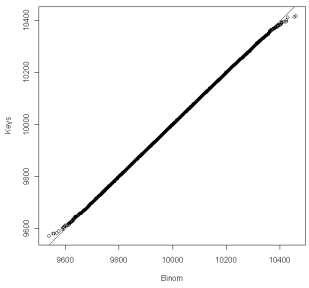
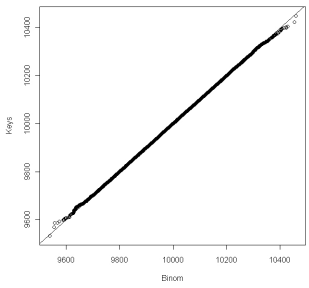
	10LSB	10MIB	10MSB
Pseudorandom 32-bit messages.			
Series 1C	p-value = 0.1577	p-value = 0.3704	p-value = 0.1214
			
Cluster 32-bit messages: the initial message is derived pseudorandomly.			
Series 2C	p-value = 0.7553	p-value = 0.8318	p-value = 0.934
			
Sparse and Cluster 32-bit messages: the initial message is 1.			
Series 3C	p-value = 0.1064	p-value = 0.9174	p-value = 0.6268
			
Dense and Cluster 32-bit messages: the initial message is $2^{32} - 1$.			
Series 4C	p-value = 0.0003124	p-value = 0.5064	p-value = 0.6865
			

Table 6.7: Chi-square and Q-Q plots of distribution test results of 32-bit $digest_{XWM}(k, m)$ (input messages are XORed with two 32-bit halves of keys prior to multiplication): $N = 2^{18}$ pairs of 32-bit messages over $\|\mathcal{K}\| = 10240000$ distint keys. In each Q-Q plot, the vertical and horizontal axes denote quantiles on observed (collision) keys and respectively binomial distribution.

Test Series	No of keys $\|\mathcal{K}\|$	Map range	Expected Mean & Max numbers of keys, and Variance	Observed Mean μ'	Observed Max y'	Observed Variance σ'^2
			$\mu = \|\mathcal{K}\|\epsilon$ $\sigma^2 = \|\mathcal{K}\|\epsilon(1-\epsilon)$ $\int_{-\infty}^{y} \epsilon^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = 2^{-1/N}$	$\Sigma_{i=1}^{N} y_i/N$	max of $\{y_1 \dots y_N\}$	$\Sigma_{i=1}^{N} \frac{(y_i-\mu')^2}{N}$
Pseudorandom 32-bit messages.						
1A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	151 151 158	100.34 99.76 99.71
1B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1146 1160 1165	995.60 999.70 1002.67
1C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10456 10428 10439	10031.05 9968.10 9907.97
Cluster 32-bit messages: the initial message is derived pseudorandomly.						
2A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	146 149 148	99.80 99.67 100.06
2B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1142 1141 1150	997.07 997.58 1000.52
2C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10425 10490 10463	10038.38 9983.42 10009.36
Sparse and Cluster 32-bit messages: the initial message is 1.						
3A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	149 151 153	99.67 100.12 99.94
3B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1149 1150 1137	996.76 999.42 997.70
3C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10436 10440 10440	9989.51 9960.43 9989.53
Dense and Cluster 32-bit messages: the initial message is $2^{32} - 1$.						
4A	102400	10LSB 10MIB 10MSB	100 - 145 - 99.90	100.00 100.00 100.00	153 147 147	100.15 99.79 99.78
4B	1024000	10LSB 10MIB 10MSB	1000 - 1144 - 999.0	1000.00 1000.00 1000.00	1144 1145 1150	1001.88 1002.80 995.98
4C	10240000	10LSB 10MIB 10MSB	10000 - 10445 - 9990	10000.00 10000.00 10000.00	10447 10418 10450	9982.14 9987.62 10011.10

Table 6.8: Comparing the mean, max (no of keys) and variance of distribution test results of 32-bit $digest_{XWM}(k, m)$ against their expected values in an ideal binomial distribution: $N = 2^{18}$ distinct pairs of message and digest output (m, d). The set $\{y_1 \dots y_N\}$ denotes the number of keys observed and counted for all N pairs (m, d).

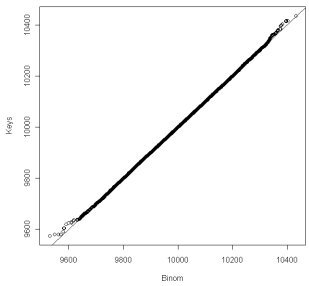
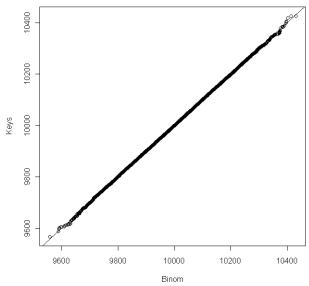
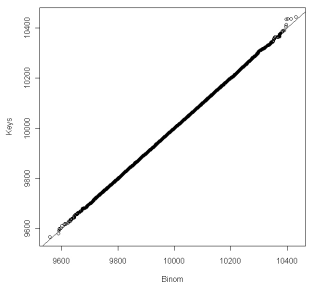
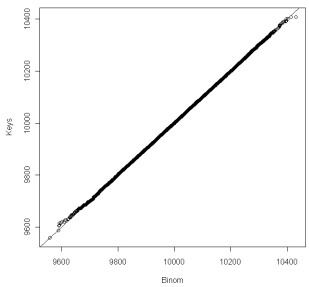
Pseudorandom 32-bit messages	
Series 1C	p-value = 0.1684 
Cluster 32-bit messages: the initial message is derived pseudorandomly.	
Series 2C	p-value = 0.283 
Sparse and Cluster 32-bit messages: the initial message is 1.	
Series 3C	p-value = 0.627 
Dense and Cluster 32-bit messages: the initial message is $2^{32} - 1$.	
Series 4C	p-value = 0.227 

Table 6.9: Chi-square and Q-Q plots of distribution test results of 32-bit $digest_T(k, m)$: $N = 2^{17}$ pairs of 32-bit messages over $\|\mathcal{K}\| = 10240000$ keys. In each Q-Q plot, the vertical and horizontal axes denote quantiles on observed keys and respectively binomial distribution.

Test Series	No of 64-bit random keys $\|\mathcal{K}\|$	Expected Mean & Max numbers of keys, and Variance	Observed Mean μ'	Observed Max x'	Observed Variance σ'^2
		$\mu = \|\mathcal{K}\|\epsilon$ $\sigma^2 = \|\mathcal{K}\|\epsilon(1-\epsilon)$ $\int_{-\infty}^{x} \epsilon^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = 2^{-1/N}$	$\Sigma_{i=1}^{N} y_i/N$	maximum of $\{y_1 \dots y_N\}$	$\Sigma_{i=1}^{N} \frac{(y_i-\mu')^2}{N}$
Pseudorandom 32-bit messages.					
1A	102400	100 - 144 - 99.90	100.00	145	100.19
1B	1024000	1000 - 1139 - 999.0	1000.00	1147	1002.38
1C	10240000	10000 - 10439 - 9990	10000.00	10436	9997.77
Cluster 32-bit messages: the initial message is derived pseudorandomly.					
2A	102400	100 - 144 - 99.90	100.00	148	99.95
2B	1024000	1000 - 1139 - 999.0	1000.00	1152	1003.82
2C	10240000	10000 - 10439 - 9990	10000.00	10427	9920.94
Sparse and Cluster 32-bit messages: the initial message is 1.					
3A	102400	100 - 144 - 99.90	100.00	145	99.53
3B	1024000	1000 - 1139 - 999.0	1000.00	1130	982.61
3C	10240000	10000 - 10439 - 9990	10000.00	10445	9949.84
Dense and Cluster 32-bit messages: the initial message is $2^{32} - 1$.					
4A	102400	100 - 144 - 99.90	100.00	143	100.16
4B	1024000	1000 - 1139 - 999.0	1000.00	1143	998.97
4C	10240000	10000 - 10439 - 9990	10000.00	10409	9910.73

Table 6.10: Comparing the mean, max (no of keys) and variance of distribution test results of 32-bit $digest_T(k, m)$ (Toeplitz method) against their expected values in an ideal binomial distribution: $N = 2^{17}$ pairs of 32-bit messages. The set $\{y_1 \dots y_N\}$ denotes the number of collision keys observed and counted for all N trials.

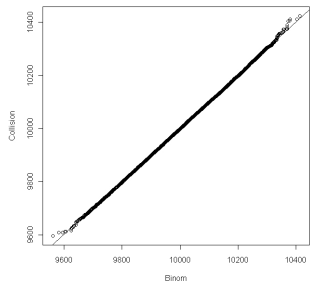
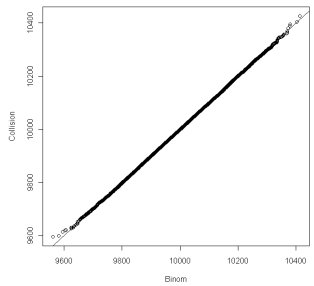
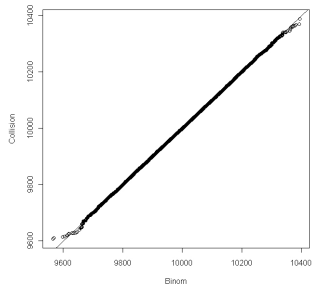
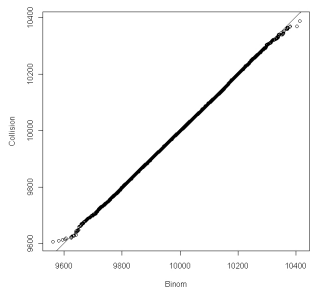
Pseudorandom 32-bit messages	
Series 1C	p-value = 0.4752 
Cluster 32-bit messages: the initial pair of messages is derived pseudorandomly.	
Series 2C	p-value = 0.2469 
Sparse and Cluster 32-bit messages: the initial pair of messages is $(1,2^8+1)$.	
Series 3C	p-value = 0.05804 
Dense and Cluster 32-bit messages: the initial pair of messages is $(2^{32}-1, 2^{32}-257)$.	
Series 4C	p-value = 0.06694 

Table 6.11: Chi-square and Q-Q plots of collision test results of 32-bit $digest_T(k,m)$: $N = 2^{16}$ pairs of 32-bit messages over $\|\mathcal{K}\| = 10240000$ keys. In each Q-Q plot, the vertical and horizontal axes denote quantiles on observed keys and respectively binomial distribution.

Test Series	No of 64-bit random keys $\|\mathcal{K}\|$	Expected Mean & Max number of collision keys and Variance	Observed Mean μ'	Observed Max x'	Observed Variance σ'^2
		$\mu = \|\mathcal{K}\|\epsilon$ $\sigma^2 = \|\mathcal{K}\|\epsilon(1-\epsilon)$ $\int_{-\infty}^{x} \epsilon^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = 2^{-1/N}$	$\Sigma_{i=1}^{N} x_i/N$	maximum of $\{x_1 \ldots x_N\}$	$\Sigma_{i=1}^{N} \frac{(x_i-\mu')^2}{N}$
Pseudorandom 32-bit messages.					
1A	102400	100 - 142 - 99.90	99.99	146	99.1
1B	1024000	1000 - 1134 - 999.0	1000.12	1153	997.6
1C	10240000	10000 - 10424 - 9990	10000.69	10425	9958.3
Cluster 32-bit messages: the initial pair of messages is derived pseudorandomly.					
2A	102400	100 - 142 - 99.90	100.04	146	99.4
2B	1024000	1000 - 1134 - 999.0	1000.09	1144	993.3
2C	10240000	10000 - 10424 - 9990	10000.31	10409	10016.5
Sparse and Cluster 32-bit messages: the initial pair of messages is $(1,2^8+1)$.					
3A	102400	100 - 142 - 99.90	100.20	142	99.6
3B	1024000	1000 - 1134 - 999.0	999.85	1137	1003.7
3C	10240000	10000 - 10424 - 9990	9998.26	10444	10005.6
Dense and Cluster 32-bit messages: the initial pair of messages is $(2^{32}-1, 2^{32}-257)$.					
4A	102400	100 - 142 - 99.90	100.06	142	99.7
4B	1024000	1000 - 1134 - 999.0	1000.16	1127	1005.6
4C	10240000	10000 - 10424 - 9990	9999.42	10461	9940.6

Table 6.12: Comparing the mean, max (no of keys) and variance of collision test results of 32-bit $digest_T(k,m)$ (Toeplitz method) against their expected values in an ideal binomial distribution: $N = 2^{16}$ pairs of 32-bit messages. The set $\{x_1 \ldots x_N\}$ denotes the number of collision keys observed and counted for all N trials.

variables can be used to generate digest functions with the properties we require. Secondly, a structural similarity between Toeplitz matrix and the "school book" algorithm for integer long multiplications exists as pointed out in Section 6.4. This similarity leads us to introduce more efficient digest computation based on word multiplications which can be calculated fast in any processors. Although there can be bias due to carry bits in an integer multiplication, we have conducted statistical and collision tests to show that no significant difference seems to exist between our two digest constructions.

It is clear that both of our digest constructions ($digest_T(k, m)$ and $digest_{WM}(k, m)$) permit parallel computation in multi-core processors to increase efficiency. This is however only possible if the key k is either really long, i.e. the same order as message bitlength $r = K + b - 1$, or derived (off-line) from a short number of bits in advance. We note that there is a theoretical bound on the digest (or universal hash) key length which says that the message length can grow exponentially with the key length: $r \geq \log(K/b)$ [51]. This suggests that other digest constructions which require shorter key length might exit.

Chapter 7

New bounds for almost universal hash functions

We showed in Sections 5.2 that a well-studied one-way authentication protocol, termed MANA I [36, 119], is neither optimal in the human work nor as secure as had been previously believed. The latter weakness comes from the fact that the key fed into the digest functions (a variant on universal hash functions which are used in the protocol) is too short relative to the digested message bitlength, the digest output bitlength and the collision probability. That work introduces us to the area of theoretical bounds on the key-length for universal hash functions — a topic which has been studied extensively to date by many research authors. For example, Carter and Wegman introduced the area [25, 120] in 1979, and then many other researchers, including Stinson [110, 111], Gemmell [39], Johansson [51], Kabatianskii [52], to name just a few, reported a number of bounds for almost (strongly) universal hash functions.

In this chapter we use the *pigeon-hole* principle to introduce a new bound for an *almost* universal hash function (AU), as defined in Definition 8 of Chapter 2. This result tells us the lower bound on the bitlength of the hash key in terms of a fixed amount of information we want to hash, the hash output bitlength and the hash collision probability ϵ.

Although there has been much work in this area, most researchers concentrate on more restrictive versions of AU known as *almost strongly* and *almost XOR* universal hash functions (ASU and AXU), which are used to construct Message Authentication Codes (MACs) in practice [54, 55, 120]. We however believe that there is a similar potential for AU. For example, a new class of authentication schemes, based on new concepts of trust derived from human actions and interactions, has been

considered in Chapters 3, 4 and 5 of this thesis to replace PKI and passwords in pervasive computing environments. Some of these protocols make use of a new cryptographic *digest* function introduced in Chapter 6 and [79], with similar security properties and purposes to an AU. In these protocols, digest or hash keys are always random and fresh in each protocol session, and so a substitution attack, which relies on the reuse of a hash key for multiple messages (as in MACs), is irrelevant. What we then require is protection against hash collision attacks (AU) as opposed to substitution attacks (ASU).

Moreover, since universal hash keys in MAC are often large, one reuses a single secret key for multiple messages as mentioned above. This opens the way for key recovery and universal forgery attacks which exploit weak key properties or partial information on a secret key; such attacks have been recently reported by Handschuh and Preneel [45]. Avoiding reusing keys would render most key recovery attacks useless, and so it is desirable to construct universal hash functions with short keys, which in turn generate the need to calculate the lower bound of universal hash key length.

Johansson et al. [51] give an equivalence between *almost* universal hash functions and error-correcting codes. This implies that every bound of coding theory potentially corresponds to another bound for universal hash functions, and vice versa. We will therefore show how to use the *Singleton* bound [62] to derive a different AU-bound which is, however, not as tight as our new bound. In particular, there exists a subclass of an AU which cannot be transformed into an equivalent code that satisfies the Singleton bound with equality, thus the Singleton bound does not give a tight result for the subclass of universal hash functions. To the best of our knowledge, this is the first time a bound for an almost universal hash function has been demonstrated to be better than a similar bound which is derived directly from coding theory bounds. Perhaps paradoxically, we find that this does *not* imply that the Singleton bound can be improved.

In comparing the new AU-bound to Stinson's AU-bound [110, 111], we demonstrate the significance of the value $(1 + \frac{b}{K-b})2^{-b}$: as ϵ increases beyond this threshold, our bound is tighter than Stinson's AU-bound. Subsequently this threshold value will be shown to have the same theoretical significance in relationships between known AXU- and ASU-bounds. What this illustrates is a behaviour of any universal hash functions, known as the 'Wegman-Carter effect' in the literature [19, 57], previously reported in [51, 52] by Johansson, Kabatianskii and Smeets: if ϵ exceeds 2^{-b} (the theoretical minimum[1]) by an arbitrarily small positive value, then the total number of messages, that can be authenticated, grows exponentially with the number of keys provided, but if $\epsilon = 2^{-b}$ it only grows linearly. However, while these authors only demonstrate this behaviour asymptotically, we are able

[1]In practice, the minimum collision probability of an AU is $\frac{2^K-2^b}{2^{K+b}-2^b}$, which is less than 2^{-b}. This occurs in an *optimally universal* hash scheme introduced by Sarwate [101].

to *quantify* it using the threshold value.

We end this chapter by proving the *optimality* of polynomial hashing over finite fields [21, 51, 115] in building an AU, i.e. the construction meets the new AU-bound with equality. This therefore extends the work of Johansson, Kabatianskii and Smeets [51, 52], where the authors proved the *asymptotic optimality* of polynomial hashing as an ASU.

In our work, we also introduce a new bound for an AXU, which can be derived from Kabatianskii's ASU-bound [52] and a connection between ASU and AXU [35, 120]. This bound is then rigorously analysed in relation to other known bounds and will be shown to be met with equality in the second version of polynomial hashing.

7.1 Notations and definitions of universal hash functions

In this chapter, all formulas are expressed in terms of the generalised bitlengths of hash keys, input messages and hash output instead of the cardinalities of the sets of these parameters (2^r, 2^K and 2^b) as in other chapters. Although the bitlengths are often integers in practice, our result reported here applies to both integer and non-integer bitlengths. The advantage of the notation will become clear when we explain why the use of the pigeon-hole principle yields better bounds than the use of coding theory bounds in Section 7.2.2.

We have defined universal hash functions and *almost* universal hash functions in Section 2.3.2: the following will give definitions of two other families of universal hash functions discussed in this chapter. Here ϵ, which is sometimes written as $2^{\theta-b} = \gamma 2^{-b}$, is referred to as the collision, differential or interpolation probability associated with ϵ-AU, ϵ-AXU or ϵ-ASU, respectively.[2]

Definition 21 [54, 55, 110] An ϵ-*almost XOR* universal hash function ϵ-AXU (r, K, b) is a set of 2^r hash functions $h_k()$ where $k \in R$ such that $h_k : X \rightarrow Y$; here $R = \{1...2^r\}$, $X = \{1...2^K\}$ and $Y = \{1...2^b\}$. Moreover, for every m, $\hat{m} \in X$ $(m \neq \hat{m})$ and any $\omega \in Y$, we have:

$$\Pr_{\{k \in R\}}[h_k(m) \oplus h_k(\hat{m}) = \omega] \leq \epsilon$$

Definition 22 [110, 120] An ϵ-*almost strongly* universal hash function ϵ-ASU (r, K, b) is a set of 2^r hash functions $h_k()$ where $k \in R$ such that $h_k : X \rightarrow Y$; here $R = \{1...2^r\}$, $X = \{1...2^K\}$ and $Y = \{1...2^b\}$. Moreover, we have:

[2] The terms collision, differential and interpolation probabilities were introduced by Bernstein in the appendix of [15] to distinguish the differences between these families of universal hash functions.

- for every $m \in X$ and $y \in Y$, $\Pr_{\{k \in R\}}[h_k(m) = y] = 2^{-b}$
- for every $m, \hat{m} \in X$ $(m \neq \hat{m})$ and for every $y, \hat{y} \in Y$: $\Pr_{\{k \in R\}}[h_k(m) = y, h_k(\hat{m}) = \hat{y}] \leq \epsilon 2^{-b}$

All *universal* hash functions discussed to date are pairwise, since we look at their properties in relation to two different messages. We will see that the new bound, and its proof, can be easily adapted to a more general version of AU, termed a l-wise AU_l, and therefore we give the definition below. We argue that not only is this of theoretical interest to study AU_l, but also useful in many applications. For example, in the new group authentication protocols discussed in the introduction, the intruder always attempts to fool multiple parties into accepting different versions of a piece of data that the protocol seeks to ensure they agree on. It is therefore desirable that we consider the possibility of a hash collision with respect to more than two different input messages. However, unless indicated, our work presented in this chapter always refers to pairwise universal hash functions.

Definition 23 A l-wise ϵ-*almost* universal hash function ϵ-AU_l (r, K, b) is a set of 2^r hash functions $h_k()$ where $k \in R$ such that $h_k : X \rightarrow Y$; here $R = \{1...2^r\}$, $X = \{1...2^K\}$ and $Y = \{1...2^b\}$. Moreover, for any l different messages $\{m_1, \ldots, m_l\}$, we have:

$$\Pr_{\{k \in R\}}[h_k(m_1) = \cdots = h_k(m_l)] \leq \epsilon$$

We assume the input message bitlength K is significantly greater than the hash bitlength b. Whenever we use the term $\log X$, we refer to the base-2 logarithm to simplify the notation.

7.2 Bounds for almost universal hash functions

We first present a new bound for an AU using the pigeon-hole principle. Subsequently, the Singleton bound [62] in coding theory will be used to derive another AU-bound,[3] which is not as tight as we had expected. In particular, when K is *not* an integer multiple of b, the new AU-bound is tighter (greater) than the latter. This result applies to both integer and non-integer values of K and b.

This therefore demonstrates that a universal hash function bound derived from coding theory might only fit certain of universal hash function parameters tightly, and for further parameter values the bound is based on the best conservative approximate that does fit coding theory. What we will discover is that the use of the pigeon-hole principle can provide a tighter bound for this second class.

[3] Although several bounds in coding theory have been transformed into equivalent bounds for universal hash functions, e.g. Plotkin [111] or Johnson bounds [52], to the best of our knowledge, the Singleton bound has never been used.

7.2.1 New AU-bound

Theorem 8 If there exists an ϵ-AU (r, K, b) then

- when K is an integer multiple of b: $r \geq \log_2 \left(\epsilon^{-1} \left(\frac{K}{b} - 1\right)\right)$
- when K is not an integer multiple of b:[4] $r \geq \log_2(\epsilon^{-1} \lfloor \frac{K}{b} \rfloor)$

Proof We make use of the *pigeon-hole* principle: given two positive integers n and m, if n items are put into m holes then at least one hole must contain more than or equal to $\lceil n/m \rceil$ items.

Let assume $K = bt + b'$, where t is an integer and $0 \leq b' < b$.

For any key k_1, there exists a hash value h_1 such that there are at least $\lceil 2^{K-b} \rceil$ distinct messages all hashing to h_1 under the same key k_1, thanks to the pigeon-hole principle. For any choice of k_2 other than k_1, there will also be a collection of at least $\lceil 2^{K-2b} \rceil$ of these messages mapping to some hash value h_2 under k_2. Repeating this process $t-1$ times will result in at least $\lceil 2^{K-(t-1)b} \rceil = \lceil 2^{b+b'} \rceil$ distinct messages that hash to the same values under $t-1$ keys, leading to two possibilities.

- When $b' = 0$ or K is an integer multiple of b, we *cannot* repeat this process any further because at least 2 distinct messages must be left after these iterations. Thus a family of hash functions is ϵ-*almost universal* when $t-1$ is smaller than or equal to ϵ portion of the key space: $\epsilon 2^r \geq t - 1 = K/b - 1$, which means that $r \geq \log(\epsilon^{-1}(K/b - 1))$
- When $b' > 0$ or K is *not* an integer multiple of b, we have $\lceil 2^{b+b'} \rceil \geq 2^b + 1$. Repeating this process for one more key will end up with at least 2 distinct messages that map to the same values under $t = \lfloor K/b \rfloor$ keys. As a result, we have $\epsilon 2^r \geq \lfloor K/b \rfloor$, which means that $r \geq \log(\epsilon^{-1} \lfloor K/b \rfloor)$ ∎

The proof of the new bound for a pairwise AU_2 can be generalised to derive the corresponding bound for a l-wise AU_l, for any integer $l \geq 2$. Instead of leaving at least 2 distinct messages after these iterations as shown in the proof of Theorem 8, we need to leave at least l distinct messages. Similar analysis leads to the following theorem.

Theorem 9 If there exists a l-wise ϵ-AU_l (r, K, b) and $K = bt + b'$, where t is an integer, then

- when $0 \leq b' \leq \log(l-1)$: $r \geq \log \left(\epsilon^{-1} \left(\lfloor \frac{K}{b} \rfloor - 1\right)\right)$
- when $\log(l-1) < b' < b$: $r \geq \log \left(\epsilon^{-1} \lfloor \frac{K}{b} \rfloor\right)$

[4] From now on, we will drop subscript '2' in $\log_2$ to simplify the notation.

Although there has been some study of l-wise *almost strongly* universal hash functions by Stinson [112] and Kurosawa et al. [56], as far as we are aware, this is the first result on l-wise *almost* universal hash functions.

We end this section with another observation: there is no restriction on any of the parameters, i.e. the generalised bitlengths (K, b, r), in both our pairwise and l-wise new AU-bounds, which makes them more attractive than a similar ASU-bound introduced by Kabatianskii et al. [52], as will be discussed in the sections to come.

7.2.2 Error-correcting codes and almost universal hash functions

While the connection between almost universal hash functions and error-correcting codes (i.e. see Theorem 10), which was first observed by Johansson et al. [51], has widely been used to derive tight bounds for universal hash functions [51, 52, 110, 111], the following comparative analysis will demonstrate that this strategy does not always give the best answer.

Let (n, T, d, q) be a q-ary error-correcting code, where n is the codeword length in symbols, T is the total number of codewords, and the minimum Hamming distance is d.

Theorem 10 [18, 51, 111]. If there exists an ϵ-AU (r, K, b), then there exists an $(n = 2^r, T = 2^K, d = n - \epsilon 2^r, q = 2^b)$ code. Conversely, if there exists an (n, T, d, q) code, then there exists an $(\epsilon = 1 - d/n)$-AU $(r = \log n, K = \log T, b = \log q)$.

We note that for the conversion from an AU into a code to work, the resulting coding parameters (n, T, d, q) must be integers. Using the connection, we can derive another AU-bound from the Singleton bound.

Singleton bound [62]: given an (n, T, d, q) code then $q^{n-d+1} \geq T$.

Theorem 11 Another bound for an ϵ-AU (r, K, b) is: $r \geq \log\left(\epsilon^{-1}\left(K/b - 1\right)\right)$

Proof Using Theorem 10, construct an $(n = 2^r, T = 2^K, d = n - \epsilon 2^r, q = 2^b)$ code from the universal hash function ϵ-AU (r, K, b). This code must satisfy the Singleton bound, so we obtain:

$$\begin{aligned} q^{n-d+1} &\geq T \\ 2^{b(\epsilon 2^r + 1)} &\geq 2^K \\ r &\geq \log\left(\epsilon^{-1}\left(K/b - 1\right)\right) \end{aligned}$$

∎

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
k_1	1	2	3	4	1	2	3	4
k_2	2	3	4	1	3	4	1	2

Table 7.1: A construction of an $(\epsilon = 1/2)$-*AU* $(1, 3, 2)$, in which there are $2^K = 8$ input messages $\{m_1, \ldots, m_8\}$ mapping onto $2^b = 4$ hash outputs under $2^r = 2$ hash keys $\{k_1, k_2\}$.

When K is an integer multiple of b, this is equivalent to the new *AU*-bound in Theorem 8.

In contrast, when K is not an integer multiple of b, the new *AU*-bound is clearly tighter (or greater) than the one derived from coding theory in Theorem 11. In this case any set of universal hash function parameters (ϵ, r, K, b) which achieves equality in the bound in Theorem 11 *cannot* be converted into an equivalent set of coding parameters (n, T, d, q), where both n (the codeword length) and d (the minimum Hamming distance) are integers.[5] Hence, it is impossible to construct an *AU* that meets the *AU*-bound based on the Singleton bound with equality when K is not an integer multiple of b.

For example, when $K = 3$, $b = 2$ and $\epsilon = 1/2$, the *AU*-bound defined in Theorem 11 gives $2^r \geq \epsilon^{-1}(K/b - 1) = 1$, which is not tight because it is impossible to construct such an *AU* with a single key.[6] The new *AU*-bound from Theorem 8, on the other hand, gives $2^r \geq \epsilon^{-1} \lfloor K/b \rfloor = 2$ corresponding to an $(\epsilon = 1/2)$-*AU* $(r = 1, K = 3, b = 2)$ (i.e. see Table 7.1) or an $(n = 2, T = 8, d = 1, q = 4)$ code.

One might note that any *AU*-bound is also a bound for error correcting codes. However when we convert the new bound into the parameters in coding, the result is, perhaps surprisingly, no better than the Singleton bound as demonstrated below.

- When K is an integer multiple of b, the two bounds are equivalent thanks to the above analysis.
- When $K = tb + b'$, where t is an integer and $0 < b' < b$. The new *AU*-bound is equivalent to: $n - d = \epsilon 2^r \geq \lfloor K/b \rfloor = t$, and so $q^{n-d} 2^{b'} \geq T = 2^{tb+b'}$. Since the number of codewords T must be an integer and $1 < 2^{b'} < q$, we have $q^{n-d+1} - 1 \geq T$.

Since the Singleton bound determines the maximum size $T = 2^K$ of a $(q = 2^b)$-ary code of length n and minimum Hamming distance d, there is no point to distinguish whether $K = \log T$ is an integer multiple of $b = \log q$ or not in the coding parameters. Hence, our new *AU*-bound when being transformed into coding theory becomes equivalent to the Singleton bound.

[5] Assume $K = tb + b'$ where $0 < b' < b$ and t is an integer. If equality in the bound derived in Theorem 11 is achieved, then $\epsilon 2^r = t - 1 + b'/b$. Using the equivalence between *AU* and error-correcting code in Theorem 10, we further have $n - d = \epsilon 2^r = t - 1 + b'/b$. Since b'/b is not integer, n and d cannot be integers at the same time.

[6] We consider the probability of hash collision ϵ as the key varies uniformly over its domain. Since there are $2^K = 8$ different messages mapping onto $2^b = 4$ different hash outputs under a single key, $\epsilon = 1 > 1/2$.

7.3 The significance of the threshold value of ϵ

We compare the new AU-bound introduced in Theorem 8 with other bounds for not only AU but also AXU and ASU to understand the accuracy and significance of our result. This comparative analysis also uncovers the importance of the value $\epsilon = (1 + \frac{b}{K-b})2^{-b}$ which represents a *threshold* in the behaviour of bounds, and therefore, for the first time, quantifies the *Wegman-Carter* effect.

In addition, we introduce a new AXU-bound derived from the ASU-bound of Kabatianskii et al. [52] and a connection between AXU and ASU due to Wegman and Carter [120].

7.3.1 Comparison between the new and other AU-bounds

Stinson's AU-bound, which can be derived from the Plotkin bound in coding theory [111], is as follows: $2^r \geq \frac{2^K(2^b-1)}{2^K(\epsilon 2^b-1)+2^{2b}(1-\epsilon)}$. When $\epsilon = 2^{-b}$, this is much tighter than the new AU-bound for then it gives $r \geq K - b$, which means that the key bitlength grows at least linearly with the message bitlength. In contrast, as we increase ϵ to 2^{1-b} then setting $r = b$ satisfies the bound, i.e. the key needs be no longer than the bitlength of the hash.

To understand the dramatic collapse, we present a different way to interpret the formula when $\epsilon = \gamma 2^{-b} > 2^{-b}$, which is the same as $\gamma > 1$.

$$2^r \geq \frac{2^K(2^b-1)}{2^K(\gamma-1)+2^{2b}(1-\gamma 2^{-b})} = \frac{2^b-1}{(\gamma-1)+2^{2b-K}(1-\gamma 2^{-b})}$$

Note that since both terms in the denominator of the right-hand form are positive for $\gamma > 1$, with the second one converging to 0 as K increases, no matter how big K gets it can never prove a stronger lower bound on r than

$$r > \log \frac{2^b}{\gamma - 1} = b + \log \frac{1}{\gamma - 1}$$

In other words, while the new bound grows in proportion to $log\, K$, this bound is essentially constant as K increases. Hence there comes a point as K and ϵ increase where Stinson's bound becomes weaker than the new one. In order to locate that point, we find the value of ϵ above which ours is greater than Stinson's. To simplify the calculation, we will round up the new AU-bound to $(2^r \geq \frac{K}{\epsilon b})$. This gives a very good approximation to the crucial value.

$$\frac{K}{\epsilon b} \quad > \quad \frac{2^K\left(2^b-1\right)}{2^K\left(\epsilon 2^b-1\right)+2^{2b}(1-\epsilon)}$$

$$\epsilon \quad > \quad \frac{K2^K - K2^{2b}}{K2^{K+b} - K2^{2b} - b2^{K+b} + b2^K}$$

Since $2^{2b} \ll 2^K \ll 2^{K+b}$, the above can be approximated as follows:

$$\epsilon > \frac{K2^K}{K2^{K+b} - b2^{K+b}} = \frac{K}{(K-b)2^b} = \left(1 + \frac{b}{K-b}\right) 2^{-b}$$

We therefore refer to $\left(1 + \frac{b}{K-b}\right) 2^{-b}$ as the *threshold* value of ϵ. Since K is always assumed to be significantly bigger than b, Stinson's AU-bound can only be tight within a very short range of ϵ. Moreover, the difference between the threshold value and 2^{-b}, which is $\frac{b}{(K-b)2^b}$, can be made as small positively as we want. This implies that if ϵ exceeds 2^{-b} by an arbitrarily small positive value the message bitlength grows at most exponentially with the key bitlength as demonstrated in the new AU-bound, but if $\epsilon = 2^{-b}$ it will grow at most linearly as shown in Stinson's AU-bound.

While the same asymptotic behaviour has also been derived from a relation between ASU and codes correcting independent errors by Johansson et al. [51, 52], it is not clear to us how we can derive the same threshold value of ϵ from the strategy used by Johansson et al. As a consequence, our approach of deriving the result quantitatively demonstrates three further important points:

- If we fix the bitlengths of an input message and a hash output then Stinson's AU-bound is still useful when $2^{-b} < \epsilon < \left(1 + \frac{b}{K-b}\right) 2^{-b}$. See Table 7.2 for more information.
- Given any value of ϵ which exceeds 2^{-b} by an arbitrarily small positive value, we can determine the threshold of input messages' bitlength ($K \geq b + \frac{b}{2^b\epsilon - 1}$) above which the message bitlength can apparently start to grow exponentially with the key bitlength, i.e. the new AU-bound gives a better estimate than Stinson's AU-bound.
- The threshold value of ϵ, perhaps surprisingly, has the same theoretical importance when we visit different ASU- and AXU-bounds in Appendix 7.3.3. See Table 7.2 for more information.

7.3.2 Comparison between the new AU-bound and known ASU-bounds

Since ASU is more restrictive than AU, intuitively we would expect that the number of bits required for the key in AU should be smaller than in ASU with respect to the same set of parameters (ϵ, K, b). This analysis is reflected by the following comparisons:

- When $\epsilon = 2^{-b}$, Stinson's AU-bound [111] ($r \geq K - b$) is smaller than Stinson's ASU-bound [110, 111][7] ($r \geq K + b - 1$) by $2b - 1$ bits. But when $\epsilon > 2^{-b}$, the gap gets closer as seen below.

[7] Stinson's ASU-bound can be derived from the second Johnson bound for constant weight binary codes [111].

- The new AU-bound in Theorem 8 is smaller than Kabatianskii's ASU-bound [52], $r \geq b + \log(\epsilon^{-1}\lfloor K/b \rfloor)$,[8] by at least b bits.

- The difference between the new AU-bound and Gemmell-Naor's ASU-bound [39],[9] $r \geq \log K + 2\log\epsilon^{-1} - \log\log\epsilon^{-1}$, gets very near to b when $\theta \ll b$: $\log\epsilon^{-1} + \log\frac{b}{\log\epsilon^{-1}} = b - \theta + \log\frac{b}{b-\theta}$

Coincidentally, it is known that if there exists an ϵ-AXU (r, K, b) then it can be used to construct an ϵ-ASU $(r + b, K, b)$, thanks to the work of Wegman and Carter [120], i.e. see Theorem 12.

Theorem 12 [35, 120]. Let $H = \{h_k() \mid k \in \{1...2^r\}\}$ be an ϵ-AXU (r, K, b),[10] then $\hat{H} = \{\hat{h}_{k,s}() \mid k \in \{1...2^r\},\ s \in \{1...2^b\},\ \text{and } \hat{h}_{k,s}() = h_k() \oplus s\}$ is an ϵ-ASU $(r + b, K, b)$.

Proof In this proof, we use the notation $\mathbf{Pr}_{k,s}$ to denote the probability of some condition being met as the pair (k, s) varies uniformly over its domain. For any message m and hash output y, we have

$$P_I = \mathbf{Pr}_{k,s}\left[\hat{h}_{k,s}(m) = y\right] = \mathbf{Pr}_{k,s}[h_k(m) \oplus s = y]$$

For any value of k, s is uniquely determined by $s = h_k(m) \oplus y$, and thus $P_I = \frac{2^r}{2^{r+b}} = 2^{-b}$.

For every pair of distinct messages $(m, \hat{m})$ and for every pair of hash outputs $(y, \hat{y})$, we have

$$P_S = \mathbf{Pr}_{k,s}\left[\hat{h}_{k,s}(m) = y, \hat{h}_{k,s}(\hat{m}) = \hat{y}\right] = \mathbf{Pr}_{k,s}\left[h_k(m) \oplus s = y, h_k(m) \oplus h_k(\hat{m}) = y \oplus \hat{y}\right]\}$$

For any value of k, s is uniquely determined by $s = h_k(m) \oplus y$. Since $h_k()$ is an ϵ-AXU (r, K, b) there are at most $\epsilon 2^r$ keys satisfying $h_k(m) \oplus h_k(\hat{m}) = y \oplus \hat{y}$, and thus $P_S \leq \frac{\epsilon 2^r}{2^{r+b}} = \epsilon 2^{-b}$. ∎

Applying Theorem 12 to Kabatianskii's ASU-bound, $r \geq b + \log(\epsilon^{-1}\lfloor K/b \rfloor)$, we can derive its AXU-variant as in the following theorem.

Theorem 13 For any ϵ-AXU (r, K, b): $r \geq \log(\epsilon^{-1}\lfloor K/b \rfloor)$, provided[11] $K/b < \sqrt{2^{r+1}(1 - 2^{-b})} - 1/2$

This theorem shows that AU-bound is strictly shorter than AXU-bound for some set of parameters (ϵ, K, b), i.e. when K is an integer multiple of b. This argument is consistent with the formal definitions, since AXU is a stronger definition of AU.

[8] Kabatianskii's ASU-bound, which is derived from the Johnson bound in Theorem 15 of [52], is valid when $K/b < \sqrt{2^{r-b+1}(1 - 2^{-b})} - 1/2$.

[9] We note that the bound was reported in the chapter of Gemmell and Noar [39] (Section 5.1). However, it was noted there that the bound was actually introduced by Noga Alon through private communication.

[10] We note that the AXU in this theorem does not need to be uniformly distributed as argued by Etzel et al. [35].

[11] As pointed out in footnote 8 and [52], there is a condition for the validity of Kabatianskii's ASU-bound, and therefore the same condition should apply to the AXU-variant of Kabatianskii's ASU-bound.

For example, when we set $\epsilon = 2^{-b}$, Stinson's AU-bound yields $K - b$ bits compared to K, derived from Stinson's AXU-bound ($2^r \geq \frac{2^K(2^b-1)}{2^b\epsilon(2^K-1)+2^b-2^K}$) [111].[12] We will see again that this comparative analysis is justified for larger values of ϵ when we visit constructions based on *polynomial hashing* over finite fields in Section 7.4.

7.3.3 Comparison between the new AU-bound and known AXU-bounds

We note that Stinson's bounds for AXU and ASU [111] have similar forms to his AU-bound [111]. Furthermore, the same similarity in form holds between Kabatianskii's ASU-bound [52], the AXU-variant of Kabatianskii's ASU-bound in Theorem 13, and the new AU-bound in Theorem 8. We therefore assert that the threshold value of ϵ has the same significance in the relationships between the two versions of ASU-bound, and of AXU-bound respectively.

The following calculation locates the value of ϵ above which Kabatianskii's ASU-bound becomes better than Stinson's ASU-bound.[13]

$$\begin{aligned} \frac{K2^b}{\epsilon b} &\geq \frac{2^K(2^b-1)^2}{2^b\epsilon(2^K-1)+2^b-2^K} \\ \epsilon &\geq \frac{K2^{b+K}-K2^{2b}}{K2^{2b+K}-K2^{2b}-b2^{2b+K}+b2^{b+K+1}-b2^K} \end{aligned}$$

Since $2^{2b} \ll 2^K \ll 2^{K+b}$ the above can be approximated as follows:

$$\epsilon > \frac{K2^{b+K}}{K2^{2b+K}-b2^{2b+K}} = \frac{K}{(K-b)2^b} = \left(1+\frac{b}{K-b}\right)2^{-b}$$

A similar calculation also leads us to conclude that Stinson's AXU-bound is overtaken by the AXU-variant of Kabatianskii's ASU-bound at the threshold value of ϵ.

A summary of the relation between all these different bounds for AU, AXU and ASU in relation to the threshold value of ϵ is given in Table 7.2.

[12] Stinson's AXU-bound is derived from the second Johnson bound for constant weight binary codes.

[13] Since the constant 1 in Stinson's ASU-bound ($2^r \geq 1 + \frac{2^K(2^b-1)^2}{2^b\epsilon(2^K-1)+2^b-2^K}$) is very small compared to 2^r, we will ignore it in subsequent analysis to simplify the calculation. In addition, we will round up Kabatianskii's ASU-bound from $2^r \geq \frac{2^b}{\epsilon}\lfloor K/b \rfloor$ to $2^r \geq \frac{2^bK}{\epsilon b}$.

7.4 The optimality of *polynomial hashing* as an *AU*

Polynomial hashing over finite fields was independently introduced by Boer [21], Johansson et al. [51], and Taylor [115]. To the best of our knowledge, polynomial hashing as an authentication code (ASU) has only been proved to be *asymptotically optimal* by Johansson et al. [51].[14]

Extending this result, we will show a different version of polynomial hashing which is designed as an AU is *optimal*, because it meets the new AU-bound in Theorem 8 with equality.

Fix some positive integer t. Let the set of all messages be $\{m = \langle m_1, \ldots, m_t \rangle; m_i \in \mathbb{F}_q\}$, here $b = \log q$ and the message bitlength is $K = tb = t \log q$.

In the first version of polynomial hashing as an AU, each message m will form a polynomial $m(x)$ of degree less than t over $\mathbb{F}_q$. For any key $k \in \mathbb{F}_q$, the universal hash of message m under key k is equivalent to $m(k)$ over $\mathbb{F}_q$.

$$h_k(m) = m(k) = m_1 + m_2 k + m_3 k^2 + \cdots + m_t k^{t-1}$$

If we fix two different messages A and $B = A + m$, then a hash collision is equivalent to: $0 = h_k(A) + h_k(B) = A(k) + B(k) = m(k)$. Since the polynomial $m(k)$ is of degree up to $t-1$, we have $\epsilon = (t-1)q^{-1} = (K/b - 1)2^{-r}$, and so $r = \log\left(\epsilon^{-1}(K/b - 1)\right)$. The equality in the new AU-bound implies optimality of polynomial hashing as an AU for any $K/b = t \in [2, q]$.

The construction above is not an AXU because if we set $\omega = A_1 + B_1$ and for all $i \in (1, t]$: $A_i = B_i = 0$, then for all $k \in \mathbb{F}_q$ we have $h_k(A) + h_k(B) = A_1 + B_1 = \omega$. In contrast, letting message m form a polynomial of degree up to t can get around this problem completely:

$$h_k(m) = m(k) = m_1 k + m_2 k^2 + \cdots + m_t k^t$$

Similar calculations show that this is an $(\epsilon = t/q)$-AXU, which meets the AXU-variant of Kabatianskii's ASU-bound in Theorem 13 with equality: $\log(\epsilon^{-1}\lfloor K/b \rfloor) = \log q = r$. This, therefore, justifies the difference between our new AU-bound and the AXU-variant of Kabatianskii's ASU-bound, i.e. when K is an integer multiple of b, AXU-bound is greater than AU-bound with respect to the same set of parameters (ϵ, K, b).

Using Theorem 12 and the above construction, we can build an $(\epsilon = \frac{t}{2^b})$-$ASU$ $(r = 2b, K = tb, b)$,

[14] Since Kabatianskii's ASU-bound has only been proved to be valid in a partial range of parameters (see footnote 8 or [52]), the optimality of polynomial hashing as an ASU remains to be proved. On the other hand, polynomial hashing as an ASU is known to be asymptotically optimal due to Johansson et al. [51], i.e. the authors used polynomials to construct an $(\epsilon = \frac{t}{2^b})$-$ASU(r = 2b, K = tb, b)$, where t is an integer, and they proved that for t fixed and $b \to \infty$ then $2^K = 2^{tb}$ is *asymptotically* the maximum number of messages that can be securely authenticated.

which was originally introduced by Johansson et al. [51]. For any pair of keys $(k, s) \in \mathbb{F}_q^2$:

$$h_{k,s}(m) = s + m(k) = s + m_1 k + m_2 k^2 + \cdots + m_t k^t$$

This meets Kabatianskii's ASU-bound ($r \geq b + \log\left(\epsilon^{-1}\lfloor K/b \rfloor\right)$) with equality.[15] However, Kabatianskii's ASU-bound and its AXU-variant have only been proved to be valid when $t < \sqrt{2^{b+1}(1-2^{-b})} - 1/2 = \sqrt{2q(1-1/q)} - 1/2$, and so the two polynomial hashings as AXU and ASU can only be proved to be optimal under the condition as was also pointed out by Kabatianskii et al. [52].

7.5 Conclusions

In this chapter we have demonstrated that the use of the connection between universal hash functions and error-correcting codes does not always give tight bounds for universal hash functions. This work opens the way for re-examination of existing bounds for universal hash functions which have been derived from theoretical bounds for error-correcting codes (ECC-bounds) [39, 52, 111, 112] or other combinatorial objects such as difference matrices [111], orthogonal or perpendicular arrays [56, 111, 112, 113], and balanced incomplete block designs [56, 93, 110, 111, 113].

Intuitively, universal hash function bounds derived from ECC-bounds might only fit some universal hash function parameters tightly. For example, there exist subclasses of some universal hash functions which cannot be transformed into equivalent codes that achieve equality in the ECC-bounds from which the universal hash function bounds are derived. Within these subclasses, equality in the universal hash function bounds are not achievable. What we have discovered is that the pigeon-hole principle can produce better bounds for the latter, as shown in the new AU-bound introduced in Section 7.2.

In addition, we have quantified the (asymptotic) Wegman-Carter effect using an important value of the hash collision probability ϵ that represents a threshold in behaviours of bounds for AU, AXU, and ASU; the behaviour is summarised in Table 7.2.

[15] There is another ASU-bound due to Gemmell and Noar (see Table 7.2 or [39]), however polynomial hashing as an ASU does not satisfy the bound with equality when $t \in [b, 2^{b-1}]$. This implies that Kabatianskii's ASU-bound is tighter than Gemmell-Noar's ASU-bound over the range of parameters where Kabatianskii's ASU-bound is valid.

	$\epsilon < \left(1 + \frac{b}{K-b}\right) 2^{-b}$	$\epsilon > \left(1 + \frac{b}{K-b}\right) 2^{-b}$
ϵ-AU	Stinson's bound [110, 111] $\log \left(\frac{2^K(2^b-1)}{2^K(\epsilon 2^b-1)+2^{2b}(1-\epsilon)} \right)$	K is an integer multiple of b *New*, Theorems 8 and 11 $\log \frac{K-b}{\epsilon b}$ K is *not* an integer multiple of b *New*, Theorem 8 $\log \left(\epsilon^{-1} \lfloor K/b \rfloor\right)$
ϵ-AXU	Stinson's bound [111] $\log \left(\frac{2^K(2^b-1)}{2^b\epsilon(2^K-1)+2^b-2^K} \right)$	AXU-variant of Kabatianskii's ASU-bound *New*, Theorem 13 $\log \left(\epsilon^{-1} \lfloor K/b \rfloor\right)$ (provided $K/b < \sqrt{2^{r+1}(1-2^{-b})} - 1/2$)
ϵ-ASU	Stinson's bound [110, 111] $\log \left(1 + \frac{2^K(2^b-1)^2}{2^b\epsilon(2^K-1)+2^b-2^K} \right)$	Kabatianskii's bound [52] $b + \log \left(\epsilon^{-1} \lfloor K/b \rfloor\right)$ (provided $K/b < \sqrt{2^{r-b+1}(1-2^{-b})} - 1/2$) Gemmell and Noar's bound [39] $\log K + 2 \log \epsilon^{-1} - \log \log \epsilon^{-1}$

Table 7.2: Classification of lower bounds on the key length r for AU, AXU and ASU in relation to the threshold value of ϵ: $\left(1 + \frac{b}{K-b}\right) 2^{-b}$.

Chapter 8

Conclusions and future research

8.1 Conclusions

In this thesis, we have surveyed the literature on a new and — we believe — important style of protocol, examining mutual and group authentication protocols as well as non-interactive and one-way schemes. To the best of our knowledge, this is the first study that looks at such a wide range of protocols in the new authentication technology and then assesses how they are linked together by several common design principles. Note, there has been another survey written by Suomalainen et al. [114] where the authors only concentrate on pairwise schemes bootstrapping security from either human interactions or secret shared passwords. In addition, there have been works done by various authors, such as Mashatan and Stinson [68, 69], and Pasini and Vaudenay [87, 119], which only look at one-way authentication schemes (both interactive and non-interactive).

In this study, we have discovered that although groups of these protocols have been invented independently by many research authors at various times, the basic design principle of *commitment without knowledge* underlies all of those that either attain or nearly attain the optimal empirical performance. This can be done by distinguishing carefully between when nodes are committed to a value and when they know it.

Not only are these protocols influenced by *commitment without knowledge*, but also the idea of *separation of security concerns* under which protocols should be designed to resist a random attack and a powerful attack separately. These concerns have an impact on the required length of the SASs manually handled by humans, as illustrated in the two following groups of protocols:

- Long authentication string: Balfanz et al., Pasini-Vaudenay and Mashatan-Stinson I/II in Chap-

ter 5;

- Short authentication string: (V-)MANA I and their improved versions of Chapter 5, pairwise and group protocols, such as (S/H)HCBK, of chapters 3 and 4.

Different from any other families of security or cryptography protocols, human interaction plays an important role in the security of these authentication schemes. For this reason, we have formally analysed whether human effort, measured in bits, is optimal relative to the obtained level of security in every protocol presented. It is worth pointing out that the amount of human effort required will crucially determine how usable each protocol is. For example, it might not be desirable to ask humans to compare 80-bit (or 20-hexadecimal-digit) numbers manually since this is not an easy task when these numbers only differ in a few positions. To make these schemes usable in practice the SASs compared by humans must be much shorter, e.g. 4 or 5 digits (or 16–20 bits) numbers. Thus the schemes with short SASs provide a convenient way to bootstrap security that can be used in a variety of ways, in contexts both (1) where all devices are co-located such as medical equipments in hospitals or electronic devices at home as well as CHIP&PIN technology in supermarkets; and (2) where they are not co-located as in many applications of e-healthcare (including telemedicine or virtual healthcare systems) and financial transactions over the phone or the Internet.

Whilst both *direct* and *indirect information binding* strategies are instrumental to attaining (joint) *commitment without knowledge*, their difference becomes apparent when we look at computational cost. In direct binding, large information ($INFOS$) is processed by a new cryptographic primitive, called *keyed digest functions*, whose output is short which is potentially efficient to compute. This is the exact opposite of indirect binding, where the same amount of data must be inputted into long output cryptographic primitives, i.e. cryptographic hash functions or commitment schemes, that are likely to be more expensive. This observation therefore will affect the choice of type of protocols when we want to bootstrap security between low-power devices or when time-consumption is an important design factor, provided that the potential speed up in efficiency of digest computation is realised in practice.

Subsequently, we investigated and showed how data can be digested much faster than it can be hashed, and begun to develop a theory of digest functions. For example, several existing constructions for universal hash functions, using (Toeplitz) matrix multiplication and random numbers, have been considered and then proved that the same algorithms can generate digest functions with properties we require. In addition, a new family of digital signature schemes and Flexi-MACs have been introduced to take advantage of efficient digest functions to speed up computation cost by a significant factor

relative to standard digital signatures and MAC algorithms.

The security of direct binding protocols naturally comes from the uniform distribution of digest functions relative to input message and key. In order to meet the condition, we have introduced a new bound for an almost universal hash function (AU, i.e. a special case of a digest function) using the pigeon-hole principle. We then showed that the new bound is tighter than another similar bound for an AU which is derived from the combination of the Singleton bound in coding theory and an equivalence between error-correcting codes (ECC) and AUs. To the best of our knowledge, this is the first time one demonstrates that the use of the relation between ECC and AU does not always give a tight bound for an AU.

8.2 Short-term public key cryptography

We anticipate that in many, probably a majority, of the practical uses of the classes of protocol described in this thesis, one of the main objectives is the bootstrapping of a means of secret and authenticated communication between electronic devices such as laptop, mobile phones and medical sensors. In almost all such cases, we expect that this will be done by establishing a symmetric session key from public keys or Diffie-Hellman tokens included in the $INFO_A$s to be used in conjunction with some encryption algorithm in a way that gives both secrecy and authentication.

In our environment where we desire low power consumption and perhaps simple processors 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 communications 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.

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, i.e. the shorter length here applies to both traditional public key cryptography (large modulus calculations) and Elliptic curve group implementations of discrete log cryptosystem (e.g. Diffie-Hellman, ElGamal and Schnorr).

8.3 Future research

8.3.1 Security proofs of the new family of authentication protocols

It has been observed throughout this thesis that many protocols introduced here or by other authors lack (complete) proofs of security either in the Random Oracle Model or the Bellare-Rogaway (standard) Model. This is particularly apparent with many group authentication schemes of Chapter 4 due to the difficulty in modelling an unlimited number of protocol participants. Another challenge in providing proof of security arises from the formalisation of a variety of empirical channels as well as looking for the right assumptions which are used to reason about the security of the channels and the protocols. For instance, data transmitted over the strong or normal empirical channels cannot be replayed, delayed and blocked by an intruder, and so it is possible to scale the security proof of a one-shot attack up to multiple-shot attack. On the other hand, proofs get more complicated with weak or bounded delay empirical channels where data sent can be either blocked, replayed or delayed for a significant amount of time.

One possible future research topic is to adapt existing security proofs of several pairwise and one-way authentication schemes in either the Random Oracle Model or the Bellare-Rogaway Model to give proofs for other schemes. The first step would be to understand the existing proofs as well as their limitations since many of them use strong assumptions or only consider certain types of attack such as impersonation, substitution or one-shot attacks.

8.3.2 Digest functions

Since the digest output is short, it is potentially possible to construct digest algorithms that are significantly faster than customised ones, but have demonstrable security.

We have presented some schemes for computing digests in Chapter 6 using a pseudorandom number generator (PRNG) and Toeplitz matrix multiplication. However, one needs to investigate further ones, and to provide convincing proof that the digest functions' requirements are in fact met. In particular, we need to optimise the use of PRNG in the definitions, together with (i) statistical and combinatorial analysis of the generation and quality control issues of such functions, and (ii) comparative performance figures with hashing and other forms of data authentication such as universal hashes.

It is interesting to note that one could perform the statistical and combinatorial tests (e.g. the *strict avalanche condition* and *chi-squared* test for uniformity) exhaustively, or at least on major

components of the algorithms, thanks to short digest output. If successful, we believe that the methods of computing digests will generate much research in the area of PRNG, and in particular almost k-wise independent random variables.

8.3.3 Applications of digest functions

The invention of short output digest functions would potentially influence the use of hashing in a vast range of existing and new applications, thanks to their efficient speed relative to cryptographic hash functions.

This has been partially demonstrated when we introduce new families of digital signature schemes, such as signatures without hashing and Flexi-MAC in Chapter 5, that have several advantages over the conventional MACs or digital signatures. We note, however, that the newly introduced signature schemes completely ignore *padding* operations prior to applying the signing operator as in many signing schemes in practice. Since the padding operations involve cryptographic hashing, we intend to replace all of the hashing computations with digest to exploit the efficiency of a digest function further.

We hope that this research topic will inspire other researchers to find other uses of digest functions, particularly in making authentication more feasible in new applications in the pervasive computing world. Examples of applications are digital rights management or telecommunication where large volumes of data need to be authenticated in real time.

8.3.4 Unifying bounds of universal families of hash functions (UHF)

For each class of UHF, there are several lower bounds on the key length, working in different ranges of the security parameter ϵ. It has appeared difficult to combine them due of some behaviour which is hard to capture within a single formula, notably the Wegman-Carter effect [120]. Equally, these classes of UHF are strongly related to one another, i.e. ASU is a stronger version of AXU, which is a generalisation of AU. One therefore might attempt to explore the possibility of unifying the various bounds of these classes.

One should expect that the inclusion of further parameters could capture a wider range of security properties. The credibility of this approach was illustrated when we generalised the pair-wise version of the combinatorial AU-bound into its k-wise variant in Chapter 7.

Some striking results of Stinson [110, 111] and Bierbrauer et al. [18] demonstrate theoretical equivalences between UHF, error-correcting codes and combinatorial designs. Hence, finding other ways to transfer between combinatorial objects, e.g. Latin squares, and combinations of different

classes of UHF would possibly give rise to better bounds.

8.3.5 Relaxation in human work

Arising from the wide range of applications of the new families of authentication protocols, the problem of how humans can efficiently and reliably play their parts would become important. As a result, there are potentially two different approaches to those of many groups working on this problem to date, aiming to reduce (1) the likelihood of human error, and (2) human work, as laid out below.

Researchers to date have taken no account of human tolerance. Since humans are error-prone, one might want to investigate popular types of human tolerance, such as mismatched digits or characters when comparing numbers, short sentences or digest values. These types of tolerance can be formalised in terms of the *minimum Hamming distance* to quantify the level of security obtained in relative to different degrees of human tolerance. This also suggests a scope for the use of error-correcting codes in displaying information.

Research in human interaction so far has mainly focused on a pair-wise situation. As a consequence, the possibility of human work distribution, as far as we are aware, has never been investigated to date. For example, a number of participants, perhaps in different locations, distribute the work of comparing different portions of the digest, and then report the results to one another at the end. The length of digest therefore can be made longer to increase the security. But on the other hand, we need to minimise the amount of overhead communication required between nodes. This, perhaps surprisingly, resembles the famous *Gossiping and Broadcasting problem* in combinatorics [116], i.e. the number of pair-wise conversations grows linearly with the number of people.

Bibliography

[1] http://www.intel.com/pressroom/archive/releases/20091202comp_sm.htm

[2] http://software.intel.com/en-us/articles/carry-less-multiplication-and-its-usage-for-computing-the-gcm-mode/

[3] http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[4] http://www.sandia.gov/scada/documents/SANDstorm_Submission_2008_10_30.pdf

[5] See: http://www.tplusmedical.co.uk

[6] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*. ISBN 0-486-61272-4.

[7] *Simple Pairing White Paper*. See: www.bluetooth.com/NR/rdonlyres/0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/SimplePairing_WP_V10r00.pdf

[8] E. A. Akkoyunlu, K. Ekanadham and R. V. Huber. *Some constraints and tradeoffs in the design of network communications.* ACM Symposium on Operating Systems Principles, pp. 67-74, 1975. Or see: http://en.wikipedia.org/wiki/Two_Generals%27_Problem

[9] N. Alon, O. Goldreich, J. Hastad and R. Peralta. *Simple Constructions of Almost k-wise Independent Random Variables.* In Proceedings of the IEEE Symposium on Foundations of Computer Science, 1990, vol. 2, pp. 544-553.

[10] 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, 2002.

[11] M. Bellare and P. Rogaway. *Entity Authentication and Key Distribution.* Advances in Cryptology - Crypto 1993, LNCS vol. 773, pp. 232-249.

[12] M. Bellare and P. Rogaway. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.* ACM Conference on Computer and Communications Security, pp. 62-73, 1993.

[13] M. Bellare, R. Canetti and H. Krawczyk. *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols.* In 30th ACM Symposium on Theory of Computing, pp. 419-428, 1998.

[14] M. Bellare, D. Pointcheval and P. Rogaway. *Authenticated Key Exchange Secure against Dictionary Attacks.* Advances in Cryptology - Eurocrypt 2000, LNCS vol. 1807, pp. 139-155.

[15] D.J. Bernstein. *Stronger security bounds for Wegman-Carter-Shoup authenticators.* Advances in Cryptology - Eurocrypt 2005, LNCS vol. 3497, pp. 164-180.

[16] D.J. Bernstein. *The Poly1305-AES message-authentication code.* Fast software encryption, FSE 2005, LNCS vol. 3557, pp. 32-49.

[17] D.J. Bernstein, H.-C. Chen, M.-S. Chen, C.-M. Cheng, C.-H. Hsiao, T. Lange, Z.-C. Lin, and B.-Y. Yang. *A Billion-mulmods-per-second PC.* Advances in Cryptology - Eurocrypt 2009, Rump Session.

[18] 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 1993, LNCS vol. 773, pp. 331-342.

[19] J. Bierbrauer. *Introduction to Coding Theory.* (pages 240-241). Published by CRC Press, 2004. ISBN 1584884215, 9781584884217.

[20] M. Blum. *Coin Flipping by Telephone.* Advances in Cryptology - Crypto 1981, pp. 11-15, 1981. Or see: `http://en.wikipedia.org/wiki/Commitment_scheme`

[21] B. den Boer. *A simple and key-economical unconditional authentication scheme.* Journal of Computer Security 2 (1993), pp. 65-71.

[22] 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, pp. 467-478, 2006.

[23] R. Canetti and H. Krawczyk. *Analysis for Key-Exchange Protocols and Their Use for Building Secure Channels.* Advances in Cryptology - Eurocrypt 2001, LNCS vol. 2045, pp. 453-474.

[24] R. Canetti, S. Halevi, J. Katz, Y. Lindell and P. MacKenzie. *Universally Composable Password-Based Key Exchange.* Advances in Cryptology - Eurocrypt 2005, LNCS vol. 3494, pp. 404-421.

[25] J.L. Carter and M.N. Wegman. *Universal Classes of Hash Functions.* Journal of Computer and System Sciences, vol. 18 (1979), pp. 143-154.

[26] A. L. Cauchy. *Cours d'Analyse de l'Ecole Royale Polytechnique, 1ere partie: Analyse Algebrique.* Paris: Editions Jacques Gabay (published 1992), ISBN 2-87647-053-5. Or see: `http://en.wikipedia.org/wiki/Fundamental_theorem_of_algebra`

[27] D. Chaum. *Secret-ballot receipts: True voter-verifiable elections.* Security and Privacy Magazine, IEEE, Jan.-Feb. 2004. Volume 2, Issue: 1, pp. 38-47.

[28] S.J. Creese, M.H. Goldsmith, R. Harrison, A.W. Roscoe, P. Whittaker and I. Zakiuddin. *Exploiting empirical engagement in authentication protocol design.* In Proceedings of the International Conference on Security in Pervasive Computing ($SPC'05$), LNCS vol. 3450, pp. 119-133, 2005.

[29] S.J. Creese, M.H. Goldsmith, A.W. Roscoe and M. Xiao. *Bootstrapping multi-party ad-hoc security.* In Proceedings of IEEE Security Track, pp. 369-375, 2006.

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

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

[32] I. Damgård. *A Design Principle for Hash Functions.* Advances in Cryptology - Crypto 1989, LNCS vol. 435, pp. 416-427.

[33] W. Diffie and M. Hellman. *New Directions in Cryptography.* IEEE Trans. Inform. Theory, vol. 22, no. 6, pp. 644-654, November 1976.

[34] D. Dovel and A.C. Yao. *On the security of public key protocols.* In Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science, pp. 350-357, 1981.

[35] M. Etzel, S. Patel and Z. Ramzan. *SQUARE HASH : Fast message authentication via optimized universal hash functions.* Advances in Cryptology - Crypto 1999, LNCS vol. 1666, pp. 234-251.

[36] C. Gehrmann, C. Mitchell and K. Nyberg. *Manual Authentication for Wireless Devices.* RSA Cryptobytes, vol. 7, no. 1, pp. 29-37, 2004.

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

[38] International Organisation for Standardisation, Geneve, Switzerland. *ISO/IEC 9798 Information technology - Security techniques - Entity authentication - Part 6: Mechanisms using manual data transfer*, 2003.

[39] P. Gemmell and M. Naor. *Codes for Interactive Authentication.* Advances in Cryptology - Crypto 1993, LNCS vol. 773, pp. 355-367.

[40] S.W. Golomb. *Shift Register Sequences.* ISBN: 0894120484, Aegean Park Press, 1981.

[41] M.T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik and E. Uzun. *Loud and Clear: Human-Verifiable Authentication Based on Audio.* IEEE International Conference on Distributed Computing Systems, (ICDCS'06), pp. 10-33.

[42] R. M. Gray. Toeplitz and Circulant Matrices: A Review. See: `http://ee.stanford.edu/ gray/toeplitz.pdf`

[43] R. P. Grimaldi. *Discrete and Combinatorial Mathematics: An Applied Introduction.* 4th edn. ISBN 0-201-19912-2, pp. 244-248, 1998. Or see: `http://en.wikipedia.org/wiki/Pigeonhole_principle`

[44] C. G. Gnther. *Alternating step generators controlled by de Bruijn sequences.* Advances in Cryptology - Eurocrypt 1987, pp. 5-14, LNCS vol. 304. Or see: *http://en.wikipedia.org/wiki/Alternating_step_generator*

[45] H. Handschuh and B. Preneel. *Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms.* Advances in Cryptology - Crypto 2008, LNCS vol. 5157, pp. 144-161.

[46] J. Hastad, R. Impagliazzo, L.A. Levin and M. Luby. *A Pseudorandom Generator from any One-way Function.* SIAM Journal on Computing, vol. 28, pp. 12-24, 1999.

[47] S.-H. Heng and K. Kurosawa. *Square hash with a small key size.* Eighth Australasian Conference on Information Security and Privacy, ACISP 2003, LNCS vol. 2727, pp. 522-531.

[48] J.-H. Hoepman. *Ephemeral Pairing on Anonymous Networks.* In D. Hutter and M. Ullmann, editors, International Conference on Security in Pervasive Computing, SPC 2005, LNCS vol. 3450, pp. 101-116.

[49] J.-H. Hoepman. *Ephemeral Pairing Problem.* International Conference Financial Cryptography, LNCS vol. 3110, pp. 212-226, 2004.

[50] M. Jakobsson and S. Wetzel. *Security Weaknesses in Bluetooth.* Topics in Cryptology CT-RSA 2001, LNCS vol. 2020, pp. 176-191.

[51] T. Johansson, G.A. Kabatianskii and B. Smeets. *On the relation between A-Codes and Codes correcting independent errors.* Advances in Cryptology - Eurocrypt 1993, LNCS vol. 765, pp. 1-11.

[52] 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.

[53] D. Knuth. *The Art of Computer Programming.* Vol 2, Seminumerical Algorithms, Third Edition (Reading, Massachusetts: Addison-Wesley, 1997), ISBN 0-201-89684-2.

[54] H. Krawczyk. *LFSR-based Hashing and Authentication.* Advances in Cryptology - Crypto 1994, LNCS vol. 839, pp. 129-139.

[55] H. Krawczyk. *New Hash Functions For Message Authentication.* Advances in Cryptology - Eurocrypt 1995, LNCS vol. 921, pp. 301-310.

[56] K. Kurosawa, K. Okada, H. Saido and D.R. Stinson. *New combinatorial bounds for authentication codes and key predistribution schemes.* Designs, Codes and Cryptography, 15 (1998), pp. 87-100.

[57] K. Kurosawa and S. Obana. *Combinatorial Bounds on Authentication Codes with Arbitration.* Designs, Codes and Cryptography, 22 (3), pp. 265-281 (2001).

[58] S. Laur and K. Nyberg. *Efficient Mutual Data Authentication Using Manually Authenticated Strings.* LNCS vol. 4301 on LNSC, pp. 90-107, 2006.

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

[60] S. Laur and S. Pasini. *SAS-Based Group Authentication and Key Agreement Protocols.* In Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, pp. 197-213.

[61] A.Y. Lindell. *Comparison-Based Key Exchange and the Security of the Numeric Comparison Mode in Bluetooth v2.1.* Topics in Cryptology CT-RSA, LNCS vol. 5473, pp. 66-83, 2009.

[62] J.H. van Lint. *Introduction to Coding Theory.* 2nd edition, GTM 86, Springer-Verlag, 61. ISBN 3-540-54894-7. See: `http://en.wikipedia.org/wiki/Singleton_bound`

[63] 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, pp. 147-166, 1996.

[64] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton Computer Science Notes. Princeton University Press. January, 1996.

[65] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-correcting Codes.* Norh-Holland, 1977.

[66] A. Madhavapeddy, D. Scott, R. Sharp and E. Upton. *Using Camera Phones to Enhance Human-Computer Interaction.* Proceedings of Ubiquitous Computing (UbiComp 2004), pp. 1-2, 2004.

[67] 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, 1990.

[68] A. Mashatan and D.R. Stinson. *Interactive two-channel message authentication based on Interactive-Collision Resistant hash functions.* International Journal of Information Security, 2009, vol. 8 (1), pp. 49-60.

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

[70] M. Matsumoto. *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.* ACM Transactions on Modeling and Computer Simulation 8: 3-1 (1998).

[71] R. Mayrhofer and M. Welch. *A Human-Verifiable Authentication Protocol Using Visible Laser Light.* International Conference on Availability, Reliability and Security. ARES 2007, pp. 1143-1148.

[72] 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, pp. 110-124, 8-11 May 2005.

[73] K. Mehlhorn and U. Vishkin. *Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories.* Acta Informatica archive, vol. 21 , Issue 4 (November 1984), pp. 339-374.

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

[75] R.C. Merkle. *A digital signature based on a conventional encryption function.* Advances in Cryptology - Crypto 1987, LNCS vol. 293, pp. 369 - 378.

[76] T. Moran and M. Naor. *Polling with Physical Envelopes: A Rigorous Analysis of a Human-Centric Protocol.* Advances in Cryptology - Eurocrypt 2006, LNCS vol. 4004, pp. 88-108.

[77] W. Nevelsteen and B. Preneel. *Software performance of universal hash functions.* Advances in Cryptology - Eurocrypt 1999, LNCS vol. 1592, pp. 24-41.

[78] 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, 2006.

[79] L.H. Nguyen and A.W. Roscoe. *Authenticating ad hoc networks by comparison of short digests.* Information and Computation, vol. 206 (2008), pp. 250-271. Special Issue of Information and Computation on Computer Security: Foundations and Automated Reasoning 2006.

[80] L.H. Nguyen and A.W. Roscoe. *Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey.* Journal of Computer Security (to appear). See: `http://www.comlab.ox.ac.uk/files/2104/Compara.pdf`

[81] L.H. Nguyen and A.W. Roscoe. *New combinatorial bounds for universal families of hash functions.* Manuscript is available. A short version of the paper is presented at the Summer School on Provable Security, 2009. See: `http://www.comlab.ox.ac.uk/files/749/uhf.pdf`

[82] L.H. Nguyen and A.W. Roscoe. *Separating two roles of hashing in one-way message authentication.* Proceedings of FCS-ARSPA-WITS 2008, pp. 195-210.

[83] L.H. Nguyen and A.W. Roscoe. *Generating short-output digest functions*, in preparation.

[84] L.H. Nguyen. *Final committee draft of ISO/IEC 9798-6 — Entity authentication using manual data transfers.* This was submitted to the committee of ISO/IEC JTC1/SC 27 (working group 2).

[85] J. Noar and M. Noar. *Small-Bias Probability Spaces: Efficient Constructions and Applications.* ACM Symposium on Theory of Computing, pp. 213-223, 1990.

[86] S. Pasini and S. Vaudenay. *SAS-based Authenticated Key Agreement.* Public Key Cryptography - PKC 2006: The 9th international workshop on theory and practice in public key cryptography, LNCS vol. 3958, pp. 395-409.

[87] S. Pasini and S. Vaudenay. *An Optimal Non-interactive Message Authentication Protocol.* Topics in Cryptology - CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, LNCS vol. 3860, pp. 280-294.

[88] C. Papadimitriou. *Computational Complexity.* Chapter 11. Addison-Wesley, 1994. ISBN 0-201-53082-1.

[89] R. Pass. *On Deniability in the Common Reference String and Random Oracle Model.* Advances in Cryptology - Crypto 2003, LNCS vol. 2729, pp. 316-337.

[90] T. Peyrin and S. Vaudenay. *The Pairing Problem with User Interaction.* 20th International Information Security Conference 2005, pp. 251-266.

[91] M.O. Rabin. *Fingerprinting by Random Polynomials.* Center for Research in Computing Technology, Harvard University. Tech Report TR-CSE-03-01, 1981. Retrieved on 2007-03-22.

[92] I. S. Reed and G. Solomon. *Polynomial Codes Over Certain Finite Fields.* Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2 (1960), pp. 300-304. Or see: `http://en.wikipedia.org/wiki/Reed-Solomon_error_correction`

[93] R.S. Rees and D.R. Stinson. *Combinatorial characterizations of authentication codes II.* Designs, Codes and Cryptography, vol. 7 (1996), pp. 239-259.

[94] J.A. Rice *Mathematical Statistics and data Analysis.* ISBN 0-534-20934-3

[95] R.L. Rivest. *The MD6 Hash Function.* Advances in Cryptology - Crypto 2008, .

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

[97] 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=WO2007052045&DISPLAY=STATUS

[98] 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.

[99] 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.

[100] S. Ross. *A First Course in Probability.* ISBN 0-13-033851-6.

[101] D.V. Sarwate. *A note on universal classes of hash functions.* Information Processing Letters, vol. 10 (1), pp. 41-45, 1980.

[102] N. Saxena, J.-E. Ekberg, K. Kostiainen and N. Asokan. *Secure Device Pairing based on a Visual Channel.* In the Proceedings of the IEEE Symposium on Security and Privacy 2006, pp. 306-313.

[103] B. Schneier. *Applied Cryptography.* John Wiley and Sons, 1996. ISBN 0-471-11709-9. Or see: http://en.wikipedia.org/wiki/Cryptographic_hash_function

[104] V. Shoup. *On Fast and Provably Secure Message Authentication Based on Universal Hashing.* Advances in Cryptology - Crypto 1996, LNCS vol. 1109, pp. 313-328.

[105] N. Smart. *Cryptography, An Introduction.* ISBN 0 077 09987 7 (PB). Or see: http://en.wikipedia.org/wiki/Chosen_plaintext_attack

[106] F. Stajano and R. Anderson. *The resurrecting duckling: Security issues for ad-hoc wireless networks.* Workshop on Security Protocols 1999, LNCS vol. 1976, pp. 172-194.

[107] 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, LNCS, 1999.

[108] F. Stajano. *Security for Ubiquitous Computing.* Wiley, 2002.

[109] F. Stajano and R. Anderson. *The Resurrecting Duckling – What Next?* In the Proceedings of the 8th International Workshop on Security protocols, LNCS vol. 2133, pp. 204-214, 2000.

[110] D.R. Stinson. *Universal Hashing and Authentication Codes.* Advances in Cryptology - Crypto 1991, LNCS vol. 576, pp. 74-85, 1992.

[111] D.R. Stinson. *On the Connections Between Universal Hashing, Combinatorial Designs and Error-Correcting Codes.* Congressus Numerantium, vol. 114, pp. 7-27, 1996.

[112] D.R. Stinson. *The combinatorics of authentication and secrecy codes.* Journal of Cryptology, vol. 2 (1990), pp. 23-49.

[113] D.R. Stinson. *Combinatorial techniques for universal hashing.* Journal of Computer and System Sciences, vol. 48 (1994), pp. 337-346.

[114] J. Suomalainen, J. Valkonen and N. Asokan. *Security Associations in Personal Networks: A Comparative Analysis.* In the Proceedings of the 4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks 2007. LNCS vol. 4572, pp. 43-57.

[115] R. Taylor. *An Integrity Check Value Algorithm for Stream Ciphers.* Advances in Cryptology - Crypto 1993. LNCS vol. 773, pp. 40-48, 1994.

[116] R. Tijdeman. *On a Telephone Problem.* Nieuw Archief voor Wiskunde, vol. 19, pp. 188-192, 1971.

[117] 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 2007. LNCS vol. 4886, pp. 307-324, 2008.

[118] 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 2006. LNCS vol. 4357, pp. 150-164.

[119] S. Vaudenay. *Secure Communications over Insecure Channels Based on Short Authenticated Strings.* Advances in Cryptology - Crypto 2005, LNCS vol. 3621, pp. 309-326.

[120] M.N. Wegman and J.L. Carter. *New Hash Functions and Their Use in Authentication and Set Equality.* Journal of Computer and System Sciences, vol. 22, pp. 265-279, 1981.

[121] A.F. Webster and S.E. Tavares. *On the Design of S-Boxes.* Advances in Cryptology 1985, LNCS vol. 218, pp. 523-534, 1986.

[122] Ford-Long Wong and F. Stajano. *Multi-channel Protocols.* Proceedings of the 13th International Workshop on Security Protocols 2005. LNCS vol. 4631, pp. 128-132.

[123] Ford-Long Wong and F. Stajano. *Multi-channel Security Protocols.* IEEE Pervasive Computing, vol. 6 (4), pp. 31-39, Oct-Dec 2007.