A System for the Verification of Location Claims

Michelle Graham

BSc.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

School of Computing

Supervisor: Dr. David Gray

October 2010

Declaration

I, Michelle Graham, hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____

(Candidate) ID No.: 51724240

Date: 17th October 2010

Contents

Abstract

As location becomes an increasingly important piece of context information regarding a device, so too must the method of providing this information increase in reliability. In many situations, false location information may impact the security or objectives of the system to which it has been supplied. Research concerning localization and location verification addresses this issue. The majority of solutions, however, revolve around a trusted infrastructure to provide a certified location.

This thesis presents an enhanced design for a location verification system, moving verification away from infrastructure-based approaches. Instead, an ad hoc approach is presented, employing regular local devices in the role usually reserved for trusted entities - the role of the evidence provider.

We begin with an introduction to the area of localization, outlining the primary techniques employed. We summarize previous approaches, highlighting the improvements and outstanding issues of each. Following this, we outline a novel metric for use with distance bounding to increase the accuracy of evidence extracted from the distance bounding process. We show through emulation that this metric is feasible within an IEEE 802.11 wireless network.

We detail the Secure Location Verification Proof Gathering Protocol (SLVPGP), a protocol designed to protect the process of evidence gathering. We employ our novel metric to confirm the presence of a device in an area. We repeatedly extend the SLVPGP's basic design to form three protocols, each with increasingly stronger security. These protocols are formally verified to confirm their specified security properties.

To complete the design of our verification system, we present two approaches to judging a claim based on the evidence supplied. We demonstrate the accuracy of these approach through simulation. We also include a brief outline of the concept of reputation and discuss an existing approach to its calculation based on the previous behaviour of devices within the system.

Acknowledgements

I wish to acknowledge everyone who has helped me with my research and this thesis. I would like to thank the Irish Research Council for Science Engineering and Technology and the Embark Initiative for supporting me through their Ph.D. Research Grant scheme. I also thank my examiners, Dr. Srdjan Čapkun and Dr. David Sinclair for their useful comments, suggestions and feedback on this thesis, and Dr. Gavin Lowe for all his help in conquering the tricks necessary to tame Casper for my own uses.

I would like to express my deepest gratitude to my supervisor, Dr. David Gray, without whom I would never have come this far. Over the years, David has been an inspiration and a sounding board, as well as a source of advice and a fresh perspective on the issues. Throughout my work, from its infancy as an idea about IP address localization right through to its current form, no matter what the area, he has been a huge influence. His support and encouragement have helped me to overcome many obstacles that seemed impossible along the way and this work would not exist without all of his guidance.

There is one other member of my DCU life that has shaped the path of my academic career in an enormous way. While working on my undergraduate degree within the college, I met John Judge and since then he has become a mentor of sorts. From assisting me with my research grant application to acting as a sounding board when I despaired of finding solutions, from working on my transfer report right through to aiding in Viva preparation and dispelling pre-Viva nerves, John has played an unmistakeable role and I will be eternally grateful for all he has done.

Beginning in my very first year of college and continuing right through to the years of research on this work, Redbrick has been an incredibly important part of my life. It has broadened both my social and academic circles within the college and beyond to other academic institutions around the country. I would like to thank the original founders of the society for starting what would prove to be such a huge part of so many lives over the years. I would also like to thank all the people available in #lobby, via hey and on the boards for providing technological advice, interesting conversation to distract when research hit a roadblock and even occasionally providing the perfect new angle I hadn't

thought of yet.

To my friends, who have been so very patient with me every time I was stuck at a computer screen rather than being able to go out and play, who were there with tea and sympathy when I needed it, and who regaled me with entertaining (and distracting!) stories when my mind was overloaded, you have no idea how much I have appreciated this. Una, your teasing served to remind me that there would be life after college, while your "gentle" comments about my capability made me remember why we're friends. To the Scoobies, who accepted that I disappeared from life for many months at a time with only minimal teasing and who have been a great source of sanity in an insane time of my life, thank you for not kicking me out of the group.

Words do not exist to convey the amount of gratitude and appreciation I feel for everything that Eoghan has done for me over these years. His unquestioning belief in my ability to succeed has been a light during some very dark times and his help has been invaluable in so many ways, from chapter upon chapter of proof reading to help with automating simulation runs through scripting. For all the nights in with pizza and tv because going out was just not possible, for your willingness to make the most absurd wishes come true (including my pony) just to keep me smiling, for your support and for your mere presence, thank you.

I was lucky enough to have been born with parents, David and Jacqueline, who believed their children should follow their dreams. They taught me to always try my best, and even when I thought this was far beyond what I was capable of, their belief that this was within my reach has never wavered. They have suffered through every rough patch and celebrated every milestone along the way, no matter how small. My siblings, Erica and Declan, have been with me every step of the way, keeping me afloat with their belief in my ability when my own had run out. Their patience has seemed unending, and though I have tried to repay them for all their support with baked goods, I know that what they have given me is priceless and no amount of brownies could ever compare.

In addition to an extraordinarily supportive family, I have been blessed with an extremely close extended family. Sinèad, Eoin and Gina, you have been there since the beginning, living through all the stress and worry and savoring all the successes. You have listened to my thoughts and even sat through presentations. You probably know my work

best out of everyone beyond Eoghan. Thank you for everything and I promise not to make you read beyond the acknowledgements. Auntie Jean and Uncle Colm, better support parents I couldn't ask for, and I know you will appreciate my freedom in France from now on. Thank you for your emergency air conditioning, as many of the thoughts in this work would probably be a puddle on the ground in Vias if you hadn't stepped in.

My grandparents, both maternal and paternal, have always been involved in my life, and nothing made them prouder than to hear about their granddaughter's progress through the school system and on into third level academia. Though I didn't become the nun as my grandad wanted, I know that the pride he felt on the day I graduated from my B.Sc. is nothing to what he feels now. Oftentimes, the thought of being able to show him my final parchment motivated me to keep going when I was extremely low, and I credit him as one of the main reasons I didn't leave my research by the side of the road.

"Nothing worth having ever comes easily" is a statement I've heard a lot over the last few years. Though at times it's seemed like this just wasn't worth the trouble, the people in my life have kept things in perspective and prevented me from throwing it all away. Saying "thank you" just isn't enough, but I hope that they understand everything it conveys.

List of Figures

List of Tables

List of Algorithms

Index of Abstract Protocol Notation

Over the course of this work, multiple protocols are presented, modelled in abstract protocol notation. For the sake of clarity, we include here an index of this notation employed.

A, B	Identities of participating devices
$A \rightarrow B:$	Prefix indicating the sender (A) and recipient (B) of a message
$\mathcal{N}$	Nonce (randomly generated long number)
$\mathcal{N}_A$	Nonce generated by or destined for A
$\mathcal{H}()$	The hash function
H_A	Hash generated by or destined for A
$H_{A,k}$	The k^{th} hash in a hash chain, generated from H_A
T_A	Timestamp generated by A
K_A	Cryptographic key belonging to A
K_A^+	Public key belonging to A
K_A^-	Private key belonging to A
K_{AB}	Session key shared by A and B
$\{\}_{K_A^+}$	Empty message encrypted with A's public key
$\{\|\|\}_{K_A^-}$	Empty message digitally signed using A's private key
$\{\mathcal{N}\}_{K_A^+}$	Nonce encrypted with A's public key
$\{\|T_A\|\}_{K_A^-}$	Timestamp signed with A's private key

Index of Casper Syntax Employed

Over the course of this work, multiple protocols are presented, translated into Casper. For the sake of clarity, we include here an index of the Casper syntax employed.

NOTATION:

A -> B:	Prefix indicating the sender (A) and recipient (B) of a message
{}	Empty message
{}{PKAgent(A)}	Empty message encrypted using Agent A's public key
m % var	Stores message m in variable var. This notation allows participants to receive encrypted messages without possessing the ability to decrypt them.
var % m	Sends message m (currently stored in variable var) to recipient. This allows participants to forward encrypted messages without possessing the ability to recreate them. The recipient must possess the ability to decrypt the message.
symbolic	Keyword used with functions indicating that Casper produces its own values to represent the function's results

COMPONANTS:

Agent	Process type representing Claimant
Prover	Process type representing Proof Provider
Verifier	Process type representing Verifier
OracleType	Process type representing Oracle
n*	Nonce

h* Hashed value

t* Timestamp

dV Verifier's final verdict

xC Claimant's location

PKVerifier() Custom function to return the public key of a Verifier

SKVerifier() Custom function to return the secret key of a Verifier

PKProver() Custom function to return the public key of a Prover

SKProver() Custom function to return the secret key of a Prover

PKAgent() Custom function to return the public key of an Agent

SKAgent() Custom function to return the secret key of an Agent

Contents of Casper script sections:

#Free_variables Definition of all variables employed within the script

#Processes Information about the participants running in the protocol

#Specification Requirements of the protocol

#Protocol description Description of the protocol's message sequence

#Actual variables Actual values to be employed in the system being checked

#Functions List of functions defined in the #Free variables section

#System List of participants to be included in the system to be checked

#Intruder Information Intruder's identity and set of data values known at initiation

Index of Abbreviations/Acronyms

AES	Advanced Encryption Standard
AoA	Angle of Arrival
C	Claimant
CBS	Covert (Hidden) Base Station
CSP	Communicating Sequential Processes
DDoS	Distributed Denial of Service
DoS	Denial of Service
ECC	Elliptic Curve Cryptography
FDR	Failures-Divergences Refinement
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IR	Infra Red
LBS	Location Based Service
MANET	Moble Ad-hoc Network
MBS	Mobile Base Station
PDF	Probability Density Function
PP	Proof Provider

PPS	Precise Positioning System
RF	Radio Frequency
RFID	Radio Frequency Identification
RSA	Rivest, Shamir and Adleman
RSS	Received Signal Strength
RSU	Roadside Unit
SHA	Secure Hash Algorithm
SLVPGP	Secure Location Verification Proof Gathering Protocol
TCP	Transmission Control Protocol
TDoA	Time Difference of Arrival
ToA	Time of Arrival
UDP	User Datagram Protocol
US	Ultrasound
UWB	Ultra-Wideband
V	Verifier
VANET	Vehicular Ad-hoc Network
VM	Verifiable Multilateration
WoA	Window of Acceptance
XML	Extensible Markup Language

Chapter 1

Introduction

In recent years, there has been a definite shift towards ubiquitous computing. One of the main advances in this area is the increased importance of mobile computing, such as Vehicular Ad-hoc Networks (VANETs) [79] and Mobile Ad-hoc Networks (MANETs) [59]. More and more services have moved to a mobile setting, relying upon mobile networks such as these. For this reason, context has become a key factor, with a user's location becoming a critical piece of contextual information. It allows security systems to grant access based on establishing a user's presence within a specific area and provides a method for businesses to tailor services to their customers. It even allows customers to get more out of existing services in the form of Location Based Services (LBS) [87, 75]. Information regarding a user's location can be gathered in two main ways, either where a user provides their location directly (self-location) or where an external entity can locate any user directly (remote-location).

In the case of self-location, a user supplies the requesting entity with its location. One simple example of self-location with a Location Based Service would be a user locating cinemas, restaurants, etc in their vicinity. The user provides their location to a central look-up service, with the service returning results in the vicinity of the supplied location. This location may be sourced in a number of ways, from a direct input by the user based on his or her own knowledge to being provided by a device such as a Global Positioning System (GPS) unit [46]. As the user is supplying this location information directly, the information provided cannot be guaranteed to be accurate. In fact, even if the GPS unit is contained within a Tamper-Resistant Module, the resulting location is not guaranteed,

as GPS signals can also be interfered with from outside the unit (spoofing). These spoofed signals can result in the computation of an incorrect location, leading to the provision of a false location to the LBS.

In the case of remote-location, an external entity locates the user itself, rather than the user providing its location. In order to accomplish this, the user must be carrying some form of device which the entity can access. It is this device which the entity locates, and not the user themselves. An obvious example of such a situation would be a mobile telephone network. Through the use of triangulation and other techniques, mobile phone companies can calculate the location of a user of their network, once the user remains within the area of the network. This method of locating a user is somewhat more difficult to tamper with, but some simple approaches can be employed. An example of such an approach is the use of a Faraday Cage or Hoffman Box to block or restrict the signal emitting from the user's device. However, this form of tampering requires some form of physical interference with the device, which may not always be viable. Therefore, while remote-location is vulnerable to interference, it is not as susceptible as self-location. Remote-location's primary weakness is its reliance on infrastructure to provide a location. This reliance limits the use of any remote-location system to those areas containing an infrastructure presence and restricts applicability of the system based on the use of specific devices.

As self-location cannot be relied upon to provide trustworthy location information and remote-location requires adherence to a specific device or area in which infrastructure exists, a new approach must be found. An alternative solution to this is to redefine the problem. Rather than attempting to locate a user's device, the user's device makes a claim regarding its location that can then be verified. By altering the approach taken, the problem of providing a trustworthy location becomes much more manageable. The issue of locating the user is eliminated and the focus is placed on the veracity of the claim. Rather than a system attempting to prove the location of the user, the user provides proof of their claim. This prevents the need for special and costly hardware or an overly complex solution.

To date, similar to the case of remote-location, the trend in localization techniques relies upon a pre-existing infrastructure to provide trusted devices with which claiming devices can interact. However, reliance upon a fixed or limited infrastructure in this manner

reduces the applicability of localization technology. In addition to this, the cost of employing the system increases dramatically. With mobile devices and mobile networks becoming a staple of current technology, the need for a specific infrastructure of devices is no longer present. This approach is particularly suited to the vehicular network environment, with devices in this type of network behaving in predictable patterns and possessing sufficient power to perform even complex encryption.

This thesis presents research on the development of a centralised location verification system, designed without the requirement of a pre-existing device infrastructure for use as evidence providers (Proof Providers), to verify the location of a user's claiming device. Instead, the location verification system employs untrusted devices in the vicinity of the claiming device (Claimant) as Proof Providers. The system's central entity extracts a verdict from the Proof Provider collective, based either on a majority verdict or the trust values of those involved. This verdict indicates the likelihood of the claim being legitimate. The location verification system employs a novel distance bounding metric (the binary metric) when computing evidence, to distinguish between direct and proxied communications over a wireless network. This system was initially developed for use within VANETs, but has not been deployed in a physical VANET to date. Therefore it is unknown if the high levels of mobility reached by vehicular mobile devices prevents the system from attaining useful levels of verification accuracy. However, the system is also suitable for use in MANETs composed of handheld mobile devices, as these do not usually reach the same mobility levels.

The development of the location verification system presented here is achieved through the combination of three main research strands. The first strand presents a novel metric for use in distance bounding [13], employing round trip times to detect proxy attacks. The second strand introduces a new security protocol designed to protect the gathering of proof by a device making a location claim. Finally, the third strand presents the design of a central entity and two approaches to the judging of a location claim.

We present a novel metric for use with distance bounding which indicates whether or not a device communicating directly or via a proxy, thus providing a method for the detection of proxy attacks. We discuss the concept of a *window of acceptance* (WoA) for use with this metric, to distinguish between the time required for a direct communication and

that required to complete a proxied exchange. We briefly discuss the design of a proxied exchange, outlining the form of proxy that this approach can detect. We present a pair of protocols for use in establishing the WoA for a specific situation. We perform timed emulations of distance bounding in both wireless and wired settings. These emulations demonstrate that the employment of this metric is reasonable in both settings, i.e. that the time required to perform distance bounding between two devices is significantly different where more than a single network hop is performed.

Following from this, we present the Secure Location Verification Proof Gathering Protocol (SLVPGP), a security protocol designed to protect the process of gathering proof for use in a location claim. This protocol employs the distance bounding binary metric in the proof calculation stage of evidence gathering. We study the security properties required to build a satisfactorily secure protocol for use in a proof gathering scenario. We discuss the reasons for developing a three tiered protocol rather than employing a single all-encompassing approach. We analyse each tier's relative costs incurred in increasing the level of security provided. This analysis demonstrates that while the third tier incurs a greater transmission cost, its cryptographic costs are slightly less than those of the second tier, thus providing a higher level of security with fewer cryptographic operations.

In order to confirm the upholding of each tier's hypothesised security properties, we formally analyse the protocols' security. This analysis is carried out in the form of model checking. We employ a high level modelling language (Casper [58]) to describe the design and generate a mathematical model for each tier. We then verify these models using the FDR [83] model checker. The results of this process support the hypothesis that the desired security properties for each protocol tier hold within the specified environment. As a supporting side work, we formally investigate the security of broadcasting secure messages within a system. This investigation is also performed through model checking. The results of this investigation support the already-held belief that broadcasting a message does not impact the security of that message, unless said message is insufficiently protected through encryption.

Finally, we present two designs for a verification engine, one employing majority verdicts and one employing trustworthiness values of the participating devices. We employ an existing set of techniques such as the beta Probability Density Function [29], used in Josang

and Ismail's beta reputation system [50], to calculate the trustworthiness of a participating device at the time of a location claim. This trustworthiness calculation system provides a method of establishing the trustworthiness of a device's verdict, for use in a trust-based verification approach. Additionally, the inclusion of a reputation system provides a method of Proof Provider selection with which a Verifier can select only those devices most likely to behave honestly for use in a specific claim.

The verification system has been designed to address the reliance of most localization systems upon trusted devices. We have designed two approaches to verification, one based on majority verdicts and the other based on the trust values of those devices providing evidence for that claim. In the summation-based approach, the Verifier employs a majority verdict system to compute the possibility of a location claim. Rather than relying on uniformly trusted infrastructure units, trust is placed in a group as a whole, and the prevailing verdict on a claim's veracity indicates the claim's possibility. In the trust-based approach, the Verifier computes the overall trust value of a claim from the individual device trust values of those providing supporting evidence, rather than treated each device's verdict as equal. By computing the trust value of each device based on previous actions, a distinction can be made between honest and dishonest devices, thus removing the need for uniformly trusted units.

We employ percentage-based thresholds to dictate the levels the claim's value must reach to receive a specific verdict, thus creating a flexible system. Through a simulation of both verification approaches, we demonstrate the effectiveness of each approach. We prove that summation-based verification successfully detects fraudulent claims while awarding positive verdicts to honest claims. This creates a location verification system designed for mobile networks which requires no existing infrastructure or specifically trusted devices for use as evidence providers.

The thesis is structured as follows:

Chapter two introduces localization and location verification before discussing existing approachs to solving this problem. We outline several key localization techniques employed within the field. We discuss a selection of existing schemes, provide context and background information for the research presented in this thesis. We also identify the

primary attacks attempted on localization schemes and techniques.

Chapter three examines the process of distance bounding and describes a novel twist on the traditional method of employing the technique, in the form of a novel metric. We present the concept of a *Window of Acceptance (WoA)* and describe two protocols for use in computing the WoA in practice. We also present emulation data which supports the plausibility of employing distance bounding in this manner and discuss remaining open issues in this matter.

Chapter four presents the primary aspect of our contribution to the area of localization: the Secure Location Verification Proof Gathering Protocol. We discuss the security properties desired for this protocol and outline the design process undertaken to achieve a protocol supporting these properties. We propose three variations on the basic design of the protocol and discuss the relative costs of each variation in comparison to the security gained.

Chapter five examines the security of the Secure Location Verification Proof Gathering Protocol in each of the three variations, using the formal verification technique of model checking. We outline the limits of formal verification and explain how the model checking process functions. We discuss the process of formally verifying the protocols through model checking, including the application of safe simplifications. We then analyse the findings of the performed verifications.

Chapter six presents the design for a central verifying entity (Verifier) responsible for the verification of location claims. We discuss the notion of trust in a device and employ existing techniques to establish the trustworthiness of all devices within the system. We outline the responsibilities of the Verifier during the initialization of a claim, including the method by which volunteers are gathered and Proof Providers selected. We describe two approaches for use in the verification of a device's claim: extracting a majority verdict from the selected group of Proof Providers and employing trust values to calculate a claim's overall trust level. Finally, we present simulations results demonstrating the accuracy of

each approach.

Chapter seven sums up this work and discusses several remaining open questions, outlining possible areas of interest for future work.

The research presented in this work has been published in the proceedings of several peer-reviewed conferences. [35] gives an early overview of the location verification system as a whole. [37] presents the SLVPGP and its extensions in an earlier form. It also briefly discusses the model checking process performed in order to formally verify the security of the protocols. Our work on distance bounding and the binary metric is presented in [36], with discussion and analysis from this presentation leading to the development of the WoA computation protocols. Finally, [38] presents a recent overview of the system as a whole. In this paper, we approach the location verification system as a method of abstracting the process of location verification away from any LBS wishing to employ verified location information in their services.

Chapter 2

Background

2.1 Introduction

This chapter provides the context and relevent background for the research presented in this work. We define the concept of device localization and outline the core techniques employed in localization to estimate the distance between two devices (ranging techniques), along with common attacks against them. After explaining the techniques involved in localization, we discuss its primary research direction, positioning (Section 2.3.1). We then discuss the specific focus of the research presented in this work, location verification (Section 2.3.2). We provide a comparative overview of existing positioning and verification systems within these headings, beginning with Want et al's ActiveBadge system [101]. We then discuss our own work in comparison to these systems, outlining their influence on the final design of our work.

2.2 What is Localization?

Within this work, "localization" is defined as the act of determining a user's location, with location taken to mean a user's position either in the real world or relative to other devices within a particular system. Currently, users can only be located relative to the devices they carry due to the current limitations of technology. Therefore, localization is effectively the act of determining a device's location, under the assumption that a specific user is located in proximity to that device. There are a number of research substrands within the area of localization, including positioning (localization's main research topic)

and location verification. While positioning has received more research attention due to its commercial appeal, research into systems designed to confirm a device's location allows location information regarding devices to be employed in new ways, such as access control. We discuss both of these areas in further detail in Sections 2.3.1 and 2.3.2.

2.2.1 Localization Techniques

A device's location can be determined relative to other devices within a particular system, or at an absolute (real world) level. In order for a device's absolute location to be computed through localization techniques, at least some of the other devices within the system must be in possession of their own absolute location. This is accomplished either through the device being in possession of a GPS unit or through being assigned a fixed position and programmed with this set location information. These devices are commonly referred to as *beacons*. In order to calculate a device D's actual location, D must be able to calculate its location relative to at least one beacon within the system. With D's relative position known, it can then calculate its absolute location. The more beacons D can calculate its location relative to, the more accurate a location can be calculated. With only a single relative position known, D can compute a set of possible locations surrounding the beacon, based on the estimated distance. As the number of beacons increases, the size of this set decreases, with at least three required to compute a specific position for a 2d system.

Localization has been investigated in many different media settings, such as Ultra-Wideband (UWB), Infra Red (IR), Radio Frequency (RF) and Ultrasound (US). The primary method of localizing a device is through the employment of ranging techniques. The main ranging techniques employed focus on the measurement of three different signal attributes; the received signal strength, the angle at which a signal arrived and the time taken to receive a signal. Here, we give a brief outline of these techniques. These allow for the calculation of a device's location relative to at least one other device within the system. The greater the number of devices that participate in the localization process, the more accurate a location can be extracted from the process. A more detailed discussion of these techniques can be found in [25].

Received Signal Strength (RSS) In ranging approaches based on *Received Signal Strength* (RSS) measurements, the distance a signal travels is estimated based on the dif-

ference between a device's stated transmit power and the power of the signal received. This loss in power is the result of signal propagation over a distance, based on the assumption that a signal's strength decreases at a specific rate as it travels. Once the signal strength loss is calculated, either theoretical or empirical models can be used to extract an estimate regarding the distance over which the signal travelled. The models employed to estimate the distance traveled by a signal assume that the environment in which the signal is propagating is constant and does not contain any unexpected materials which may affect the rate at which the signal's power is expected to decrease. It also assumes that signal reflections (resulting in multipath signals) do not occur, as this can lead to incorrect distance estimations. Through the combination of distance estimates from multiple devices and their positions, a location for the source device can be extracted. However, a device may subvert this approach through supplying a false power level to the receiving devices, thus allowing the calculated power loss and consequentially the distance estimate to be controlled to a degree.

Angle of Arrival (AoA) In *Angle of Arrival* (AoA) approaches, the angle at which a signal arrives is estimated and a device's location extracted based on the combination of AoA information from multiple sources. The AoA technique is primarily employed within sensor networks, where nodes are equipped with an array of directional antennas rather than a single omni-directional one. The angle of arrival of a signal is defined as "the angle between the propagation direction of an incident wave and some reference direction, known as *orientation*" [72]. For each node, orientation is defined as a fixed direction and represented in degrees moving clockwise from North. Using the multiple directional antennas on the node in conjunction with the RSS of a signal allows the relative AoA to be estimated. Calculating the AoA of a signal can be employed to calculate a device's location through node co-operation. If node X is being localized, multiple other nodes in its vicinity attempt to estimate the AoA of a signal from X, relative to their orientation. This information can then be combined and a point of overlap found, giving a location. However, AoA approaches must contend with an issue that also affects RSS localization: multipath signals. These are caused by a signal hitting a surface in its path and reflecting back off it on another direction, resulting in a multipath signal. The ability of signals to reflect off surfaces means that the angle at which a signal is received may not be the

angle from which it was originally sent. The multipath problem can be addressed using maximum likelihood algorithms [76, 104], which calculate the most likely angle of arrival of a signal based on statistics.

Time Based Approaches There are two main time-based ranging techniques: *Time of Arrival* (ToA) and *Time Difference of Arrival* (TDoA). In both approaches, the time at which a signal reaches a receiving device is recorded and employed to calculate the distance between two devices. In ToA-based approaches, this time is used to calculate the time required for a signal to propagate from its source to the receiver. In practice, the most common method of accomplishing this is through timing the round trip of a signal using a challenge-response exchange, a concept also refered to as distance bounding [13]. However, in order for this approach to function correctly, the device being localized (D) must not be able to respond to a challenge signal prior to receiving it. If this stipulation is not upheld, D could transmit a response early, thereby reducing the round trip time calculated and decreasing the estimated distance between the receiver and D. We discuss distance bounding and its role in localization in greater detail in Chapter 3.

In the case of TDoA-based localization, a signal's recorded time of arrival is employed slightly differently. Rather than timing round trips between the device being localized (D) and a single receiver (R), D transmits a single signal which is received by several devices in its vicinity, and its time of arrival is recorded. The time at which D's signal arrives at each receiving device is slightly different, due to the difference in distance between D and each receiver. These time differences are then converted to distance differences and used to estimate D's location through hyperbolic computations. A hyperboloid is a surface with a constant distance difference from two points (in the case of localization, two devices). Where three receivers (therefore two hyperbola) are available, a device may be localized in two dimensions through finding the intersection of the hyperbola. Where four receivers are available, this process can be employed to localize the device in three dimensions. It must be noted that in order to function correctly, TDoA-based localization requires receiver devices to possess a tightly synchronised clock. This ensures the time of arrival recorded by each receiver is comparable to those recorded by the others involved. The most well known TDoA-based system is the Global Positioning System [27], which applies the TDoA technique to received satellite signals in order to compute a device's location relative to the

positions of the transmitting satellites. This approach to positioning is further discussed in Section 2.3.1.

2.2.2 Localization Attacks

As localization technology has increased in importance, more attention has been given to designing methods to subvert the techniques employed in the localization process. These attacks have been designed to interfere with the accuracy and functionality of localization. They range in scale from lone devices interfering with local traffic to collaborative efforts by multiple devices. There are two possible approaches which can be used to undermine the outcome of a localization attempt: attacks directed at the underlying localization techniques employed in the system and those which attempt to manipulate the system itself. These approaches are described below.

Common Attacks on Localization Techniques

One of the simplest attacks which can be attempted on a communications system is a *Denial of Service* (DoS) attack. The primary objective of a DoS attack in this situation is to prevent a specific device or devices within an area from localizing others or being localized themselves. In an RF-based system where an attacker does not have access to the internal workings of other devices, this is achieved through interfering with the devices' ability to receive and send messages (signals), usually through flooding the network with messages of its own. Where an attacker has access to specialized equipment, it can conduct an alternative DoS attack, using a signal jamming device to disrupt radio signals in the area. This prevents those devices present from receiving or sending messages, thus rendering them incapable of localizing or being localized using RF-based approaches. A similar jamming approach can also be employed on US and UWB systems, assuming the attacker is in possession of the correct jamming equipment.

However, while DoS attacks are effective at removing a device's ability to participate in localization, they are not particularly suited to manipulation of the process's outcome. In order for a malicious device to increase the distance estimated based on localization techniques, it must employ some form of *attenuation* attack. In an attenuation attack, specialised equipment, such as an absorbing barrier, decrease the speed and strength at which a signal can propagate. This makes it appear as though the signal has travelled

farther, thereby tricking the receiver into believing that the source device is farther from the receiver than it truly is. Inversely, in an *amplification* attack, a malicious device may employ specialised equipment, such as a directional antenna with an extended range, to achieve localization at a location closer to the receiver devices than it truly is. However, while amplification attacks can impact localization results where RSS is employed, the time-based approaches employed in localization are not as easily defrauded. This is because even though the signal transmitted has been strengthened, the time required for it to reach a receiver from its source cannot be altered. Therefore, in order to influence a receiver's localization result, malicious devices perform *collusion* or *proxy* attacks. These attacks operate at a system level rather than on localization techniques and are discussed in the following section.

Common Attacks on Localization Approaches

In this section, we discuss attacks aimed at localization approaches, rather than those aimed at altering the results produced by the localization techniques employed. The most common attack attempted on localization systems is the *collusion* attack. In this form of attack, a malicious device attempts to influence its own localization results by having a colluding device masquerade as the device being localized and transmit signals from a closer position to the receivers. The receivers therefore calculate the location of the colluding device, rather than that of the device being localized, thus providing a malicious device with a false location in closer proximity to the receivers. This form of attack is effective primarily due to the fact that ranging techniques (such as those described above) localize the source of a signal, not a specific device. Measures regarding the security of the exchange are left to the scheme in which the techniques are employed. Proxy attacks capitalise on this weakness by substituting the device being localized with an alternative source closer to the signal receivers. However, this weakness can be addressed by the localization scheme tying any signals (or messages) involved to the source device in some manner. This forces the malicious device to attempt a *terrorist fraud* [22] in order to deceive the system, where the messages are forwarded to the receiver by a colluding device acting as a proxy for the device originating the messages. This approach increases the amount of time required for the signal to reach the receiver. Therefore, unless there exists an accelerated means of communication between the colluding device and the device being localized, the

advantage of employing a colluder is lost, assuming the goal of the device being localized is to appear closer. If said device wished to appear farther from the recipient, a terrorist fraud attack (without the use of accelerated communications equipment) would allow it to do so. However, this can be achieved using far simpler approaches, such as by delaying the sending of its response message and falsifying any incriminating time stamping within the message.

Another form of collusion attack is the *wormhole attack* [47], in which an attacker records messages transmitted within its range. It then "tunnels" them to another area for retransmission, either by another node under its control or by a colluding agent within the network. "Tunnelling" involves transmitting the recorded messages directly from one agent to another, without forwarding through multiple nodes. In the case of tunneled distances longer than the range of a single wireless hop, or where the replay has a time-critical element, the colluding devices may employ an enhanced long-range wireless link or a direct wired link. In the case of the latter, systems involving distance bounding may find themselves vulnerable to terrorist fraud, due to the decrease in latency incurred by transmitting messages over a higher-speed connection.

Finally, there also exists a common form of attack with a similar impact to the collusion attack, but with only a single attacker involved. This is known as the *sybil attack* [26]. In this attack, a system believes itself to be employing several different devices in a localization, when in reality it employs only one. This is achieved through an attacker creating multiple identities for itself and attempting to pass them off as individual devices, a simple task without the presence of some central authority registering all devices. Where a central authority is employed, an alternative method may be used to circumvent this measure, where the attacker steals the identity of another device within the system. This approach may not be detected if the attacker prevents the legitimate owner of that identity from communicating with the central authority, or if the owner is not active at that point.

2.3 Localization Methods

While the focus of our work is on location verification, many security and design issues raised in verification approaches are also found in positioning systems. For this reason, we have conducted research into both positioning and verification approaches to localization.

In this section, we discuss the development of research in both areas, outlining a selection of schemes put forth and the subsequent improvements made by those following, up to the present day. This is summarized in Tables 2.1 and 2.2. We then discuss the contribution made by this work to the area of localization and how it addresses the issues of current approaches.

2.3.1 Positioning

Within this work, positioning is defined as the determination of a device's location, either relative to other devices in a system or on a global level. The Global Positioning System (GPS) is the most well known and widely used technology to position a device, employing TDoA to calculate the position of a device relative to a constellation of satellites orbiting the Earth. However, GPS readings can be forged, and even if users were prevented from interfering with the output of a GPS unit through including the GPS within a device's Tamper-Resistant Module, GPS signals can be spoofed [33]. This renders GPS an untrustworthy method for localization. The United States military has created a similar system to GPS known as the Precise Positioning Service (PPS) which encrypts all signals to prevent spoofing, but this technology is unavailable to civilians. An alternative to GPS, known as Galileo, has been proposed and is undergoing construction in Europe. Galileo's proposed design is a significant improvement on that of civilian level GPS, with an increased level of resistance to jamming in addition to including encryption within the signal, thus preventing spoofing [70]. However, although GPS and similar satellite-based systems can be useful for determining a device's position, their usefulness is limited to external locations. In addition to this, the level of accuracy reached by satellite-based systems decreases significantly when employed within highly built up areas. Therefore, alternative approaches to positioning are required.

The first major positioning system proposed was ActiveBadge [101], created by Want et al. Designed to track objects within a specific area, ActiveBadge functions through the combination of a wearable device periodically emitting its unique identity over IR and a fixed infrastructure of receivers which receive these beaconing messages. The beaconing message emitted by the device bounces off (but does not escape) the walls bounding the room, thus ensuring the signal fills the room. The receiver fixed within the room receives

the message and can infer that the device is located within its walls. However, IR is vulnerable to dead-spots and interference from other light sources, leaving devices unlocatable in some areas. Also, due to the poor range of IR, the ActiveBadge system scales poorly. Want et al presented an improved approach to positioning in the form of the ActiveBat system, which removed the technology's reliance on IR and instead employs US and RF TDoA techniques. In the ActiveBat system, a combination of base stations, receivers and devices are employed to calculate a device's location. An RF-based base station acts as a centralized starting pistol, transmitting an RF signal to each device in turn to indicate that they are to be localized. Upon receiving this signal, the device being localized sends an US signal, which is heard by the receivers in the vicinity. The receivers have also received the initialisation signal sent by the base station and calculate the time difference between receiving this and receiving the device's US signal. From this difference, each receiver can extract the distance between their location and that of the device being localized and send this information back to a central hub. This central hub can then combine the distances from multiple receivers at fixed locations and calculate an location for the device.

Despite addressing some issues within the ActiveBadge system, ActiveBat is heavily dependant on a centralized approach with a fixed infrastructure, thus limiting its applicability. In contrast, the Cricket [74] system, has been developed to be highly decentralized and reduce overheads. Though Cricket also employs US and RF techniques, the system operates in reverse to the approach taken by both ActiveBadge and ActiveBat. The devices being located (referred to in Cricket as *listeners*) do not transmit their location. Instead, beacons concurrently transmit RF messages indicating their location, along with an US signal. The listener records the location information from the received RF message, along with its ToA and waits to receive the corresponding US signal. Upon receipt of the US signal, the listener can compute the TDoA for the message and consequently calculate its distance from that beacon. It then combines the location information gathered from all beacons within range and extracts its own location. Through reversing the flow of information, the listener's privacy is preserved and the system does not rely on a centralized architecture. However, this approach is vulnerable to attack by malicious beacons broadcasting false information or transmitting their US signals prior to sending the RF messages, thus tampering with a listener's location calculation. In addition to this, the burden of

computation is placed on the listening devices, requiring an increased amount of power to be contained within the devices.

Despite Cricket's decentralized approach, there remains a reliance upon a pre-existing infrastructure of beacons. In the Ad-Hoc Localization System (AHLoS) [86], Savvides et al presented an alternative to this approach by designing a localization system which does not rely on any form of infrastructure. Instead, localization is done in an ad-hoc fashion, with those nodes already in possession of their location ("beacons") aiding in the localization of "unknown" nodes - i.e. nodes without knowledge of their own location. Once an unknown has calculated its location, it becomes a beacon and aids in the localization of other unknowns. Similar to Cricket and ActiveBat, AHLoS employs US and RF TDoA to calculate location, with beacons broadcasting their location simultaneously over RF and US. Also included within the system is a method of localizing unknowns where three beacons are not within range, however this approach has been questioned [93]. In addition to this, while the beacons are not technically infrastructure, some percentage must have been in possession of their location at the time of system initialisation. Due to the nature of the nodes employed in the system and its design as an indoor system, ruling out the use of GPS or similar systems to establish a location, this implies that some position information was input upon installation.

Many other systems have been proposed which include some variant on the approaches described above. Systems such as the Precision Asset Location (PAL) system [31, 30] employ TDoA techniques within an UWB setting to compute asset locations in an indoor setting. Similarly, an UWB relative location system was put forth by Correal et al [20] which employs distance bounding to calculate a location relative to other nodes within the system. However, within the schemes presented to this point, attacks such as the sybil attack [26], wormholes [47] and collusion attacks have been left undefended against. In particular, decentralized systems such as Cricket and AHLoS are particularly vulnerable due to their reliance on the word of other devices. The introduction of SeRLoc [54, 55], Lazos and Poovendran's Secure Range-Independent Localization scheme for wireless sensor networks brought this issue into the spotlight. Its range-free approach to localization removes any possibile vulnerabilities from attacks seeking to disrupt the measurement of distance between devices, as it does not rely on ranging techniques to compute a loca-

tion. Instead, locator nodes equipped with sectored antennas transmit beacon messages encrypted with a global symmetric key (shared by locators and regular sensor nodes), composed of their location and the sector of the antenna from which the message was sent. This sector information indicates the angle at which the message is transmitted at from the locator. The beacon message is received by devices in the locator's vicinity and employed to calculate their current location through combining the information received from multiple beacons to find their point of overlap. SeRLoC is robust against sybil attacks, with the exception of the compromise of a locator's set of secret data. SeRLoC is also robust against wormholes, as those with a retransmission point in close proximity to the source of the original beacon message are detected due to the antenna *sector uniqueness* property. This property states that no device can receive multiple beacons originating from the same locator which were sent through different antenna sectors, as they are transmitted on different, non-intersecting, trajectories from the locator. Additionally, even when beacons are tunneled a significant distance away, the presence of legitimate beacons in that area indicate to a sensor that it is under attack.

However, despite increasing security awareness within the area of localization and addressing both the sybil and wormhole attacks, SeRLoC remains vulnerable to jamming attacks. This issue is addressed in Lazos et al's Robust Position Estimation (ROPE) system [56]. ROPE combines SeRLoC's approach to location information dissemination with the Verifiable Multilateration (VM) technique put forth in SPINE [98, 95], Capkun and Hubaux's range-dependent secure positioning system based on distance bounding. In VM, the distance between the sensor being localized and each of at least three locators (at known positions) is calculated using distance bounding. Each distance is used to compute the radius of the circle centred at that locator's position, and the sensor's position is determined by calculating the point of intersection between these circles. Through employing at least three locators in this process, a triangle is formed around the sensor. Due to basic geometric principles, it is unable to maliciously alter its location with one locator without also correcting the distance results obtained from the others. If this correction is not performed, the device's malicious behaviour would be discovered, as the distances computed would not agree. Assuming that all distance bounding exchanges are performed simulateously, the employment of distance bounding prevents the sensor from achieving this, as

it cannot decrease the distance calculated between itself and a locator, only increase it. If it increases the distance between itself and a locator, it must decrease at least one of the other distances involved, which is impossible where distance bounding is employed. However, this functionality requires at least three locators to be in the vicinity of a sensor, therefore the number of locators required within the field is very high. ROPE improves on SPINE's VM implementation by providing an alternative approach to positioning for use if three or more locators are not available. In this approach, the sensor's location request is disseminated farther afield by the locators that initially received it. Those locators receiving the disseminated request send a message indicating their location and details of the area covered by the antenna employed to transmit the message. The initial sensor receives these messages and calculates its location based on the intersection of areas contained within and a pre-computed region of intersection. Through the inclusion of this back up approach, the number of locators required within the field is significantly decreased. While ROPE cannot protect against a complete jamming-based DoS attack, it is robust against selective jamming attacks, where attackers jam specific transmissions at will. If a sensor is capable of communicating with just one locator, it can estimate its position to some point within transmission range of the locator's known position. However, jamming to this level incurs a very high cost, making it infeasible to employ.

2.3.2 Location Verification

As mentioned previously, the focus of our work on localization has been in location verification. The concept of location verification is that a device may prove its location or presence in a particular area to a central entity or other device. Excluding those approaches which employ self-localization, many of the schemes described in the previous section for positioning can also be employed for the verification of a device's location. If a device's supplied location matches the location estimated by the positioning scheme, the supplied location is proven. However, a number of approaches have been designed solely to verify a device's provided location or presence in a given area, with the latter being designed as a method of access control [4]. These approaches rely primarily on time-based ranging, with many employing distance bounding [13] to confim a claiming device's presence within a given distance of another device.

One of the first distance bounding based location verification schemes was proposed by Sastry, Shankar and Wagner in [85]. In this work, they propose the Echo protocol, a protocol designed to gauge the upper bound on the distance between 2 devices, allowing the device to prove its presence in a particular location. This is accomplished using a combination of RF and US signal ToA measurements. RF is used by a verifier node V to transmit a packet containing a nonce to device D. When D receives this packet, he echoes the nonce back to V using ultrasound. As the reply relies on the initial message, D cannot cheat and transmit the packet early. Therefore D cannot pretend to be closer to V. However this protocol is vulnerable to a proxy attack, as a colluding device (C) between D and V can allow D to echo the nonce over RF as far as C. C can then replay the reply back to V using ultrasound. As RF has a faster propagation time than ultrasound, the attack would not be detected. Sastry et al also proposed a keyed variant [85] to this protocol, as the original protocol does not combat the localization vulnerability exploited by collusion attacks. It proves only that a device is in the area claimed, not that a specific device is located there. However, this variant is still susceptible to the same attack as the encrypted message can be forwarded by the colluding device as easily as the unencrypted message could be.

Also in 2003, Waters & Felten proposed the Proximity-Proving Protocol [102], a protocol also designed to prove a device's presence in a particular area. Similar to Sastry et al's work, Waters & Felten's approach uses distance bounding in the round trip times of packets to calculate an upper bound on the distance between devices, However, its reliance solely on RF removes the primary vulnerability found in Sastry et al's approach: its susceptibility to collusion attacks due to the inclusion of US technology in the challenge-response exchange. An additional advantage of employing RF is that it is a increasingly common technology, present in the majority of mobile nodes, therefore it is far more widely employable. Despite this improvement, the Proximity-Proving protocol remains vulnerable to the terrorist fraud form of collusion attack. This is due to the protocol lacking some form of tie between the distance bounding portion of the protocol and the device being located. A colluding device acting as a proxy for a malicious prover could participate in the distance bounding aspect of the protocol in place of the prover. This shrinks the time taken for a reply to reach the receiver, making the prover appear closer. In addition to this, the

protocol relies upon an infrastructure of trusted entities, limiting its range of applicability.

In [97], Capkun et al proposed a novel approach to address a previously unconsidered aspect of the verification problem. Traditionally, where an infrastructure is employed in location verification, the locations of the locators or base stations are known in advance. This gives an attacker an advantage in its attempt to deceive them, as it is possible to compute the time required to give a specific distance in distance bounding, an advantage of particular relevance when dealing with non-simultaneous VM. Capkun et al's proposal employs hidden or mobile base stations within the verification process, removing that advantage from the attacker. In the case of verification employing covert (hidden) base stations (CBS), the device being verified has no knowledge of the location of the CBS and transmits blindly. When verifying using mobile base stations (MBS), the device may know the location of the MBS prior to receiving a localization challenge. However, when the device acts on this challenge (after waiting for a given time limit), the MBS has changed location. These base stations measure the TDoA between messages sent simultaneously over RF and US from the device being localized. As the device cannot tell where the base stations are in either case, it does not know how to correct the timing of the messages to achieve a different location. Although it is possible to identify and locate a device based on fingerprinting [77], meaning that a hidden base station may be detectable and thus lose its advantage over an attacker, this is not yet a practical attack.

In addition to those infrastructure-dependent localization schemes developed specifically to verify a device's location, some positioning systems have the potential to also be amended for use as verification schemes. In the case of SPINE, location verification is easily achieved once a device is within range of at least three locators. If the locators deem the device being verified to be in the same location as has been claimed by that device, then the claim is verified. However, this requires a high number of locators to be deployed in the field, a requirement which ROPE addresses. Due to its self localization approach, ROPE is not immediately employable as a verification scheme, though authors do include a simplistic approach to verification. In this approach, a device proves itself to be within range of a single locator node through distance bounding. Unfortunately, no mention is made of tying the interaction to a specific device, leaving the approach open to collusion. The authors note that this is only a simplistic approach and that a more thorough solution

could be designed.

Recently, Capkun et al have presented an updated version of their approach to secure verification which includes a move away from reliance upon infrastructure. In [99], verification within mobile ad hoc networks is achieved using neighbouring devices in place of base stations. The neighbouring devices employ TDoA on a signed message which has been broadcast simultaneously over US and RF to calculate a distance to the device being verified, although the authors do stipulate that any form of passive ranging may be employed within the scheme. The transmitted message contains the location being claimed by the device, along with its identity and a timestamp. If the distance computed by the neighbouring device corresponds to the distance from there to the included location, that neighbour issues a signed and encrypted statement indicating that the location was verified to it. These statements can either be transmitted directly back to a central server or sent via the device being verified. However, while this does address the issue of infrastructure reliance, the scheme does not address the issue of colluding neighbours working either for or against the device being verified and leaks personal data to all devices in its vicinity. Additionally, unless all devices are tightly time-synchronised, a terrorist fraud form of collusion attack is still possible as the only method of retaining a sense of timeliness is through the inclusion of a timestamp.

2.3.3 Our contribution: the Secure Location Verification Proof Gathering Protocol

Tables 2.1 and 2.2 summarise a variety of the proposed approaches to positioning and verification within the area of localization. A closer look at the "Weaknesses" column reveals that though significant progress has been made in the area, one of the biggest issues is that of infrastructure-reliance. While schemes such as AHLoS [86] and Capkun et al's Mobile/Hidden/Covert Base Stations [97] do address this issue, they are in the minority and, in the case of AHLoS, sacrifice participant privacy in order to achieve infrastructure-independance. The approach presented in this work attempts to create an alternative method of localization, specifically in the area of location verification, that is not only infrastructure-independant but also addresses two other major issues in the area - terrorist fraud and private information leakage (privacy).

Name	Technique	Strengths	Weaknesses
GPS/PPS/ Galileo	TDoA using satellites	Highly accurate	Vulnerable to spoofing (depending on approach); Only feasible for external locations; Relies on infrastructure; Accuracy decreases in heavily built-up areas
ActiveBadge	Device emits location-limited beacons	Functions indoors	Relies on infrastructure; Scales poorly; Dead zone issues due to use of IR
ActiveBat	US & RF TDoA	Functions indoors; Employs more reliable technology (RF & US instead of IR)	Relies on centralized system & infrastructure
Cricket	US & RF TDoA	Functions indoors; Highly decentralized; Preserves privacy of participants	Vulnerable to spoofing; Requires infrastructure; All computation done by device being positioned
AHLos	US & RF TDoA	Functions indoors; No need for infrastructure; Ad-hoc/communal approach	Requires some level of start-up knowledge; No regard for privacy issues
SeRLoc	Combining beacon messages with source locations & angles of transmission	Functions indoors; No reliance on ranging techniques; Robust against Sybil attacks & wormholes	Vulnerable to jamming; Sector uniqueness property may cause honest messages to be discarded due to reflections; Reliant on infrastructure
ROPE & SPINE	Verifiable Multilateration (Distance Bounding)	Functions indoors; VM can detect false location claims; High tolerance to jamming (ROPE)	SPINE vulnerable to jamming; ROPE reliant on infrastructure

Table 2.1: Summary of Positioning-based Localization Approaches

Name	Technique	Strengths	Weaknesses
Echo Protocol	RF & US TDoA	-	US renders scheme vulnerable to replay attack; Requires infrastructure; No identifiers used to tie specific device to location being proven (addressed in encrypted varient); Vulnerable to jamming
Proximity-Proving Protocol	Distance Bounding	Employs a more generic medium (RF only); Protects privacy of participants	Vulnerable to terrorist fraud; Requires infrastructure;
Covert/Hidden/ Mobile Base Stations	US & RF TDoA	Device being localized cannot predict where receiver will be; Recent revision removes reliance on infrastructure	Possible future vulnerability to device fingerprinting, thus removing main strength

Table 2.2: Summary of Verification-based Localization Approaches

Privacy

With information regarding participants (such as identity or location) being transmitted during the course of location verification, privacy is of grave concern. In order to protect the integrity and privacy of this information, we have created the Secure Location Verification Proof Gathering Protocol (SLVPGP) [37]. This protocol provides a three tiered approach to security and privacy, employing an increasing combination of digital signatures and encryption to protect any private data being transmitted. The structure of the protocol has been based on that of the Proximity-Proving protocol, however alterations have been made that address some of the original scheme's vulnerabilities, notably those which leave it vulnerable to terrorist fraud. The SLVPGP's basic design has been extended to three practical protocols, with each increasing the level of security provided to protect the privacy of participants. The first level is on par with schemes such as ROPE, which call for the unsecured transmitting of private information. The second level secures these transmissions through encryption for the devices involved in the exchange, which protects those involved from eavesdroppers but not malicious internal nodes. The third level provides complete privacy, preventing the leakage of any personal information regarding any participant.

Terrorist Fraud

As discussed previously, a terrorist fraud (or "proxy attack") is where two or more devices collude to convince the system that a specific device is in one location when it is really in another, through proxying its messages to the correct area. Many verification schemes (including both the Echo protocol and the Proximity Proving protocol) are vulnerable to terrorist fraud, making reliable location verification difficult. In order to protect our location verification approach from terrorist fraud, we have presented a new distance bounding metric for use in the SLVPGP. This Binary Metric is used to confirm that a distance bounding exchange has not been proxied. It does so by timing the exchange and comparing the result to a previously calculated limit. The limit is calculated to be less than that required to proxy a message over a single additional hop, and so if a distance bounding exchange is less than this limit, it must be a direct exchange. The design of the SLVPGP prevents any circumventing of this metric by ensuring that the distance bounding portion of the protocol contains a connection to the device being verified. By including a digital signature on the distance bounding response, it forces the claiming device to produce this

message itself. If it wishes to prove a false location, it must therefore proxy any distance bounding responses back to the correct area, which would be detected by the binary metric. If the message has not been proxied, it must originate in the area, thus providing proof of a device's location. An exception to this statement is where amplification equipment is employed by the claiming device to extend its reception and transmission ranges. This exception is discussed more fully in Chapter 3.

Reliance on Infrastructure

Similar to Capkun et al's ad hoc verification scheme, the SLVPGP does not rely upon infrastructure to enable verification. Instead, it employs neighbouring devices to supply proof to a central server for final judgement. By employing only RF distance bounding to confirm or deny the presence of a specific device at a claimed location, the majority of wireless devices can participate in location claims, making it a highly generic approach. Just as in schemes employing an existing infrastructure of locator devices designed to verify a device's location, a device can gather proof of its presence at a specific location by distance bounding with random neighbouring devices to confirm it is within their transmission range.

However, it is not enough to merely allow a claiming device to gather proof of its location from all devices in its vicinity and supply this as evidence, as there is far too much scope for abuse. For this reason, we have designed a management entity known as the Verifier, which serves as a back-end to the system. This entity manages the selection of suitable local devices from a pool of volunteers and calculates the trustworthiness of the devices involved based on their behaviour during past interactions with the system. The Verifier is also responsible for extracting a final verdict on the veracity of a location claim, based on the evidence supplied to it by the claiming device.

2.4 Summary

In this chapter, we have introduced the area of localization. Research in this area focuses primarily on positioning approaches, with some attention being paid to the concept of verifying a device's location claim or presence in a specific area (location verification). It is within this sub-strand that our research lies.

We defined the concept of localization before giving a brief description of some of the key ranging techniques employed in the area of localization, including TDoA and ToA. We then discussed some of the major threats to ranging techniques and localization schemes as a whole, such as DoS, amplification and collusion attacks.

With the area of localization and its common techniques outlined, we proceeded to discuss the research areas of positioning and location verification, moving from the Active-Badge [101] positioning system to more recent works such as Capkun et al's hidden and mobile base stations [99]. We discussed each scheme's contribution to the progression of research, while identifying unaddressed flaws which have still to be addressed.

Finally, we discussed our own work on the SLVPGP and how it relates to already existing research, identifying our inspiration for the scheme's structure and how it addresses open issues within other schemes in the area.

Chapter 3

Distance Bounding and the Use of a Binary Metric

3.1 Introduction

In Chapter 2, we outlined a number of ranging techniques for use in localization, designed to estimate the location of a device relative to others within its system or on a more global level. A common method of computing this information was through the estimation of distance between devices. One such method of estimating distance is distance bounding [13], in which a challenge-response scenario is used to gauge the time required for a message to be received. This information allows a distance to be extracted, through combining the time required for a message to be received with the speed of the communications medium being employed to give a maximum possible distance traveled in that space of time.

In this chapter, we discuss the technique of distance bounding and its use within our localization system (Section 3.2). We outline an alternative metric which foregoes the precise calculations traditionally employed by distance bounding and indicates whether a pair of devices are in direct communication, or if one device is attempting a collusion attack (Section 3.3). We investigate the usefulness of the binary metric in IEEE 802.3 [6] (wired) and IEEE 802.11 [5] (wireless) environments through emulation, discussed in Sections 3.4 and 3.5. In Section 3.6, we introduce the concept of a window of acceptance and outline a method for its computation, before discussing the need for authentication within the exchange in Section 3.7. Finally, we summarise the contents of the chapter in

Section 3.8.

3.2 Distance Bounding

Before discussing our work on the binary metric, an alternative metric for use in distance bounding, we outline the technique of distance bounding itself. We discuss proxy attacks - the main form of attack launched on this form of ranging technique - on both a basic and a more technologically complex level.

3.2.1 What is Distance Bounding?

Distance bounding is a process in which bits are sent rapidly between two devices, in order to establish a limit on the distance between the two. The process involves a device A (the *sender*) sending a challenge bit to device B (the *prover*) and timing the delay between transmission and receiving the corresponding response bit. This delay time is then used to calculate an upper bound on the distance between devices A and B. In practice, a series of these exchanges is done to lessen the effects of network delays on the overall result. This process is primarily useful in limiting the distance within which a device could be found. This makes it a prime technique for use in the location proving area.

Distance bounding is employed within this research to provide proof of a device's presence in the area for use by the location verification system. The technique allows the location verification system to distinguish whether a device is within a claimed area, through the claiming device (the Claimant) distance bounding with neighbouring devices (Proof Providers) from that area. In the distance bounding exchange, the Claimant plays the role of the prover, with the Proof Provider taking the role of the sender. This proof is used by the system to corroborate a device's claim of being in a particular location at a given time. However, distance bounding is vulnerable to many attacks [44]. Examples of these attacks include prematurely responding to challenges and proxy attacks with the aid of colluding devices. Proxy attacks are discussed further in Section 3.2.2.

The premature response method of attack is technically simple, whereby the prover sends its response prior to even receiving the challenge bit. This gives the impression that the prover is closer to the sender than they are, by shortening the time taken to respond to the challenge. This attack can be prevented in a number of ways. Examples of these

include forcing the bits being sent back by the prover to depend on those bits received from the sender, or by the sender sending out bits with randomly chosen delay times, preventing the prover from anticipating the time at which the bit will be sent.

3.2.2 Proxy Attacks

As discussed previously, a proxy (collusion) attack on a ranging technique is where two or more devices work together in order to convince a third that one of the attackers is in its vicinity. In the case of a proxy attack on distance bounding, the prover colludes with one or more interim devices in the hope of convincing the sender to unknowingly give a false reading for its location. When a sender directs a challenge to the prover, the colluding or "proxy" device can either respond in place of the prover or forward the challenge on to receive the appropriate response. If the proxy has the ability to respond as the prover without detection, the attack is invisible to the sender. However, the response may require some piece of information that only the prover has access to and cannot share. An example of such a situation would be where the appropriate response is to encrypt the frame, using encryption keys only available in a tamper resistant unit. In this case, the proxy is forced to forward all challenges received from the sender to the prover and wait for the appropriate response in return. The forwarding of messages between the receiving device and the proxy in this manner greatly delays the sending of the response back to the sender, thus allowing the attack to be detected.

Within our system, the simplest form of proxy attack involves only a single additional device acting as a proxy between the Proof Provider (the sender) and the Claimant (the prover). However, this basic form attack has two distinct subtypes, with vastly differing levels of impact. In the first and simplest basic subtype, all devices communicate using the same medium, thereby meaning that a proxy attack would require at least twice the time taken by a direct challenge-response exchange. In the second subtype, the proxy and Claimant communicate over a much faster medium, thus providing the attackers with an advantage and decreasing the time required to complete the challege-response exchange. While the time required in this situation would still be greater than that required to complete an honest exchange, the additional hops incurred could, depending on the communications medium employed, be so rapidly accomplished that an attack would be undetectable.

For example, if a Proof Provider was to communicate with a proxy device using wireless communications, it would expect a reply back using the same medium. This medium is relatively slow, particularly in comparison with wired communications, which can reach speeds of over 1 gigabit/s. If the proxy was in possession of a direct wired connection to the Claimant, it could easily complete the additional hops required to transmit the challenge to the Claimant and receive back its response for forwarding without adding any significant period of time to the overall exchange, due to wireless's slow transmission speed. However, this approach requires the existance of a wired connection between the Claimant and its proxy, a requirement that is not usually common of a mobile node. Therefore, when discussing proxy attacks throughout this work, we deal solely with the first subtype, where all devices employ the same communications medium, and acknowledge that further work is required to investigate the detectability of the second form of basic proxy attack. Although a similar effect could be achieved within the wireless medium through the use of a directional antenna with increased power levels, the impact on time would be greater than where a wired connection is employed, increasing the likelihood of detectability.

In order to discover whether a proxy attack could successfully receive a false positive on a location claim through completing distance bounding within the time frame of an honest exchange, a security analysis of the first basic proxy subtype scenario was conducted. This analysis was carried out through the emulation of both honest and proxied exchanges in various network settings, and the resulting times from each were compared.

3.3 The Binary Metric

In this section, we outline the premise and examine the usefulness of a binary metric for use with distance bounding in a wireless network. The binary metric proposed is a yes/no "visibility" metric. It is assumed that a device can only participate in a distance bounding exchange via a direct connection to a verifying device or through a proxying device. Therefore if a Claimant appears visible to a Proof Provider, it is either participating honestly or is attempting to mount a proxy attack. The binary metric employs this fact to detect whether or not a Claimant is a single wireless network hop from its Proof Provider. As the distance over which a message can travel in a single hop on a wireless network is limited, this definition provides an upper bound on the possible distance between the

Claimant and its Proof Provider.

We employ the binary metric within the Secure Location Verification Proof Gathering Protocol (SLVPGP), which protects the proof gathering exchange. The binary metric protects the integrity of the verdict produced by distance bounding through confirming that the Claimant involved is not attempting to perpetrate a proxy attack. Neighbouring devices (Proof Providers) within the wireless network produce a yes or no verdict indicating whether the Claimant is able to communicate with them within a reasonable time limit. It is these verdicts which are utilised as a tool to verify the location claim, in place of the round trip times used by other approaches. If a distance bounding exchange results in a positive binary metric verdict, then the Claimant participating in that exchange is both able to communicate with the Proof Provider and is not communicating via a proxy, proving that they are in the area. One possible exception to this is where a Claimant employs amplification equipment to extend its transmission range, thereby allowing its direct message range to extend beyond that of a normal device.

3.3.1 Honest vs Proxy Exchanges

The proposed use of a binary metric in distance bounding removes the reliance on timing to calculate an upper bound on the possible distance between the Claimant and a Proof Provider. However when dealing with the binary metric, a upper bound is placed on the allowable delay time between the Proof Provider sending its challenge and receiving the Claimant's response. This limit is included to distinguish between a proxy attack and an honest distance bounding exchange. The calculation of the limit is further discussed in Section 3.6. When a Proof Provider engages in distance bounding with an honest Claimant, the response time is 2x+d, where d is the computational delay and x is the message's transmission time for a single network hop. However, when a Proof Provider engages in distance bounding with a malicious Claimant via a proxy, the response time must be greater than 2x+d. This is because during the course of a proxy attack, the message being transmitted undergoes one or more additional network hops in each direction, rather than only a single hop. Finally, when a Proof Provider distance bounds with a Claimant employing amplification equipment, the time required to complete an exchange is 2y+d (where y is the message's transmission time for a single amplified hop). This is because the

signal is required to travel farther in both directions. However, unlike in proxied exchanges, the computational delay remains unchanged as only one set of devices are involved in the exchange.

In an honest exchange, a challenge is sent by the Proof Provider to the Claimant and the Claimant's response is sent directly back. However, during a proxy attack, the Proof Provider's challenge is sent to a proxy device within its transmission range. The proxy then forwards it on through at least one network hop to reach the Claimant. The Claimant's response must then also travel the extra steps to reach the proxy before being relayed back to the Proof Provider. It is these additional message hops that increase the response time of a proxy attack. We propose to use this flaw in the proxy attack's design to detect its occurrence. A Claimant cannot communicate with its Proof Providers without either a direct or proxied connection. Therefore, the ability of the binary metric to detect the presence of a proxy attack also allows it to function as a verification method, proving that the Claimant is in the claimed area. However, while this is the case for comparing proxied and direct exchanges, it is not necessarily so for amplified exchanges. Computational delay is higher than signal transmission times, and while proxied exchanges incur multiple rounds of this form of delay (with the number depending on the number of proxy devices involved), amplified exchanges incur only one. This difference means that amplified exchanges can be expected to require less time to complete than proxied exchanges, and therefore may not be as easily detectable, particularly in a congested/high-traffic network.

In order for the binary metric's method of detection to function, there must be a substantial difference between the times required to complete a proxy attack in comparison to that required for an honest exchange. If there is no overlap between the possible time taken for an honest Claimant to complete its distance bounding and a malicious Claimant to perform a proxy attack, then the occurrence of a proxy attack is easily detectable. One danger when dealing with this method of detection is that the computational delay will drown out the difference in transmission times between an honest exchange and a proxy attack. This would make honest and proxied exchanges indistinguishable from each other. In order to confirm that this issue does not pose a threat to the functionality of the approach, we have conducted multiple emulations in both IEEE 802.3 (wired) and IEEE 802.11 (wireless) environments. These emulations are discussed in Sections 3.4 and 3.5.

The research presented in this work focuses on the ability of the binary metric to distinguish between direct and proxied exchanges. Due to equipment limitations, no emulations have been attempted to ascertain the difference between direct and amplified exchanges. Therefore, while the binary metric has been confirmed to distinguish between direct and proxied exchanges, there is no evidence to support its capability of detecting amplified exchanges, meaning that a device may still reside beyond the area of the claimed location. This possibility represents an avenue for future research.

As is common in this area of research, the value of d (computational delay) is left as an unknown, with no attempt to address it. The Proof Provider/receiving device removes the computational delay incurred by itself, as this value is known, leaving a final value composed of the transmission time plus the computational delay incurred by any other participating devices. Therefore the possible distance traveled will be slightly less than that calculated, as part of the time attributed to transmission has been used for computation. As the definition of distance bounding is a process to compute the maximum possible distance a signal can travel, and is not intended to give an exact distance, this is considered acceptable.

3.3.2 What Does a "Yes" Verdict Mean?

When using the binary metric to detect the presence of a proxy attack, there are two possible verdicts which a Proof Provider may provide: the Claimant is visible to it during distance bounding ("yes"), or the Claimant is not visible ("no"). While these two verdicts appear to be straightforward, their meanings are not so. If a Claimant is found to be visible by a Proof Provider and a proxy attack is not deemed to have occured, then that Claimant is deemed to be within the area of that Proof Provider. It is impossible for a Proof Provider to mistakenly find that a Claimant is in its area when it is not, due to the employment of digital signatures on the response portion of the exchange. This is dealt with in further detail in Section 3.7.

If a Proof Provider deems a Claimant visible during distance bounding, it must first have received multiple valid responses to its challenges from that Claimant. Within this work, we assume that the Claimant is not in possession of amplification equipment and therefore if it is not in the claimed area, it requires the assistance of a proxying device

located within the area. Without the aid of a proxy device at a location closer to the Proof Provider than its own or amplification equipment to increase its reception and transmission capabilities, the Claimant is not capable of proving itself to be in the area of the Proof Provider. Proxy attacks are ruled out through employing an upper bound on an acceptable round trip time, as discussed in Section 3.3.1, and so any positive visibility results on the part of a Proof Provider cannot be a mistake.

3.3.3 What Does a "No" Verdict Mean?

Unlike in the case of a positive visibility verdict, there are many possible reasons for a Proof Provider deeming a Claimant not visible during the distance bounding process. The three main reasons for a negative visibility verdict are: the Claimant is not present in the area, the Claimant is in the area but not within range of the Proof Provider or that there are network issues preventing the Claimant from completing distance bounding within the time limit.

The simplest reason for a negative verdict is that the Claimant is not present in the area. There are various causes for this to occur, both innocent (where a Claimant mistakenly having made a false location claim) and malicious (where a Claimant attempts to deceive the system). Additionally, the device could have moved on from that location before the distance bounding occurred, a distinct danger if the system is based in the area of VANETs [79], where participants are usually in motion during exchanges.

The remaining reasons for a negative verdict are caused by a failure of the technology upon which this system is built. Although the Claimant is in the area in both cases, a falsely negative verdict is received. Either the range of the Proof Provider is too limited to allow for the Claimant to distance bound directly with it, or the network over which distance bounding is to occur is too unreliable to allow it. Distance bounding is undertaken multiple times to lessen the effects of network issues on the eventual outcome. However, if the network is consistantly poor, either too noisy or too lossy, then the Claimant will not be able to distance bound successfully within its allowed time frame. It is for these reasons that we do not factor negative visibility verdicts into the calculation of location at the end of the distance bounding process. In some cases, negative verdicts are genuinely deserved. However, there are also multiple scenarios in which a false negative is received, which could

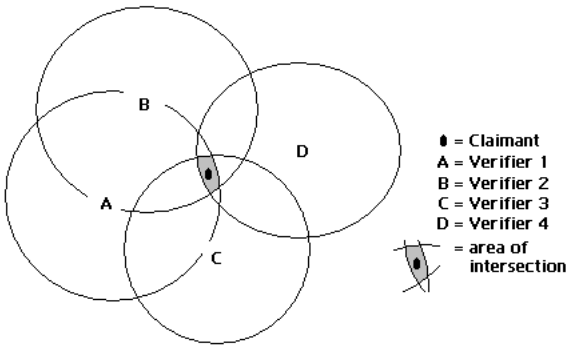


Figure 3.1: Employing the Binary Metric Results

pollute the location calculations. Therefore, as there can be no mistaken positive verdicts we employ only these in the final location calculation. This is discussed further in Chapter 6.

3.3.4 The Binary Metric vs Precise Calculations

Typically, distance bounding through location verification is used to verify that a device is within range of a specific resource or location ([80, 43, 96, 94]). This is achieved through calculating an upper bound on the possible distance between the device and required resource/location and confirming that this location is within an allowable range. In order to discover this upper bound, the delay between a Proof Provider sending a challenge and receiving the appropriate response back from the Claimant is precisely timed. This round trip time is then used to calculate a greatest possible distance, through using the speed at which the signal can travel. In the case of the Echo protocol [85], both the speed of sound and the speed of light are factored into the calculation of distance. When dealing with distance bounding in wireless networks, such as with Waters and Felten's proximity proving protocol [102], the processing time on the part of the Proof Provider is subtracted from the round trip time. This figure is halved and multiplied by the speed of light (approximately $3 * 10^8$m/s) to calculate an upper bound on the possible distance between the Claimant and the Proof Provider.

The above approach is similar to that applied in this work, however the work put forth here does not require the upper bound limit to limit the distance. Instead, this upper bound merely distinguishes a proxy attack from an honest exchange. Receiving a positive visibility verdict from multiple Proof Providers regarding the same location claim allows

for the area in which that Claimant could be located in to be reduced down through a form of triangulation. In this process, the possible broadcast ranges for each of the Proof Providers contacted are noted, and the broadcast areas (usually circles or spheres) are overlaid. This is shown in Figure 3.1. In this figure, the Claimant has distance bounded with Proof Providers A, B, C and D. This has resulted in the Claimant's possible location being reduced down to a minimally sized area of intersection. This area of intersection of the transmission ranges of all Proof Providers involved is the only region in which the Claimant could possibly be located, thus giving a location for the Claimant and potentially proving the location claim being investigated. This concept is employed in the extraction of a more precise location in Section 6.8 of Chapter 6.

3.4 Distance Bounding Emulations in an IEEE 802.3 Wired Network

In order for the binary metric to be considered practical for reliable location verification, it must be able to detect whether a proxy attack is underway during a distance bounding exchange. If the metric cannot detect this attack, results gained with this approach would not be trustworthy. The nature of a proxy attack on an exchange increases the transmission time of any given message and it is this increase in response time which the Proof Provider checks for when calculating its visibility verdict, in order to confirm that a proxy attack is not being carried out. However, one of the main issues with this method of proxy detection is the possibility that the time lost to network latencies and computational delays would drown out the difference between honest and proxy exchanges. In order to confirm that this does not threaten the integrity of the detection approach, we have emulated the round trip of a message in both situations. An emulation of the two scenarios was carried out rather than employing a simulation tool such as ns-2[1], as we wished to gather practical results rather than relying upon assumptions made within third party software. The emulations were performed on the DCU computing network at normal hours, with multiple users active at the time. As the scenarios were emulated on a real network, the exchanges incurred a normal level of packet loss and network delay due to existing network traffic.

The experiments conducted deal with two scenarios. In the first scenario, devices

[1] http://www. isi. edu/nsnam/ns ns-2

Figure 3.2: The Path of a Direct Message (IEEE802.3)

behave honestly when engaged in a distance bounding exchange. In the second, they behave dishonestly and conduct a proxy attack on the system ("cheating"). Ethernet frames are sent back and forth for distance bounding. These were chosen to minimise the effect of computational delay on the results, as they are the most basic networking building blocks available and are at the lowest point in the networking stack. This decreases the delay between when the frame is received by the device and when it is analysed by the emulation program. UDP has been selected (User Datagram Protocol, from the Internet Protocol suite) [73] as the packet type for use when transmitting between the proxy device and the final device. This was done as unlike TCP (Transmission Control Protocol) [69], UDP gives the option of performing no error checking on the packet for the sake of speed, giving as fast a result as possible while still using a routed protocol.

3.4.1 IEEE 802.3 Emulation Outlines

To emulate distance bounding in a vehicular network, we conducted experiments within an IEEE 802.3 wired CAT5 Ethernet network. Though the transmission protocol employed within vehicular networks is based on this (IEEE 802.11 is based on IEEE 802.3), it is inherently different due to the medium over which it transmits. Wireless transmissions are lossy by nature, with a higher level of transmissions lost than over wired networks due to a high level of interference from objects in the environment. However, vehicular networks have the advantage of being ad-hoc in nature. This allows for direct transmission between devices, thus cutting out extraneous devices such as switches which add further hops to the journey. Mobile networks cannot achieve such high transmission speeds as an IEEE 802.3 network due to the limitations imposed on the bandwidth by the radio spectrum, as there is only a small segment available for wireless transmissions. With a wired network there is a dedicated transmission medium with no other objects interfering and less competition for slots in which to send data.

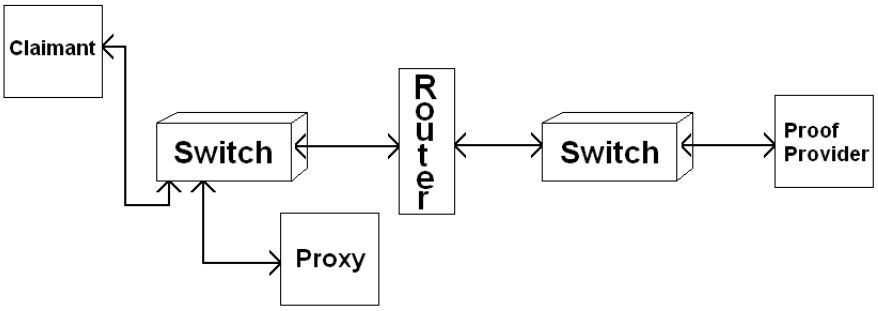


Figure 3.3: The Path of a Proxied Message (IEEE802.3)

However, though there are differences between the theoretical mobile network setting of IEEE 802.11 and the IEEE 802.3 network in which these emulations are set, the results remain useful as they represent the optimal possible round trip time for an attack. The minimum round trip time produced by our emulations for a proxy attack scenario is still nowhere near a reasonable response time for an honest exchange. This is despite higher transmission speeds and fewer packet losses than in the theoretical mobile network, thus proving that there is no possibility of a proxy attack deceiving the system and receving a positive result from distance bounding.

The direct exchange of a message between two honest agents in a network is emulated by the transmission of a single ethernet frame to the ethernet address of the receiving agent (modelling the Claimant). The receiving agent then creates a new frame and transmits this frame back to the ethernet address of the initial agent (modelling the Proof Provider). These frames are forwarded through a switch device, causing each frame to be transmitted twice, once from the sender to the switch and a second time from the switch to the intended recipient. This path is shown in Figure 3.2.

The emulation of a proxy attack is structured in a similar manner to that of the direct exchange emulation. However, due to the increase in participating agents, the number of agents and the type of message container are altered. The emulation models a simple form of proxy attack - the message is only proxied through a single extra agent, but there is additional infrastructure involvement also. As the message must be proxied outside of the local area, its contents are copied from an ethernet frame to a UDP packet, to allow for IP (Internet Protocol) routing.

The path a message must travel during a proxy attack is depicted in Figure 3.3. The

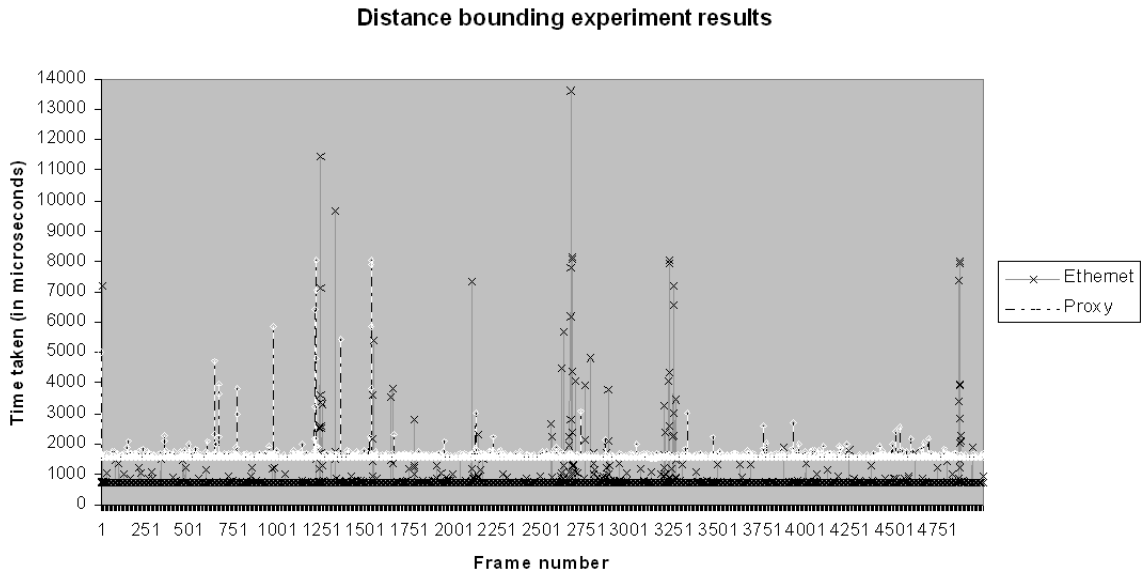


Figure 3.4: Emulation Results of Proxied and Direct Exchanges Performed Over an IEEE 802.3 Network

message begins as an ethernet frame, which is transmitted from the sending agent (modelling the Proof Provider) to the proxy agent's ethernet address via a switch device. The contents of this frame are then copied into a new UDP packet and sent to the IP address of the true recipient (the agent modelling the malicious Claimant). As the packet is being routed, it travels from the proxy agent to a switch and from there is forwarded to a router. The router forwards the packet to the correct switch and this switch transmits it to the true recipient. The true recipient copies the received contents to a new UDP packet and returns the new packet back over the same route it arrived on.

The network equipment employed for these emulations was comprised of three pcs, each with a 100 mb/s (megabit per second) network card connected by 10/100 ethernet to a Cisco 2498G switch. The switch was connected to a Cisco 6509 router using a fibre backbone, with a speed of 1 gigabit/s. The switch employed a "store and forward" approach to frame forwarding, to allow for complete error checking. A "store and forward" approach involves the switch storing all message data until the entire frame has been received, rather than starting the forwarding process prior to receiving the complete message.

3.4.2 IEEE 802.3 Emulation Results

The graph seen in Figure 3.4 displays the result times of both the direct round trip and proxy attack emulations for 5000 message exchanges. The lower line (solid black with grey

peaks) represents the direct exchange emulation results, while the upper line (in white) represents the proxy exchange emulation times. The graph shows that the average round trip time of an honest exchange is roughly half that of the average round trip time for a proxy exchange, with a clear break between the time required for direct exchanges and that required for proxied. This break between average round trip times provides a large window within which to establish a acceptance limit. This concept is discussed further in Sections 3.5.2 and 3.6.

The average time required by a direct exchange emulation run to complete a message exchange was discovered to be 710 microseconds. This is less than half the average time of 1560 microseconds required by a proxied exchange. This difference in round trip times is clear evidence that there is enough of a distinction between a direct exchange and a proxied exchange to allow for detection of attacks on distance bounding and the binary visibility metric. Aalthough there were a small number of overlaps between the results returned for direct exchanges and proxy results, these can be attributed to network issues and are an expected occurrence within our figures. The addition of multiple iterations of distance bounding removes this issue from question, as the odds of network problems repeating over a series of distance bounding exchanges are very low.

3.5 Distance Bounding Emulations in an IEEE 802.11 Wireless Network

While the employment of a time-based metric to distinguish honest from proxied exchanges initially seems like an ideal solution to the collusion detection problem, if the window for a reasonable exchange is extended, proxy attacks again become an issue. The IEEE 802.3 practical emulations of distance bounding provided a positive result on the impossibility of a proxy attack going undetected. However, the issue remains as to whether or not employing the binary metric to distance bounding set in an IEEE 802.11 wireless network will work in practice. The practical reality of data transmission in wireless networks is that due to only one device in a certain range having transmission capability at any one time, data collision algorithms are heavily relied upon. These algorithms slow the sending of frames through the employment of techniques to avoid collisions. Collision detection algorithms also employ delay windows in retransmission approaches, again slowing down

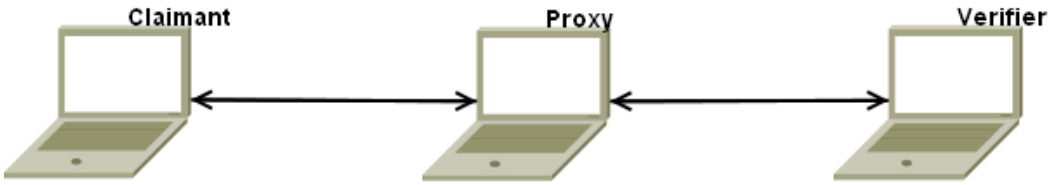


Figure 3.5: The Path of an Emulated Proxied Exchange in an IEEE 802.11 Environment

sending speeds. Finally, in addition to interference from non-computing devices, even with algorithms to detect and avoid collisions, at times packets simply get lost. These issues mean that employing the binary metric to distance bounding in a wireless setting may simply not be feasible, due to the high time windows which would be required to allow for a successful exchange. Therefore, in order to confirm the practicality and suitability of employing the metric in this environment, emulations similar to those described within an IEEE 802.3 setting have been carried out. These emulations indicate whether the variation in time required for a direct round trip will mask the extra time required to carry out a proxy attack on the devices. If this was the case, the binary metric would be ineffectual in this situation as it cannot differenciate between a direct exchange between honest devices and a proxy attack involving malicious devices. However, unlike in the case of the IEEE 802.3 emulations, the IEEE 802.11 emulations were not carried out on a live network, due to a lack of available equipment. Instead, the emulations were performed on a small test network, featuring only the devices involved in the emulations. Although this does decrease the accuracy of the results, it also allows the results to be analysed to detect the different between direct and proxied communications without any interference. While the emulation results do include reasonable computational delay, the network load and traffic levels were significantly lower than those expected in practice. This issue is further discussed in Section 3.5.2.

3.5.1 IEEE 802.11 Emulation Outlines

The network equipment employed for the IEEE 802.11 emulations was comprised of three laptop computers, each running ubuntu with a minimum of one gigabyte of memory and an IEEE 802.11b network card. The units were connected to an ad-hoc network with a maximum connection speed of 11mb/s. As in the case of our wired emulations, the direct exchange of a message between two honest neighbouring agents is emulated through

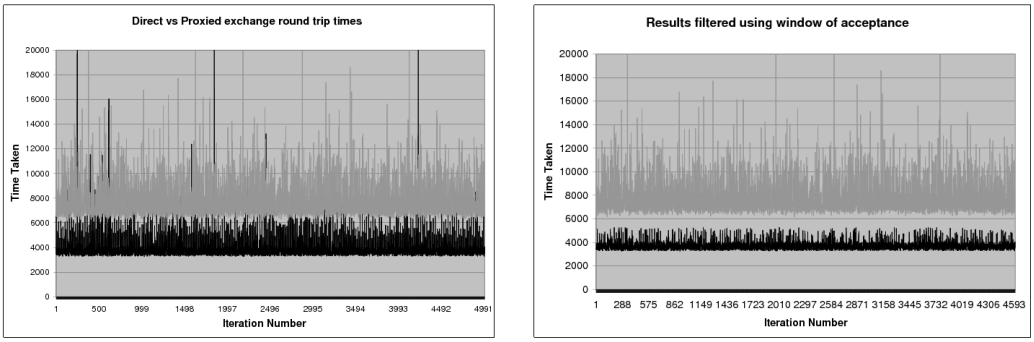
the transmission of a single ethernet frame by the initialising agent (modelling the Proof Provider) to the ethernet address of the receiving agent (modelling the Claimant). The receiving agent copies the contents of the received frame into a new message and transmits this frame back to the initialising agent's ethernet address.

Unlike the case of an honest message exchange, a proxy attack on distance bounding may be executed in a number of ways. However, at its most basic level, every proxy attack involves at least one extra device forwarding messages in order to enable communications between a two devices, in this case a Claimant and its Proof Provider. The functionality of this interim device may be repeated over multiple other devices, allowing a Claimant's communication capability to be extended enormously. In extreme cases, an attack may even involve the routing of messages over the internet to a Claimant located many miles from its claimed location. As discussed in Section 3.2.2, the proxy attack scenario considered within this work involves both the Claimant and the Proof Provider communicating via a single proxy device, using a wireless communications medium. For this reason, the proxy scenario employed as a test of the binary metric within this work is composed of only three devices: the Claimant, a Proof Provider and a single additional device functioning as a proxy (Figure 3.5).

In this emulation, the Proof Provider transmits a frame to the address it believes belongs to the Claimant. In reality, the frame is transmitted to an interim proxying device. The proxy device then forwards the received frame on to the true Claimant. As in the honest emulation, the Claimant copies the contents of the received frame into a new frame. The new frame is transmitted back to the Proof Provider via the proxying device, thus allowing the Claimant to trick the Proof Provider into believing it to be located closer than it is in reality.

3.5.2 IEEE 802.11 Emulation Results

Graph (a) seen in Figure 3.6 displays the results recorded for 5000 message exchanges within direct and proxied round trip attack emulations. The lower line (black) represents the results recorded for direct exchanges, while the upper line (grey) displays the proxied exchange results. The graph shows that the average round trip time of an honest exchange is significantly lower than that required for a proxied exchange. It was discovered that



(a) (b)

Figure 3.6: Emulation Results for Direct vs Proxied Exchanges

the time required for a message to complete a round trip between two devices fluctuated over several thousand microseconds. However, a fraction of the results of direct exchanges overlapped with those received for proxied exchanges. This overlap poses a problem for the practicality of the binary metric, as without a clear division between the expected time for the completion of a direct exchange and that required for a proxied exchange, the metric cannot reliably function. However, upon further analysis of these figures, a *window of acceptance* (WoA) emerged (Section 3.6), within which 92.04% or 4602 of the 5000 direct exchanges graphed were received. This WoA was discovered to extend to within 2000 microseconds of the fastest round trip time achieved within that emulation's run. The WoA provides an upper bound on the acceptable amount of time taken to complete a direct round trip. Graph (B) shown in Figure 3.6 depicts the filtered result set, where any exchanges with a round trip time of greater than 2000 microseconds above the fastest exchange are removed from the original result-set from which graph (a) is generated. By removing any results beyond the upper limit on the window of acceptance, the issue of overlapping timings is resolved, as demonstrated by graph (B). Note that there is now a clear distinction between direct and proxied exchange results. We applied the window of acceptance filter to a extended set of 27 direct emulation runs, each containing 10000 exchanges. When applied to the results of each emulation run, an average of 93.67% were deemed acceptable as direct exchanges.

The average fastest time achieved for a round trip over the 27 emulation runs was found to be 3256 microseconds, with an average of 9362 frames per emulation received within 2000 microseconds of this time (5256 microseconds). Therefore if this figure were enacted as an upper limit on the time in which a frame could acceptably be received in during direct

distance bounding, 93.62% of the frames transmitted during these experiments would be accepted by a Proof Provider. The inclusion of multiple iterations of distance bounding in the interaction aims to reduce the risk that a single frame's slow round trip would remove the ability of the Claimant to prove itself visible.

The results gathered from the outlined emulations support the hypothesis that the binary metric can function within an IEEE 802.11 environment. However, further study is required to evaluate the effect of a densely populated network on the performance of the metric. While the emulations outlined have been conducted within an environment comparable to a real world wireless ad hoc network, the population of the network vastly differs from that of an ad hoc network. The environment in which the emulations were conducted contained only the devices playing a role in the emulation itself. This was done to gain a clear picture of whether or not the binary metric would function in an IEEE 802.11 environment and to discover what the minimum possible exchange time achieveable was. A more populated environment would interfere with the gathering of this information, as participating devices would be required to compete for slots in which to transmit messages. In addition to the level of collisions and required retransmissions, this will cause a slower response rate, masking any information regarding minimum exchange times. In the future, it is advised that the metric be tested in a larger network to discover its viability.

Whilst carrying out these emulations, it was discovered that employing devices with vastly varied specifications and capabilities led to inaccurate and inconclusive distance bounding results. This is due to the need for predictability in the resulting round trip times. If Claimant A) can complete an exchange in 3564 microseconds, but Claimant B) has slower equipment and can only complete the exchange in 12834 microseconds, it will appear to the Proof Provider as though Claimant B) is attempting to carry out a proxy attack even if Claimant B) is behaving honestly. For this reason, equipment used to carry out distance bounding employing the binary metric must be standardised to achieve roughly the same speed of exchange. An alternative approach to standardisation could be employed whereby the Proof Provider is informed of the speed to be expected from a Claimant prior to completing distance bounding. However, this is vulnerable to attack, as the Claimant could lie about its minimum possible exchange speed, thus allowing enough time to carry out a proxy attack without detection.

3.6 Defining the Window of Acceptance

In the previous section, it was shown that a window of acceptance can be defined to provide an upper bound on the time taken for a round trip in order to distinguish between a direct and a proxied exchange. Those exchanges with round trip times within the window are deemed acceptable and considered to be direct exchanges, while those with round trip times of greater than the maximum time in the window are rejected. Therefore, the window of acceptance forms the basis of the binary metric, through providing it with a method of distinguishing direct from proxied exchanges.

In the examples shown in the previous section, the window was computed retroactively, based on the results gathered. However, this is not a practical approach and so some method of computing a window of acceptance based on the current state of the surrounding network is required. We propose a scheme in which the network speed is periodically assessed, thus providing devices with a relatively up to date estimate of the network's performance and the probable performance to be expected for an exchange. This assessment is achieved using a two stage process, with the first stage estimating a reasonable time frame for direct exchanges and the second estimating the same for a proxied exchange.

1. $A \rightarrow B : R_{dir}$
2. $B \rightarrow A : N_B$
3. $A \rightarrow B : N_B, N_A$
4. $B \rightarrow A : N_A$

In the first stage, basic distance bounding is performed with multiple devices in the area to gauge the local network's current performance capability. This is illustrated above, using abstract protocol notation. For the sake of clarity, an index of this notation has been provided at the beginning of this work. Device A initiates the process through transmitting a direct estimation request (R_{dir}) to a local device of its choice, B. In practice, this may be done using broadcasting over the local network, advertising that A wishes to gauge the speed of the network. B starts its timer and responds with a nonce (N_B) indicating that it has begun the estimation process. A starts its own timer and replies to this with a message

containing both B's nonce and a new nonce of its own (N_A). Finally, upon receipt of the nonce pair, B stops its timer and responds to A with A's nonce. A then stops its own timer, ending the process. Both B and A can then compute the time required for a direct exchange. This process is repeated multiple times in order to better gauge the network's current level of performance.

In the second stage, distance bounding including intentional proxying is performed to estimate the time required to complete a proxy exchange with a single additional device. A selects an interim device, B, through which it can proxy messages. B then selects a device to forward these messages on to (using the same approach as A), thus creating a proxy exchange "circuit". In addition to providing both the initiating (A) and final destination (C) devices with proxied exchange timings, this approach also provides the device acting as a proxy (B) with direct exchange timings from two devices, A and C.

1. $A \rightarrow B : R_{px}$
2. $B \rightarrow C : R_{px}, Ack_{px}$
3. $C \rightarrow B : N_C$
4. $B \rightarrow A : N_C, N_{B_1}$
5. $A \rightarrow B : N_A, N_{B_1}, N_C$
6. $B \rightarrow C : N_A, N_{B_2}, N_C$
7. $C \rightarrow B : N_A, N_{B_2}$
8. $B \rightarrow A : N_A$

The design for second-stage performance estimation is shown above, described using basic protocol notation. In this exchange, device A initiates the estimation through transmitting a proxied estimation request (R_{px}) to a random local device, B. B, noting that there is no acknowledgement included in this message, forwards the request on to a third random local device, C, along with an acknowledgement of the presence of a proxy in the chain (Ack_{px}). C, upon receiving the request to estimate proxied timings and knowing that a chain is already in place due to the inclusion of Ack_{px} in the request message, starts its timer and transmits its nonce (N_C), beginning the exchange. B, having received C's

message, starts its own timer and sends both C's nonce and the first of its own nonces (N_{B_1}) to A. A receives this message and starts its own timer, then attaches its nonce (N_A) to the message and sends it back to B. B stops its timer, records the time (which will be used to compute the time required to complete a direct exchange between itself and A) and starts the timer again, transmitting both C and A's nonces back to C along with the second of its nonces (N_{B_2}). C receives this message and stops its timer, then sends back both N_{B_2} and N_A to B. B receives the pair of nonces from C and stops its timer for the second time before forwarding the nonce N_A to A. Finally, A stops its timer and computes the time required to complete proxied distance bounding between itself and C, while B computes the time required to complete direct distance bounding between both itself and C and itself and A. As in the case of direct performance estimation, this multi-hop process is repeated with multiple devices to better gauge the local network's current performance level.

With both first and second stage measurements taken, a device can then compare the fastest proxied results with the slowest direct exchange results and set an upper bound to create that device's acceptance window. This boundary should be selected such that a clear distinction is created between the two sets of results. It should be less than the fastest proxied result acheived, but should also accept a reasonable percentage of direct round trip times. Additionally, by each device computing its own window of acceptance, there is a greater degree of flexibility within the system. However, there are two main drawbacks to this approach to gauging acceptance windows - device windows may vary greatly and it relies upon honest behaviour on the part of the participating devices.

While the process described in this section does allow information to be gathered regarding the current performance level of the local network, it relies upon honest behaviour among neighbours. If a device wished to skew the windows of acceptance for other devices in its vicinity, it could cause artificially high measurements for those entering into the estimation protocols with it through holding messages rather than responding immediately. This attack would be undetectable, except in an area where device behaviour is predominently honest. In this situation, the results provided or caused by a malicious device would be overridden by those generated using honest devices.

The second issue with gauging acceptance windows in this manner is the permittance

of devices to set their own windows. While this may provide the system with a greater level of flexibility, the results reflected by these windows may not be equally reliable. Under the system's current design, each device's acceptance window is unknown to all but that device, including the Verifier. Therefore even if one device's window is excessively small or large, the results it determines carry equal weight to those based on more reasonably sized windows. This issue can be addressed without the standardization of window sizes, through the provision of information on each device's individual window size along with any verdict information. However, this approach is vulnerable to abuse due to its reliance upon honest behaviour on the part of the participants, repeating the secondary flaw in the original design.

Finally, it should also be noted that the size of a device's window does not guarantee that that device will behave accordingly, i.e. a device with a very small window of acceptance will not necessarily mean that the device will always report its results honestly. For example, a device may employ a very strong window, composed of only a small acceptance range, within which any messages received are sure to be honest responses, but then lie about the number of responses it received within the window. This aspect of device trustworthiness is dealt with in Chapter 6.

3.7 Authenticating the Exchange

In order for distance bounding to function as a method of detecting a terrorist fraud attack, the Claimant must be required to actively participate in the exchange. Fulfilling this requirement ensures that the challenge reaches the location of the Claimant and not just an interim proxy device. A common approach to fulfilling this requirement employs pre-commitment to tie the Claimant to exchange. In this scheme, distance bounding is comprised of a series of single bit exchanges, rather than complete nonces. Prior to the timed portion of the exchange, the Claimant commits to a sequence of bits for use in generating its responses. The Claimant XOR's a received challenge bit with the corresponding bit from its pre-committed sequence before returning it to the Proof Provider. As the generation of this commitment employs a secret key shared between the prover and a Proof Provider, the Claimant is tied to the sequence. This approach was first proposed in Brands & Chaum's initial discussion of distance bounding [13], and has inspired many

others. However, it does not truly ensure that the Claimant is involved in the challenge-response portion of the exchange. Agents wishing to deceive a Proof Provider regarding the location of a Claimant can circumvent this approach through supplying an intermediate device with the sequence and having them reply in place of the Claimant. Some schemes, such as [13] and [96], also require that the process terminate with the Claimant signing some message generated from the bits received and supplying this message to the Proof Provider for verification. However, this stage is not timed and therefore a dishonest Claimant could compute this message himself and supply it either directly to the Proof Provider or to the proxy for forwarding without detection.

This flaw is also found in another common approach, employing tables of values calculated based on two nonces, one supplied by each party in the exchange. The response to each challenge can be found in the table, saving time usually required for calculation during the time-critical section of the exchange. As in the case of XOR-ing with pre-commitment, a malicious Claimant could supply a colluding device with this information prior to the timed phase of the exchange, thereby successfully deceiving the Proof Provider. An example of this approach can be found in Hancke & Kuhn's RFID distance bounding protocol [43].

In order to achieve this condition within the system, an online form of authentication has been employed, where authentication is done during the timed phase of the exchange. A digital signature has been included on the return leg of the challenge-response exchange, forcing the Claimant to receive the message and compose the corresponding reply, rather than a proxy device doing so in its stead. This also almost erases the possibility of the Claimant guessing the appropriate response and transmitting it prior to receiving the message.

When dealing with authentication during distance bounding, the concept of employing any form of encryption is usually dismissed. This is due to the belief that when using distance bounding to gauge physical distance, the time required to encrypt something will overpower the time required to send a message on a round trip. In particular, Clulow et al [19] argue that this approach introduces inaccuracy into the calculation of distances based on response times, in addition to allowing an advantage to those with more powerful hardware. However, the impact of authentication using encryption techniques upon the

time required for a distance bounding exchange may be removed. In order for a distance bounding exchange to be completed, the return message must be authenticated by some device. Therefore, authentication must be performed at some point within the exchange.

Through employing tamper-resistant devices to complete the required calculations within a known amount of time, the process can be standardised and the impact of authentication upon the time required for an exchange is removed. Therefore, in order for the binary metric to function accurately, it requires the use of standardised tamper-resistant hardware. As the time needed for authentication is known due to the use of standardised hardware, it can then be subtracted from the overall time, leaving only the time taken to complete an exchange. As this approach to distance bounding is not intended as a method of gauging distance itself, the issue of introducing inaccuracy into the distance calculated from these figures is removed, thereby negating both aspects of the argument.

3.8 Summary

In this chapter, we discussed the technique of distance bounding, as put forth by Brands and Chaum. We have proposed the concept of the binary metric, a metric which differenciates between an honest distance bounding exchange and an exchange in which a proxy attack is being perpetrated. Under this metric, the results of a distance bounding exchange no longer indicate an upper bound upon the distance between two devices. Instead, distance bounding determines whether the time required for the exchange to take place is reasonable for a direct exchange. We have acknowledged the limitations on the applicability of the metric, put in place by the assumption that Claimants do not have access to amplification equipment allowing them to extend reception and transmission range. We have defined both the positive and negative meanings of the metric, as only a positive is intuitively understood. In the case of a positive result, there can be no misunderstanding or mistake, as the exchange includes authentication. However in the case of a negative outcome, a direct exchange may have been in progress but due to issues beyond the detection of the metric, the delay in receiving responses causes the time required for an exchange to fall outside the window of acceptance. We have presented a method of calculating the window of acceptance in practice, using neighbouring devices to estimate the time required for both a direct exchange and an exchange proxied via one additional device. We

have demonstrated the applicability of this metric over both IEEE 802.3 and IEEE 802.11 networks through emulations of both direct and proxied distance bounding exchanges. Our analysis of the IEEE 802.11 network applicability is limited to networks containing only the devices participating in an exchange, therefore it is recommended to pursue further trials in more densely populated networks to investigate the effect of increased network traffic upon the time required for both proxied and direct distance bounding exchanges.

In addition to dealing with the detection of a proxy attack, we have outlined the need for some form of tie between a participating Claimant and the distance bounding exchange, in order to ensure that the device itself participates. We have discussed the concept of committing to values for use within the distance bounding process prior to entering into the timed phase of the process, but dismissed this approach as it can easily be circumvented by colluding devices wishing to deceive the system. We have proposed the use of digital signatures on the response leg of the distance bounding exchange, in order to tie the Claimant to the exchange during its timed phase. We address the issue of impractibility through proposing the use of tamper-resistent hardware to standardise the time required for a signature to be generated. The standardisation of this requirement allows for its removal from the time taken for an exchange, allowing for an accurate round trip time to be established whilst including online authentication of the participating Claimant.

Chapter 4

The SLVPGP - A Security Protocol for the Protection of the Location Verification Process

4.1 Introduction

As wireless networks become increasingly ubiquitous and location becomes a highly sought-after piece of context information, the demand for a method of securely locating a device has increased dramatically. When dealing with VANETs in particular, this ability has many novel applications for users, including the possibility of vehicular insurance directly related to the area in which the vehicle commonly travels. In general, LBS [87] are commonplace, however traditionally they rely upon self-location, with no guarantee of the location being accurate (Chapter 2). As we discussed in Chapter 2, researchers have proposed a variety of positioning and location verification algorithms for use in sensor and ad-hoc networks. However, many of these algorithms rely on a pre-existing infrastructure in order to function, an extremely limiting requirement. In addition to this, many do not include a method of verifying a specific device's participation in the distance estimation process, leaving them vulnerable to collusion attacks.

In this chapter we outline our approach to the gathering of evidence for use by a device to prove its location - the Secure Location Verification Proof Gathering Protocol (SLVPGP) [37]. Our protocol design aims to satisfy three security properties, authentication of origin,

confidentiality of location and anonymity of identity, in order to provide a complete level of security for the protection of the proofs gathered and the privacy of those involved. We make use of the binary metric, also dicussed in Chapter 3, to provide a method of detecting the execution of a proxy attack, allowing a Proof Provider to come to a verdict regarding the presence of a Claimant within its transmission range. It is assumed that the binary metric is an accurate method of gauging a Claimant's presence within range of a Proof Provider. Once this is the case, the SLVPGP will function as intended, i.e. the accuracy of the binary metric is a required property for the accurate functionality of the SLVPGP. If the binary metric is unable to correctly infer whether a Claimant is within range of its Proof Providers, the resulting verdicts supplied within the SLVPGP cannot be relied upon. In order to provide flexibility in the level of overheads incurred when attempting the verification of a location, we have developed three versions of the protocol, each with a different level of security and consequentially a different level of overhead costs incurred.

The structure of this chapter is as follows: In Section 4.2 we describe the system model and its assumptions. We outline possible attacks on the system and discuss the reasons for the selection of our three security properties. In Section 4.3 we outline the protocol's basic design, devoid of any security. We discuss the security concerns and vulnerabilities which the complete protocol should address. In Section 4.4 we repeatedly extend the basic protocol to create three versions with increasing levels of security. In Section 4.5 we analyse and compare the costs incurred by each extension. Finally, in Section 4.6 we summarize the chapter.

4.2 The Secure Location Verification Proof Gathering Protocol

In this section we discuss the technical information pertaining to the Secure Location Verification Proof Gathering Protocol (SLVPGP). We outline the system model, terminology employed and assumptions made regarding those nodes involved, the powers of an intruder and the system itself. We discuss the relevent threats which the SLVPGP must protect against and the security properties required in order to meet our definition of a complete and secure protocol. Finally, we briefly outline the role of the Verifier within the protocol, a topic further elaborated on in Chapter 6.

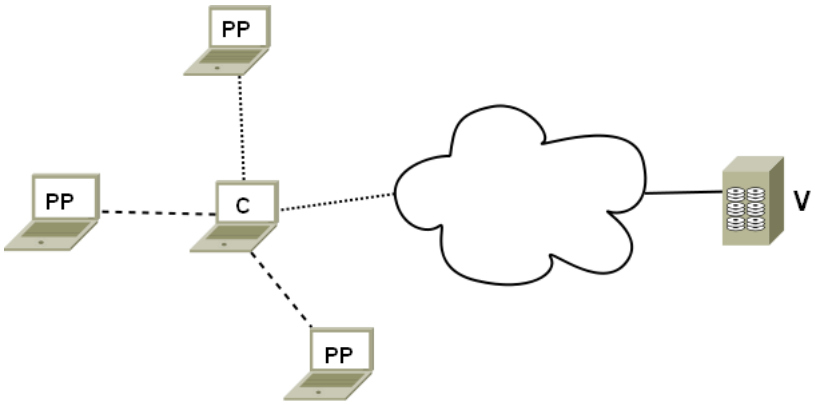


Figure 4.1: The Structure of the Location Verification System

4.2.1 System Model, Assumptions and Terminology

Our system consists of a set of untrusted mobile nodes and a central trusted entity knows as the Verifier (V). An untrusted device known as a Claimant (C) wishes to prove its location to the Verifier within an untrusted IEEE 802.11 based environment [5]. The Claimant sends a message to the Verifier stating its identity and claimed location. This message is known as a location claim. The Verifier provides a list of devices in the vicinity of the location included within the location claim for use as Proof Providers (PPs). The Claimant engages in distance bounding (see Chapter 3) with each of the Proof Providers to ascertain that it is within transmission range of that device, thus proving itself to that particular Proof Provider. It is assumed that the distance bounding process employed will provide an accurate reading of the proximity of the Claimant to a Proof Provider. The ability of distance bounding to accurately place a device in an area is an underlying security property of the SLVPGP and is required for its correct functionality. It is also assumed that proxy/terrorist fraud attacks will be detected through the use of the distance bounding employed. The results of the distance bounding exchanges are supplied to the Verifier via the Claimant. The Verifier then makes its decision based on this proof information. This situation is illustrated in Figure 4.1.

The system was initially designed with the intention of deployment within Vehicular Ad-Hoc Networks [79], and as such employs the standard technological assumptions for that setting (outlined below). However, during development it was realised that the high levels of mobility reached by vehicular networks could impact the accuracy and usefulness of the system in this environment. With devices moving at speeds of 50 kilometers per hour

and up, and with the potential to reach upwards of 120 kilometers per hour, the duration of connectivity between devices is unknown. Therefore the ability of a vehicular device to gather evidence from others in its vicinity is questionable. For this reason, VANETs are one of the hardest environments in which to prove a device's location using proof gathered from local devices. As there was no access to a physical VANET for testing purposes, there is currently no evidence to gauge the system's practicality in this environment.

However, networks composed of less powerful devices with lower levels of mobility are highly suited to the system. The communications medium employed by this approach to location verification is standard IEEE802.11 wireless transmission, the same employed by all internet-ready mobile devices. The rate of advancement in power and technology in hand-held mobile devices is high, with Moore's law of increasing power and decreasing size holding true. Therefore, smaller devices have become powerful enough to successfully deal with complex calcuations and encryption/decryption at high speeds. Additionally they can now support high levels of communication without worries over battery consumption. For these reasons, assuming they meet the requirements outlined below, this location verification system is employable on smaller devices.

Communications within the system are secured using public key cryptography, employing a sufficiently secure keysize. Possible examples of suitable asymmetric cryptosystems include RSA [82] and Elliptic Curve Cryptography (ECC) [53, 64]. In order to provide a suitably strong resistance to attacks, the keysize required by RSA is quite large, with keys as large as 512 bits factorised and their cryptographic strength broken [18]. In comparison, the same level of security can be provided by ECC using keys of a much smaller size. For example, a 1024 bit RSA key provides equivalent security as a 160 bit ECC key. However, while the data transmitted within the system is only useful for a short period of time after its transmission, using a lower key-strength would leave the key vulnerable to attack and discovery. This is a particular danger if a key-owner is malicious, as they have control over the data being encrypted and could attempt a chosen-plaintext attack to compute their private key. Therefore, all keys employed within the system should be of significant cryptographic strength.

We assume that all nodes are capable of encrypting and decrypting messages and generating and verifying digital signatures, employing the relevant cryptographic keys. As

the Claimant and Proof Providers are untrusted devices, we do not trust them to retain direct control of their key pairs. Instead, it is assumed that they contain tamper-resistant modules in which all cryptographic keys are stored, along with a reasonably accurate clock to provide loose time synchronization for timestamps. The storing of a device's keys in such a module prevents the sharing of keys and other private information with others in an attempt to sabotage the system's security. This tamper-resistance is central to the functionality of the location verification system. If a device can gain control of their own keys or the keys of others, they can share them with other malicious entities. If this occurs, the authentication aspect provided by cryptography would be lost, and a malicious device could pass itself off as another without fear of detection.

Within this location verification system, a device's public key functions as their identity. This allows any device the ability to communicate with another in ciphertext once it possesses that device's "identity", rather than requiring consultation of a table to access the correct public key. In order to provide an additional layer of security and privacy to all participants in the system, each device's tamper-resistant unit contains a list of multiple cryptographic key pairs. This provides multiple "identities" for use within the system. However, the use of public keys as identities creates a limitation on the system, as where a device's identity is required to be kept secret (to ensure anonymity), it cannot receive encrypted communications. This key pair list can be reloaded with new values, maintaining their freshness and reducing the probability of a device being traceable or linked to a previous "identity". The Verifier has knowledge of all public keys for each device within the system, allowing it to match participation using a particular "identity" to the correct device and preventing devices from successfully performing a sybil attack (Chapter 2).

4.2.2 Threat Model

Honest nodes behave exactly as their roles within the system dictate. They can communicate with any device within transmission range of their location. In addition to this, they can communicate with the Verifier. They can receive all messages broadcast within their range, but do not act on messages not intended for them. Malicious nodes fall into three categories: malicious Claimants, malicious Proof Providers and intruders, i.e. malicious

nodes external to the exchange. Malicious nodes cannot occupy the role of the Verifier as this is a trusted, un-compromisable entity. All malicious nodes share the capabilities of honest nodes, in that they can communicate with any device within their transmission range, and can communicate with the Verifier. They can receive any message transmitted within their range, but unlike honest nodes, they can act on messages not intended for them. If a message transmitted within their range is sent in cleartext, they have complete knowledge of its contents. However, if an overheard message is transmitted in ciphertext, the eavesdropper can only gain access to its contents either through procuring the correct decryption key from its owner or through breaking through the message's encryption. Malicious nodes can also manipulate and retransmit received messages in an attempt to circumvent the system's security or damage the chances of participants.

A malicious Claimant is a malicious node participating in the protocol in the role of Claimant. The aim of a malicious Claimant is to manipulate the location verification system to provide it with a verified false location claim. As it is a genuine participant in the claim, it may have access to information that an intruder would not have, such as the identities and locations of all other participants. It possesses its own public and private key pair, along with being provided the public keys of other participants. It can make both false and genuine location claims, and can select which pieces of proof are forwarded on to the Verifier in support of its claim.

A malicious Proof Provider is a malicious node participating in the protocol as a Proof Provider. A malicious Proof Provider may wish to cause a Claimant's claim to fail or it may wish to lie in order to support the claim of a friendly device. While it has access to more information than an intruder, it cannot manipulate the process in the same ways as a Claimant. It can decide whether or not to provide a proof for a location claim and can control the proof's contents, but cannot ensure that it will be included in the Claimant's final list of proofs. Additionally, it has access to information on the Claimant involved in a claim, but not on other Proof Providers.

When analysing the vulnerabilities of the SLVPGP, multiple attacks are considered. The majority of these attacks are attempted by malicious Claimants attempting to influence the result of their location verification. The first considered is the *guessing attack*. In this scenario, a malicious Claimant attempts to guess the correct response to a challenge

and transmit it prior to even receiving the challenge itself. If this process was successful, the malicious Claimant would successfully prove itself to be within range of the Proof Provider. However, this attack is only a realistic threat to schemes employing single bit challenges. As the SLVPGP employs long random numbers (nonces) as challenges, the odds of a malicious Claimant successfully guessing the correct response even once are prohibitively high. Therefore, the SLVPGP is not considered vulnerable to guessing attacks.

The second attack considered is a form of collusion attack known as the *terrorist fraud* (Chapter 2). This attack was first described in [22] and involves a malicious Claimant collaborating with another device willing to act as a man-in-the-middle between the Claimanat and the verifying device. It allows the malicious Claimant to convince the Proof Provider that he is closer than he truly is. This attack in particular is of great concern when dealing with a location verification system, as the system must verify that a specific device is at a specified location, not just some device is at that location. The terrorist fraud or proxy attack is addressed within the SLVPGP through the employment of the binary metric (see Chapter 3) in conjunction with distance bounding.

A more general attack which the SLVPGP is vulnerable to is the *denial of service (DoS) attack* (Chapter 2). In this attack, a malicious device in the area of the Claimant attempts to prevent the participation of other devices through flooding the network with useless packets (congestion-based DoS). The transmission of these packets prevents any other devices from also transmitting, effectively cutting off communication. An extension of this attack is the *distributed denial of service attack (DDoS)*, where multiple devices participate in the denial of service attack. DoS attacks can also occur accidentally, where other devices in the area are all attempting to transmit at once. This reduces the speed at which messages can be transmitted to a crawl. Proposed solutions to the DoS attack lie at the MAC layer of the network and are discussed in [41] and [11]. Also of concern is a protocol-specific form of DoS attack, whereby malicious devices transmit false proof messages to the Claimant in the hope of deceiving him into forwarding them back to the Verifier as part of his final proof collection. This attack and its solution are further explained in Section 4.4.3.

Finally, in a *snooping attack*, an intruder attempts to gain access to location or identification information through eavesdropping on protocol runs. As any device within trans-

mission range can receive all transmitted messages, this method of attack is both easily achieved and potentially highly destructive to the security of the participants. This attack is thwarted through the employment of encryption on messages containing private information. If messages are no longer transmitted in plaintext, only devices possessing the correct decryption key are capable of learning their contents upon receipt.

However, although cryptography is used on messages transmitted within protocol extensions two and three, the absence of encryption during the distance bounding portion of the protocol constitutes a security issue and leaves the SLVPGP vulnerable to a more subtle form of eavesdropping attack, relying on identifiers. When an observer eavesdrops on traffic being transmitted in its vicinity, it can see the contents of the message headers, along with the data contained within the message's payload. If the data is encrypted and the observer is not in possession of the appropriate key, it cannot identify whether it is a stage within SLVPGP or a completely unrelated message. As the nonce messages are not encrypted, the contents and structure of the message are visible, meaning that the observer knows that the devices exchanging these messages are involved in a run of the protocol. Though the values of $H_{i,k}$ and $H'_{i,k}$ are not connectable by an observer, $\mathcal{N}''_i$ is present in both distance bounding messages. By matching the $\mathcal{N}''_i$ values being transmitted between the Claimant and various Proof Providers and analysing the structure of the messages, the observer would be able to infer which sender is the Claimant and which are Proof Providers. The observer could then check the message headers to discover the MAC addresses of these devices, allowing them to be identified again at a later date. Without this, only the Claimant's messages would be connectable to the device transmitting them. This is because the Proof Provider messages do not employ their digital signature or contain any information pertaining to their identity, unlike those sent by the Claimant.

With identifiers known for each of the participants in the distance bounding exchanges, the observer is left with two similar possible approaches to attempt a location extraction. If the observer is also a participant in the location verification system and has very recently completed a verification of its own, it may be in possession of the locations of some of the devices employed as Proof Providers for the Claimant's current claim. It can then check the IP or MAC addresses within the headers for these devices to see if any of the identifiers matches those of a device employed in the current claim. Alternatively, if the observer

requests a verification of its own, it can check the IP or MAC information of the Proof Providers supplied and use this to identify any devices that were involved in the original Claimant's verification. Once it receives the proof messages from those Proof Providers, it is also in possession of their current location. Using this information and the knowledge that the Claimant is distance bounding with them, the observer can then attempt to extrapolate an area of intersection. If the location information of the Proof Providers is still accurate relative to when they interacted with the Claimant, meaning that the devices involved have not been highly mobile and the observer's claim was very recent, then the Claimant's location should be within this area of intersection, thus compromising location confidentiality.

However, as stated previously, this attack is reliant upon an observer's ability to label the devices involved in the claim. If the observer cannot assign identifiers to the participating Proof Providers, it cannot distinguish between devices and cannot complete the final stage of the attack, where identifiers and locations are combined to provide an area of intersection. Due to the fact that all messages have source and destination information embedded in their structure, in addition to permanent unique identifiers (MAC addresses) the issue of providing anonymity is a major issue in the area of IEEE 802.11 networks. Research is being carried out in the area of disposable identifiers and preventing the tying of one pseudonymous identity to the next to prevent the labelling and tracking of devices as they move through a network [48, 39, 32, 89]. By employing some of these techniques to prevent the labelling of Proof Providers when participating in claims, an observer would no longer be able to extract location information regarding a Claimant, thereby protecting confidentiality of location.

Finally, due to a Claimant relying upon neighbouring devices in order to prove himself at a particular location, the SLVPGP is also vulnerable to another protocol-specific form of attack. In this attack, the Proof Providers involved in a claim behave maliciously and attempt to prevent an honest Claimant from proving his claim, through the provision of false proofs. This form of attack is addressed using device behaviour histories and reputation values in order to detect devices with a pattern of perceived dishonest behaviour. The concept of reputation and its uses by the Verifier within the location verification system are discussed in Chapter 6.

4.2.3 Securing the Exchange

Before designing the SLVPGP, we first analysed the security properties required from such a protocol. After considering both the needs of the users involved in an exchange and those expected to accept final proofs from the system, we identified three key properties required in order to meet their demands. These properties are authentication of origin, confidentiality of location information and anonymity of identity. Therefore, we define a secure and complete protocol for employment within this situation as one which satisfies these three properties.

Authentication of Origin

This location verification system is based upon the idea of a Claimant proving itself to be at a specific location. However, the distributed nature of the system prevents the Verifier from gaining any first hand information regarding the veracity of a location claim. Instead, the Verifier employs other devices in the supposed location of the Claimant to act as Proof Providers. These devices provide the Verifier with a verdict regarding the claim's veracity. As the Verifier is not present at the location, it cannot access these verdicts directly. Instead, it must have them transmitted to it from a remote location. In order for the Verifier to have any confidence in the verdicts being received, it must have some method of verifying their origin. Therefore, we define a secure and complete protocol in this situation to support the property of authentication of origin. This property requires that the origin of any given message must be known and verifiable, and guarantees that any proofs supplied to the Verifier can be traced back to the Proof Provider that created it. The inclusion of this property prevents the Claimant from providing false proofs without detection, as the origins of all messages are known. We further extend this property to also guarantee the detectability of tampering or alterations to any message. This prevents a Claimant or any other device with malicious intent from altering a valid proof message to support its desired outcome.

Confidentiality and Anonymity

In order for a user to trust any location verification system, they must be sure that private data, such as identity and location information, pertaining to their devices is protected. Without this trust, the system would have little value as many users would be unwilling to

participate in claims, either of their own or as Proof Providers. Therefore, we require that for a protocol to be considered secure and complete within this system, it must uphold the security properties of anonymity of identity and confidentiality of location information. We formally define anonymity of identity to mean that a device's identity may not be discoverable by any party other than the Verifier, other than where expressly given. In addition to the use of encryption within the protocol to protect identity information, each device possesses a list of multiple "identities", thus reducing the probability of a device being traceable even if an identity is learned.

Confidentiality of location information is formally defined as the promise that the location of a device with a specific identity may not be discovereable by any party other than where expressly given to it. As distance bounding in a wireless network leaks information regarding locations to those in the vicinity of the exchange [78], protecting the transmission of location information does not guarantee that a device's location will remain unknown at a local level. Therefore, protecting location information at this level does not truly protect the location of a device. However, it is simpler to protect location information throughout the protocol than it is to protect only messages sent beyond the local area. Additionally, should the issue of location leakage through distance bounding be addressed in future research, the protocol itself does not leak any location information.

The formal definitions of anonymity and confidentiality lend themselves to a graduated approach, where each incremental increase in the security of the protocol provides a more rigorous version of protection. As some applications require a lesser amount of security when dealing with this information, we have developed a two-tier definition for both security properties. Where there is a greater need for protection on the part of the users, complete anonymity and confidentiality are employed. Complete anonymity and confidentiality guarantee that no device can learn any identity or location information regarding another device. However, where users do not require such stringent security, external confidentiality and anonymity are employed. When this property is upheld, it guarantees that only those devices participating in a particular exchange are granted access to identity and location information regarding other participants. Neither intruders on the exchange nor devices participating in other exchanges can learn this information, whether or not they are in range of the pertinent transmissions. This partial form of the desired properties

allows the protocol to provide a medium level of security without incurring high secrecy costs (see Section 4.5).

4.2.4 The Role of the Verifier

Unlike the Claimant and its Proof Providers, the Verifier is a special trusted device located outside of the participating devices' mobile networks. It possesses the identities and public keys of all devices within the entire Location Verification system and is trusted to behave honestly with this information. It is charged with deciding on the possibility of a location claim and is given the location information of all devices involved in a location claim. The exact functionality of the Verifier is dealt with fully in Chapter 6. However, in order to understand the protocol's part in our approach to solving the location verification problem, the Verifier's basic processes involved in the protocol are outlined here.

The SLVPGP relies upon the Verifier for two tasks; A) supplying suitable Proof Providers for use during a run of the protocol, and B) determining the final verdict on a claim using the proofs gathered from these Proof Providers. The selection of neighbouring devices for use as Proof Providers is a difficult task. Without a secure process for selecting Proof Providers, the SLVPGP is vulnerable to undetectable collusion attacks which render the entire protocol insecure prior to any distance bounding step occurring. If the Claimant is involved in the selection of Proof Providers, it has the ability to manipulate the process. This allows it to select devices which will participate in a collusion attack, giving it complete control over the claim's final verdict. Therefore, even if the Claimant is many miles from the claimed location, its location claim will always be successful. For this reason, the Claimant is not involved in the Proof Provider selection process. This is discussed more fully in Chapter 6.

The principle task of the Verifier is to assess the possibility of a location claim, based on pertinent information gathered by the Claimant from its Proof Providers. This is acheived using one of two verification approaches, summation-based or trust-based. These approaches are discussed further in Chapter 6.

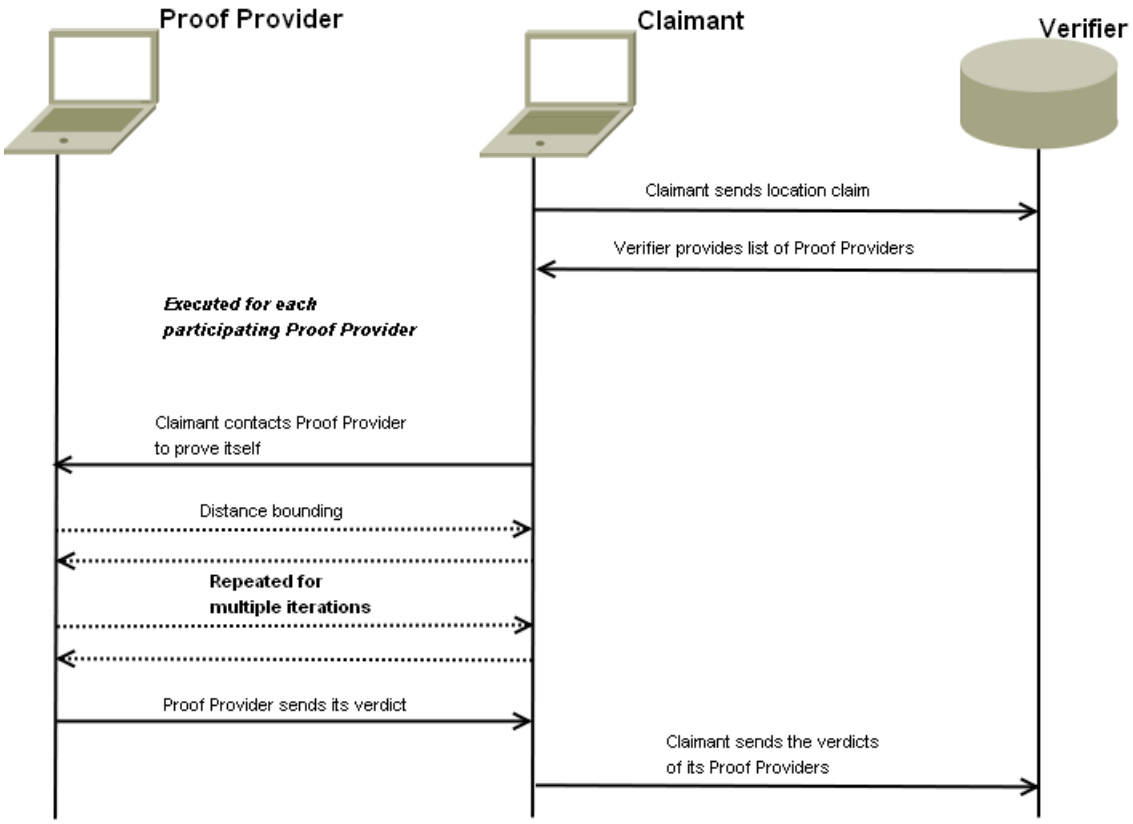


Figure 4.2: The Proof Gathering Process

4.3 Designing the Protocol - the Basic Outline

The basic premise of the location verification is the gathering of proof from neighbouring devices by a Claimant in order to prove itself in that area. This proof either supports or refutes the possibility that a Claimant is indeed within transmission range. The basic process by which this is achieved is shown in Figure 4.2. However, in order for the proofs gathered by the Claimant to be credible, the process must be protected. In this section we outline the steps taken in developing a protocol for the securing of this process.

4.3.1 Protocol Outline

In order to understand what security risks are present in the proof gathering process, we first model a protocol devoid of any security, written in abstract protocol notation (defined at the beginning of this work). This model depicts the framework within which devices participating in a claim provide evidence of a Claimant's presence in their vicinity for the Verifier. We outline the protocol as a sequence of steps taken by the parties involved in the exchange: the Verifier (V), the Claimant (C) and N Proof Providers (B_i where i

$\in\{1...N\}$). Up to M iterations of distance bounding are completed, with k used to index through them ($k \in\{1...M\}$).

1. C→V: C, X_C

   The Claimant sends a message to the Verifier containing its identity (C) and location (X_C), requesting that its location claim be verified.

2. V→C: $B_1, B_2, ...B_N$

   The Verifier sends a list of devices in the claimed area to the Claimant for use as Proof Providers.

3. C→ B_i: $C, B_i, \mathcal{N}_i, \mathcal{N}_i^{'}$

   The Claimant broadcasts a message for each Proof Provider containing their identity and two nonces (long random integers), $\mathcal{N}_i$ (the initiating nonce) and $\mathcal{N}_i^{'}$ (the replying nonce).

4. The Claimant and Proof Providers create two chains of M hashes, one for each of the received nonces. This is explained in detail in Section 4.3.2. The hashes are noted as $H_{i,k}$ (the initiating chain) and $H_{i,k}^{'}$ (the replying chain) respectively, where $i \in \{1...N\}$ and $k \in \{1...M\}$.

5. Distance Bounding

   This stage is performed multiple times to lessen the effect of any network issues which may arise.

   (a) B_i starts its timer.

   (b) B_i →C: k, $H_{i,k}$, $\mathcal{N}_i^{''}$

   B_i sends a message to the Claimant containing a new random nonce $\mathcal{N}_i^{''}$, a randomly selected value from the initiating hash chain, $H_{i,k}$, and its position in the chain, k. k decreases with each iteration of distance bounding. The Claimant checks whether the k^{th} value in the $\mathcal{N}_i$ hash chain matches $H_{i,k}$ and if so, continues to the next step.

   (c) C→ B_i: $H_{i,k}^{'}$, $\mathcal{N}_i^{''}$

   The Claimant sends a message to B_i containing $\mathcal{N}_i^{''}$and the k^{th} value in the replying hash chain. B_i compares this with the k^{th} value in its replying hash

chain. If the two values match and the received value of $\mathcal{N}_i''$ matches its own sent value, B_i stops its timer and calculates the round trip latency (subtracting out its own internal processing time).

6. $B_i \rightarrow$C: B_i, T_i, X_i, L_i

   B_i sends the Claimant its proof, comprised of its identity, its current location (X_i) and overall decision regarding the presence of the Claimant in its vicinity (L_i). B_i also includes a timestamp in the proof to tie the location to this specific point in time.

7. C$\rightarrow$V: C, T_C, X_C,

   B_1, T_1, X_1, L_1,

   ...,

   B_N, T_N, X_N, L_N

   The Claimant compiles all the proofs gathered from its Proof Providers and forwards them to the Verifier, along with its identity C, the current location X_C and a timestamp T_C, tying the proofs to that point in time.

8. The Verifier combines the evidence gathered by the Claimant and extracts a final verdict (D_V) on the validity of the location claim.

9. V$\rightarrow$C: D_V, X_C, T_V

   The Verifier sends its final verdict to the Claimant for use as proof of its presence in the claimed location (X_C). Additionally, V includes a timestamp to tie the location proof to this specific point in time. This proof can be supplied to other entities and systems as verified evidence of the Claimant's presence in the stated area at the stated time.

4.3.2 Protocol Discussion

The simplistic protocol depicted above represents the basic design for a proof gathering protection protocol. It defines a method to allow a Claimant to submit a location claim to the Verifier, indicating its interest in having its location verified in a provable fashion. The protocol goes on to define a framework for the gathering of proof from a pre-approved list of neighbouring devices and specifies the components required within a valid proof message.

Additionally, the protocol defines distance bounding as the method by which a Claimant proves itself to be within contact range of a Proof Provider and specifies a framework for the exchange.

Due to the possibility of network issues such as delays and lost packets, the result of a location claim could be affected through the true results of a single distance bounding exchange being distorted. To compensate for this possibility, multiple distance bounding exchanges have been included within the protocol's design. In order to avoid increases in data overheads, a hash chain [42] has been employed to calculate multiple nonces from a single initial nonce. This chain is constructed as follows: The initial nonce is hashed using the hash function $\mathcal{H}(\mathcal{N})$, producing the value H_1. This value is then hashed using the same function ($\mathcal{H}(H_1)$) to produce H_2. Therefore, the k^{th} value H_k in a hash chain is calculated from $\mathcal{N}$ using the formula $H_k = \mathcal{H}^k(\mathcal{N})$. New values are created with this formula until a sufficient number of entries have been created within the chain. However, one danger of creating chains in this manner is that if given the value H_k an intruder could then calculate all values above H_k in the chain. In order to address this, the value of k should decrease with each iteration of distance bounding, moving downwards through the chain. Although this reduces the number of useable values within the hash chain, it removes the risk of a security leak.

Two chains are employed in the protocol, one initialised using $\mathcal{N}_i$ and the other using $\mathcal{N}_i^{'}$. These are used within the distance bounding exchange as a method of ensuring that both devices involved, i.e. the Claimant and Proof Provider B_i can confirm that they are interacting with their expected exchange partner. The hash chain values generated from the initiating nonce, $\mathcal{N}_i$, prove to the Claimant that the message originated from a Proof Provider engaged in the current location claim. The values from the hash chain derived from the replying nonce, $\mathcal{N}_i^{'}$, are used by the Proof Provider to verify that the message received originated from the Claimant involved in that specific location claim.

In order to reduce the probability of a Claimant successfully guessing the approprite response from the chain and transmitting its response before receiving the Proof Provider's challenge, an echoing nonce $\mathcal{N}_i^{''}$ is included. As the Claimant's reply message contains both a nonce from its reply chain and the echoing nonce, it is forced to wait until it receives the value of $\mathcal{N}_i^{''}$. This dramatically reduces the possibility of an early transmission attack

successfully being accepted as a valid response, as the probability of successfully guessing the value of $\mathcal{N}_i''$ is very low. The use of a randomly selected value from the hash chain ($H_{i,k}$, the k^{th} value in the hash chain created from $\mathcal{N}_i$) adds to the unpredictability of the Proof Provider's distance bounding challenge message for the Claimant.

While the simpler and more intuitively correct approach to communicating the evidence regarding the claim from the Proof Providers to the Verifier is to have the Proof Providers transmit their proofs directly to the Verifier, we have chosen to employ a different tack. In our approach, the Claimant gathers the final proofs from its Proof Providers and forwards them to the Verifier along with its final message indicating the completion of the evidence gathering process. Although this approach to the supplying of proofs does add complexity to the protocol, it also protects the Claimant from a scenario where the Proof Providers participate in distance bounding, but do not transmit their proof messages to the Verifier. In this situation, if the Claimant was to supply its final message to the Verifier, who had received no supporting proof messages, it would assume that the claim was false and decrement the Claimant's honesty value as punishment for submitting a false location claim. By transmitting the proof messages via the Claimant, the Claimant is not deceived into believing the Verifier has received evidence regarding its claim, only for that evidence never to materialise.

In order to ensure that the proofs supplied by the Proof Providers and the message containing the Verifier's final verdict are unique and cannot be employed in a replay attack, we have included timestamps within these messages. These timestamps can be generated based on the reasonably accurate clock within the tamper resistant unit, thus providing a loose level of syncronization. Upon initialisation of the claim, an acceptable window is defined by the Verifier, and any messages timestamped within this window are deemed applicable to this claim. As the Verifier registers the time of initialisation, any messages with a timestamp preceeding initiation of the claim can be detected and discarded, thus preventing reuse of older proofs. However, as any proofs timestamped within the window may be accepted, a highly mobile device (such as a vehicle within a VANET) may significantly change its location over the course of the window. This means that a device may no longer be in the vicinity of the location being claimed by the time the claim is verified. If a Claimant was to initiate two claims, one a short period of time after the other, it

may be provided with a similar list of Proof Providers and could, in theory, provide proofs gathered based on the first claim's list. This is partially protected against through the ability of the Verifier to discard proofs dated prior to the initiation time of a claim, though an overlap is possible. An alternative approach to this issue is to employ an identifying nonce, which identifies the claim a proof corresponds to. However, if nonces were employed without timestamps, a device could gather proofs from its Proof Providers at one time. It could then hold these proofs for a period of time and then supply them to the Verifier, thus generating a location verification containing its earlier location, timestamped with a current time. Therefore, even where nonces are employed, an acceptance window must also be used.

As the protocol is currently presented, the security of both the final proofs and the Verifier's verdict is compromised. Both the distance bounding exchange and the transmitted proof messages are susceptible to tampering, either by an intruder or a malicious Claimant which wishes to alter the verdict given by a Proof Provider or the Verifier's overall verdict. In addition to this, fraudulent proofs can be fabricated by a malicious Claimant without making contact with a Proof Provider. Finally, the overall protocol is vulnerable to the mafia and terrorist frauds as there is no method of tying the messages sent during the distance bounding portion of the protocol to the Claimant. In order to address these vulnerabilities, we have repeatedly extended this basic protocol design, increasing security with each extension. In the following section we discuss these extensions, their improvements to security and the costs incurred in enacting these.

4.4 Extending the Protocol

In the previous section we defined a basic approach for the protection of the proof gathering process. We designed a simplistic protocol to enact this protection and discussed its functionality and flaws. However, this simplistic approach is undeniably insecure and upholds none of the security properties outlined in Section 4.2.3 which define a secure and complete protocol in this situation. For this reason, we extend the basic protocol using security techniques such as the employment of encryption and digital signatures to produce a secure and complete protocol which fulfills the requirements of the system.

The extension of the protocol has been split into three distinct iterations. In exten-

sion one, we amend the basic protocol to support authentication of origin, thus preventing false and amended proofs from being accepted. In extension two, we increase the level of security provided by extension one to include external confidentiality and anonymity, thus protecting the private data of devices involved from intruders. The addition of these properties satisfies the definition of a secure and complete protocol within this system, creating the first complete version of the SLVPGP. Finally in the third extension, we increase the level of confidentiality and anonymity provided from external to complete. As with the second extension, extension three satisfies the definition of a secure and complete protocol within this system, creating a second secure and complete proof gathering protocol. Each of the extensions outlined are discussed in depth in this section.

As before, we outline a protocol as a series of steps taken by the Claimant (C), Proof Providers (B_i where $i \in \{1...N\}$) and Verifier (V). We assume that all parties involved have asymmetric key pairs associated with them. These key pairs are noted as K_A^- and K_A^+ for the private and public keys respectively, where A is the owning party's identity. We also assume that the identity each party is referred to by is its public key. This means that if a device has knowledge of another's identity, it can encrypt messages for their decryption.

4.4.1 The SLVPGP: Extension One

In this extension, we introduce the first of our desired security properties: authentication of origin. This property ensures that the Verifier can trace the origin of any message received back to its source, thus removing the danger of false proofs. However, this extension does not provide a secure and complete protocol as the properties of confidentiality and anonymity are not supported. This extension has been designed as a lightweight version of the protocol, to function in situations where confidentiality and anonymity are not required or highly desired security properties for a system.

1. C→V: $\{|C, X_C|\}_{K_C^-}$
2. V→C: $\{|B_1, B_2, ..., B_N|\}_{K_V^-}$
3. C→ B_i: $\left\{|B_i, C, \left\{\mathcal{N}_i, \mathcal{N}'_i\right\}_{K_{B_i}^+}|\right\}_{K_C^-}$
4. The Claimant and Proof Providers create two chains of M hashes, one for each of the received nonces.

5. Distance Bounding (executed multiple times)

   (a) B_i starts its timer.

   (b) $B_i \rightarrow$C: k, $H_{i,k}$,$\mathcal{N}_i''$

   (c) C$\rightarrow B_i$: $H_{i,k}'$, $\left\{|\mathcal{N}_i''|\right\}_{K_C^-}$

6. $B_i \rightarrow$C: $\{|B_i, T_i, X_i, L_i, C|\}_{K_{B_i}^-}$

7. C$\rightarrow$V: $\{|C, T_C, X_C|\}_{K_C^-}$
   $\{|\{|B_1, T_1, X_1, L_1, C|\}_{K_{B_1}^-}$,
   ...,
   $\{|B_N, T_N, X_N, L_N, C|\}_{K_{B_N}^-}|\}_{K_C^-}$

8. V$\rightarrow$C: $\{|D_V, X_C, C, T_V|\}_{K_V^-}$

This extension to the protocol removes many of the vulnerabilities identified within the basic design. The addition of encryption to the initiating and replying nonces, $\mathcal{N}_i$ and $\mathcal{N}_i'$, prevents intruding devices from gaining knowledge of these values without the collusion of malicious involved parties. Previously, these values were sent in cleartext, negating the reasoning behind their inclusion. With these values sent in an encrypted message, only legitimate participants and devices colluding with malicious participants have access to them for use within the distance bounding process, thus providing authentication. However, even if a colluding device was granted access to these values, the danger of a terrorist fraud attack going undetected is negated through the addition of a digital signature on the echoing nonce. By tying the Claimant device directly to the distance bounding process, the protocol's vulnerability to proxying attempts is removed (see Chapter 3).

An alternative approach to authentication during distance bounding was also considered, in which a symmetric key is established between the Claimant and that Proof Provider for use on the return leg of the exchange. Employing a shared key would reduce the complexity of the protocol in further extensions and decrease the time required to complete distance bounding. However, this option is only viable if such a key is held within the tamper-resistant unit. Without this protection, a malicious Claimant would be free to distribute this key to any colluding devices, thus removing the element of authentication and reinstating the threat of undetectable collusion attacks. This issue can be negated

through the use of a key-exchange protocol such as the Diffie-Hellman key exchange [23], which runs through the tamper resistant module, preventing the resulting key from being supplied to other devices. If this approach is taken, the time requirements for distance bounding would be significantly reduced, due to the vast difference in computation time required for symmetric key cryptography in comparison to asymmetric schemes, particularly RSA. Additionally, the amount of data to be transmitted would decrease, as the key size required for symmetric cryptography schemes is significantly less than that required for comparable security when using asymmetric schemes.

The use of digital signatures to provide authentication of origin also addresses other security concerns within the protocol. A malicious Claimant can no longer fabricate proof for the Verifier, as it cannot generate the Proof Provider's signature to match the proof. Nor can it undetectably alter the content of an existing proof from a Proof Provider, as the signed hash of the message would no longer match the message being transmitted. Similarly, neither the Claimant nor a malicious intruder can alter the Verifier's overall verdict in the final verification message. Additionally, the inclusion of the Claimant's identity within the signed proof messages from the Proof Providers and the final verification message prevents a malicious Claimant from using messages pertaining to another Claimant as evidence of its own claim. Finally, an intruder cannot participate in the exchange as its signature key would not produce a valid message signature, nor can it intercept the final message and undetectably insert false proofs.

4.4.2 The SLVPGP: Extension Two

In the previous section, we extended the protocol to provide authentication of origin and prevent attacks such as the mafia and terrorist frauds. However, during the execution of an SLVPGP extension one exchange, an intruder to the system can read all transmissions in the exchange. Therefore, the properties of confidentiality and anonymity are still not upheld. In this section we further extend the protocol to provide a form of anonymity and confidentiality. We achieve this through the addition of encryption.

The functionality of the extension two protocol remains as in extension one, with the Claimant requesting that the Verifier allow it to prove its location using proof from neighbouring devices (Step 1). The Verifier again provides a list of devices it believes to

be in the vicinity of the Claimant for use as Proof Providers (Step 2), and the Claimant contacts each of these, using distance bounding to establish that it is within that device's range (Steps 3-5). When the Proof Providers employed have decided on the veracity of the Claimant's claim that it is within their range, they supply the Claimant with proof messages (Step 6). The Claimant then provides these to the Verifier for use in the final decision (Step 7) and finally, the Verifier supplies the Claimant with its judgement on the veracity of its location claim (Step 8). However, in this version of the protocol, the contents of all messages (excluding those sent during the distance bounding stage) are encrypted for the eyes of their recipient. This prevents those without the appropriate key from gaining access to the private values contained within, more specifically the identities and locations of the participants in the protocol.

1. C→V: $\left\{\{|C, X_C|\}_{K_C^-}\right\}_{K_V^+}$
2. V→C: $\left\{\{|B_1, B_2, ..., B_N|\}_{K_V^-}\right\}_{K_C^+}$
3. C→ B_i: $\left\{\left\{|B_i, C, \mathcal{N}_i, \mathcal{N}_i'|\right\}_{K_C^-}\right\}_{K_{B_i}^+}$
4. The Claimant and Proof Providers create two chains of M hashes, one for each of the received nonces.
5. Distance Bounding (executed multiple times)
   (a) B_i starts its timer.
   (b) B_i →C: k, $H_{i,k}$,$\mathcal{N}_i''$
   (c) C→ B_i: $H_{i,k}'$, $\left\{|\mathcal{N}_i''|\right\}_{K_C^-}$
6. B_i →C: $\left\{\{|B_i, T_i, X_i, L_i, C|\}_{K_{B_i}^-}\right\}_{K_C^+}$
7. C→V: $\left\{\{|C, T_C, X_C|\}_{K_C^-}\right\}_{K_V^+}$,
   $\{\{|\,\{|B_1, T_1, X_1, L_1, C|\}_{K_{B_1}^-}$,
   ...,
   $\{|B_N, T_N, X_N, L_N, C|\}_{K_{B_N}^-}\,|\}_{K_C^-}\}_{K_V^+}$
8. V→C: $\left\{\{|D_V, X_C, C, T_V|\}_{K_V^-}\right\}_{K_C^+}$

While the first extension to the protocol added authentication through the use of digital signatures, the addition of encryption in this extension builds upon the existing framework to include external anonymity and confidentiality. Upholding these properties protects the transmitted identity and location information of all parties involved from any device external to the exchange. While the Claimant and any Proof Providers involved have access to the location and identity information being sent over the run of the protocol, no other devices can discover it, whether they are within range of the transmissions or not.

However, as discussed previously, location information is leaked when a device participates in distance bounding. Therefore, though the transmitted location information is protected, local observers within the network can still extract the location of devices in their vicinity. Despite this leakage, the location information transmitted in steps five and six has also been protected through encryption, as doing so provides a simpler message structure. Additionally, the protocol structure remains fit for employment should the issue of location leakage be resolved.

As mentioned in the previous section, an alternative protocol structure may be built in which a symmetric key is agreed upon between the Claimant and each Proof Provider. This key may be agreed upon during the fourth step of the protocol. It could then be employed during the distance bounding process to decrease speed and data transmission costs, as well as to encrypt the Proof Provier's final proof message. However, even if this approach is employed, the Proof Provider's digital signature must remain present on the proof message in order to provide authentication of origin to the Verifier.

4.4.3 The SLVPGP: Extension Three

The previous extension to the SLVPGP increased the level of security surrounding the proof gathering process. However although a form of anonymity and confidentiality exist, complete anonymity and confidentiality have not yet been achieved. Devices participating within the proof gathering exchange could record information on the parties involved while they have access and then use this information maliciously at some point in the future. For example, if a malicious observer is also a participant within the system and is provided with information on other devices, including their identities, it can retain this information for use at a later date. With the Claimant in an exchange identified through the observational

analysis of traffic and message structure during the distance bounding stage, the observer can then attempt to solve the equation:

$$\mathcal{N}_i'' = \left\{ \left\{ |\mathcal{N}_i''| \right\}_{K_C^-} \right\}_x$$

by substituting x with each of the identities it has acquired over its time within the system. Assuming the observer is in possession of the correct identity within its list of previously encountered devices, both the identity and location of the Claimant would be compromised. Although the identities of devices are frequently changing pseudonyms, it should be noted that there is a possibility that the identity in use by the Claimant is already known by the observer. For this reason, we extend the protocol once more to provide complete anonymity and confidentiality from all devices, even if they are involved in the exchange. This security increase causes the protocol's complexity to increase greatly, which is discussed in Section 4.5.

1. C→V: $\left\{ \{|C, X_C|\}_{K_C^-} \right\}_{K_V^+}$
2. V→C:
   $\left\{ |\left\{ \left\{ |B_1, \mathcal{N}_1, \mathcal{N}_1', K_{CB_1}| \right\}_{K_V^-} \right\}_{K_{B_1}^+}, ..., \left\{ \left\{ |B_N, \mathcal{N}_N, \mathcal{N}_N', K_{CB_N}| \right\}_{K_V^-} \right\}_{K_{B_N}^+} | \right\}_{K_V^-}$,
   $\left\{ \left\{ |\mathcal{N}_1, \mathcal{N}_1', K_{CB_1}, ..., \mathcal{N}_N, \mathcal{N}_N', K_{CB_N}| \right\}_{K_V^-} \right\}_{K_C^+}$
3. The Claimant decrypts and stores each of the nonces received from the Verifier.
4. C→ B_i: $\left\{ \left\{ |B_i, \mathcal{N}_i, \mathcal{N}_i', K_{CB_i}| \right\}_{K_V^-} \right\}_{K_{B_i}^+}, \{C\}_{K_V^+}$
5. The Claimant and Proof Providers create two chains of M hashes, one for $\mathcal{N}_i$ and one for $\mathcal{N}_i'$.
6. Distance Bounding (executed multiple times)
   (a) B_i starts its timer.
   (b) B_i →C: k, $H_{i,k}$,$\mathcal{N}_i''$
   (c) C→ B_i: $H_{i,k}'$, $\left\{ |\mathcal{N}_i''| \right\}_{K_C^-}$

7. $B_i \rightarrow$C: $\mathcal{H}(K_{CB_i}), \left\{ \mathcal{N}_i, \left\{ \left\{ |B_i, \left\{ |\mathcal{N}_i''| \right\}_{K_C^-}, T_i, X_i, L_i, \{C\}_{K_V^+} | \right\}_{K_{B_i}^-} \right\}_{K_V^+} \right\}_{K_{CB_i}}$

8. C→V: $\left\{ \{|C, X_C, T_C|\}_{K_C^-} \right\}_{K_V^+}$,
$\{| \left\{ \left\{ |B_1, \left\{ |\mathcal{N}_1''| \right\}_{K_C^-}, T_1, X_1, L_1, \{C\}_{K_V^+} | \right\}_{K_{B_1}^-} \right\}_{K_V^+}$,
...,
$\left\{ \left\{ |B_N, \left\{ |\mathcal{N}_N''| \right\}_{K_C^-}, T_N, X_N, L_N, \{C\}_{K_V^+} | \right\}_{K_{B_N}^-} \right\}_{K_V^+} |\}_{K_C^-}$

9. V→C: $\left\{ \{|D_V, X_C, C, T_V|\}_{K_V^-} \right\}_{K_C^+}$

With the extension of the protocol to provide complete anonymity and confidentiality, a new security issue emerges. Due to the presence of encryption, the Claimant is unable to differentiate between valid and fraudulent proof messages. In this situation an intruder could launch a denial of service attack, supplying the Claimant with fraudulent proofs for inclusion as evidence for the Verifier. The Verifier would fail the claim for including proofs from illegitimate sources, as it cannot distinguish whether or not the fraudulent proofs were intentionally included by the Claimant in an attempt to deceive the system. This dismissal renders the efforts of those involved devices wasted. Therefore, the Claimant requires a method of matching a received proof to one of its unknown Proof Provider. For this reason, the Verifier generates a symmetric key, K_{CB_i}, for each Proof Provider involved in the claim. The Proof Provider creates a message containing its proof and the original nonce $\mathcal{N}_i$ (supplied by the Verifier and known only by the Claimant, Verifier and that Proof Provider). It encrypts this message with the symmetric key and transmits it to the Claimant, accompanied by a hash of the key. The hashed value allows the Claimant to identify which key should be used to decrypt the message. If the Claimant can successfully decrypt the message, that proof is confirmed as being created by a legitimate Proof Provider. This removes an intruder's ability to undetectably insert illegitimate proofs into the exchange. However, the message being decrypted is a piece of ciphertext. Without the presence of a known value in the message, the Claimant would be unable to differenciate between a message sent by a legitimate Proof Provider and one supplied by a malicious intruder. For this reason, the initiating nonce $\mathcal{N}_i$ is included in the encryption. Upon decryption using its symmetric key, the Claimant can immediately identify that nonce and

confirm that it matches the key employed to decrypt the message. This re-confirms the veracity of the received proof.

The functionality of K_{CB_i}, much like that of $\mathcal{N}_i$ and $\mathcal{N}_i'$, is tied to the fact that no devices other than the Claimant and a Proof Provider have knowledge of its value. In order to prevent intruding devices from learning the value of any of these nonces, the messages containing them must be encrypted for each recipient. As a device's identity is its public key, the Claimant is no longer capable of encrypting messages for its Proof Providers. If the Claimant was capable of this, complete anonymity and confidentiality would not hold, rendering the protocol no longer complete. Therefore, a new approach is required to maintain complete anonymity and confidentiality whilst still enabling the encryption of these values to protect them from eavesdropping attackers.

In order to fulfill these requirements, we have expanded the role of the Verifier within the protocol. In earlier extensions, the Verifier's role consisted merely of transmitting a list of the Proof Providers for use in the exchange. However, in this extension the Verifier assumes much of the initialisation role of the Claimant, generating the nonces for use within the exchange and encrypting them for the participating Proof Providers. As the Claimant also requires a copy of these values, the Verifier is forced to create and encrypt two messages per Proof Provider, one encrypted with the Claimant's public key and one encrypted with that of the Proof Provider's. With these messages pre-encrypted, the Claimant need then only broadcast them to all devices in its vicinity. It is through the implementation of this approach, where the Claimant is forced to broadcast messages without knowing the identity of their intended recipient, that complete anonymity and confidentiality is achieved. However, the satisfaction of these properties also precludes the participating Proof Providers from successfully verifying the Claimant's digital signatures during the distance bounding exchange. If unaddressed, this inability would render the protocol vulnerable to collusion attacks, as the signature employed on the return leg of the exchange could be undetectably falsified by any device. This negates any protection offered by its inclusion. In order to address this potential vulnerability, a randomly selected digitally signed nonce from the distance bounding exchange is included by the Proof Provider in its final proof message. This signed nonce is forwarded to the Verifier as part of the proof message, where the Verifier can confirm the signature's authenticity as it possesses the

appropriate public key. As the nonce is selected from the array sent during the exchange at random, the Claimant cannot predict which run should be participated in correctly. If it wishes to attempt a collusion attack and participate honestly in only a single run, it has at best a one in x chance of successfully cheating the system (where x = the number of distance bounding exchanges performed).

With the satisfaction of external confidentiality and anonymity, the inclusion of a renewed protection against collusion attacks and the continued satisfaction of authentication of origin, we believe that we have designed a second secure and complete protocol which satisfies the three security properties outlined in Section 4.2.3 to the fullest degree. Unfortunately, the creation and transmission of two initialisation messages for each Proof Provider involved, the inclusion of a signed nonce in the proof messages and the heavy use of encryption causes this protocol extention to come at a high overhead cost. This cost explosion is discussed in detail in Section 4.5.3. However, the cost explosion may be lessened through the further use of symmetric keys. As in the case of extensions one and two of the SLVPGP, the use of symmetric keys between the Claimant and Proof Providers when doing distance bounding would decrease both the transmission and time costs required to complete the stage, as symmetric cryptographic operations are far faster and result in smaller pieces of cipher text than asymmetric cryptography. This has the additional benefit of removing the need for a randomly selected signed nonce to be included in the proof message sent to the Verifier, as the Proof Providers would again be able to confirm that the value was encrypted correctly. The removal of this section of the proof message also decreases the amount of data to be sent and time required for sending, thereby reducing the cost explosion further.

However, though employing symmetric cryptography on this stage would decrease both the time and data costs on the part of the Claimant, the costs incurred by the Proof Provider would actually increase. In the current approach, the Proof Provider does not incur any verification costs, it does not possess the appropriate key. Therefore, there is a trade off between the Claimant's savings and the increased costs incurred by the Proof Provider. Future work on this protocol should investigate the benefits vs costs of switching to this approach. Additionally, the optimization possibilities of redesigning the extension overall and completely incorporating symmetric keys into the structure should

be addressed, as this may lessen the complexity of the protocol in addition to decreasing the costs incurred.

4.5 Analysing the Protocol

In this section we discuss the various extensions to the SLVPGP and their associated overhead costs. Each extension to the SLVPGP increases the security provided to those participating, however in order to provide the increased security, the extensions require additional data to be transmitted and computing time to generate encryptions or digital signatures. We define the term *costs* to mean this additional data and/or time required in order to increase the level of security provided. In this section we discuss the costs incurred within each extension, in order to provide that extension's level of security (Sections 4.5.1-4.5.3). We then discuss the practical costs incurred by each extension, broken down by cost type (Section 4.5.4).

4.5.1 Extension One

In the first extension of the protocol, support is added for authentication of origin. This property is supplied through the application of digital signatures to messages within the protocol. However, the employment of digital signatures causes an increase in the costs associated with the protocol. As the digital signature for each message must be computed and verified during the protocol's run, the length of time taken to complete each step increases. In addition to this, the digital signature for each message must be transmitted with that message, increasing the amount of data required to be transmitted. These increased overhead costs apply to all messages including a digital signature.

In addition to the overhead costs associated with providing authentication, the first extension to the SLVPGP increases the level of security within the protocol through the prevention of collusion attacks such as the terrorist fraud. This is achieved by tying the distance bounding stage of the protocol to the Claimant through employing a digital signature on the echoing nonce. An improvement in costs may be made here, through the employment of symmetric cryptography instead of the currently employed asymmetric approach. This would reduce both the time required for encryption and the size of the resulting message. However, if the protocol's design is amended to incorporate this approach, an

additional step must be included within the protocol, during which a key agreement scheme (such as Diffie-Hellman [23]) is run. Though this does increase the costs incurred, the extra amount of data being transmitted and time required is negated by the cost savings per distance bounding iteration.

The security of the protocol is also improved upon by including the Claimant's identity within the proof message. This incurs only a slight increase in data overheads, to tie the proof to the Claimant involve and ensures that the protocol is not vulnerable to attack from a dishonest Claimant wishing to use proofs concerning other Claimants within its own verification. As this message is digitally signed by the Proof Provider supplying it, the Claimant is unable to undetectably remove this piece of evidence. This approach is also employed in the final message from the Verifier to the Claimant, tying the verification to that Claimant. Again, as the message is digitally signed by the entity providing it, the Claimant is unable to undetectably alter the identity included. This prevents it from altering verification messages pertaining to other devices to suit its own needs.

However, although the first extension does protect the integrity of the information transmitted, the lack of encryption and its associated overhead increases means that this extension does not afford the devices involved either confidentiality or anonymity. These properties are supplied in varying degrees within the second and third extensions.

4.5.2 Extension Two

The second extension to the SLVPGP increases the security properties supported to include external anonymity and external confidentiality. In addition to authentication of origin, these properties are achieved through the application of encryption to communications between the various parties. Encrypting these messages ensures the contents are known only to those with the appropriate decryption keys and those with whom the key owners share the contents. This newfound protection provides external anonymity and external confidentiality - i.e. no device outside the exchange can learn identities or location information without aid from a dishonest party participating in the exchange.

However, encryption greatly increases the transmission costs associated with the protocol. Whilst digital signatures require only the digest of a message to be encrypted, encrypting an entire message increases the amount of data required to be transmitted im-

mensely. This increase not only increases the data transmission costs incurred, but also causes the amount of time required to transmit the message to increase. These increases in costs are incurred many times over. In addition to this, the cost in time overheads also increases as the encryption of each message must be computed, along with its corresponding decryption. Overall, the application of encryption to the protocol is very costly, with large increases incurred in both computation time and data to be transmitted. However, the security purchased at this cost ensures a greatly increased level of all devices participating, providing a degree of protection to both identity and location information.

Finally, the amendment of the protocol's design suggested in Section 4.5.1, where symmetric cryptography replaces digital signatures during distance bounding, would significantly decrease the costs incurred by that particular stage of the protocol. Additionally, the savings can be increased by the employment of their symmetric key by each Proof Provider to encrypt their proof message, although (as mentioned in Section 4.4.2) the proof must remain digitally signed in order to retain the property of authentication of origin.

4.5.3 Extension Three

The final extension to the SLVPGP incurs the greatest increase in transmission overheads in an effort to provide both complete anonymity and confidentiality. The provision of these security properties provides an increased level of protection and prevents a wolf in sheep's clothing attack. In this scenario, a device participates (either honestly or dishonestly) in a verification exchange as either a Claimant or Proof Provider, but stores the information it receives for use at a later date. As the device is participating within the verification process, it is granted access to the location information and identities of the devices involved. Extension three of the SLVPGP prevents this through the protection of all information from even those devices participating in the exchange, excluding the Verifier.

However, increasing the protection provided by the protocol in this manner introduces new complications, as the Claimant requires the ability to share secrets with each of the involved Proof Providers during its claim. This is done to enable identification during the distance bounding portion of the protocol and to ensure that false proofs cannot be inserted into the exchange by malicious devices. As the Claimant does not know the identities or public keys of its Proof Providers, it can no longer securely provide them with

these secrets. However the Verifier, as a trusted entity, has knowledge of all public keys within the system. It is employed to create the secrets shared by a Claimant and its Proof Providers, encrypting them to secure them from intruders. This process increases the costs associated with this extension, as for each Proof Provider involved, the Verifier is required to create, encrypt and transmit multiple nonces and a shared AES key for both that Proof Provider and the Claimant.

In addition to this overhead increase, the proof message created by the Proof Providers and forwarded by the Claimant increases in size. Each Proof Provider includes a randomly selected digitally signed echoing nonce from its distance bounding exchange. As the Proof Providers can no longer verify the validity of the received digital signatures, they enclose a sample from the process, allowing the Verifier to validate the proofs. This increases the costs involved not only in the proof message received by the Claimant from each of its Proof Providers, but also in the final proof sent from the Claimant to the Verifier. However, the inability of the Proof Providers to verify the digital signatures including during distance bounding causes a substantial decrease in costs, as shown in Figure 4.5. Section 4.4.3 discusses the possibility of employing symmetric encryption on the distance bounding stage of the protocol as a cost-reduction mechanism, but the issue of whether this will decrease or increase the costs incurred remains open for future investigation.

It should be noted that while Figure 4.5 and Tables 4.8, 4.9 & 4.10 reflect the verification costs for extension three of the SLVPGP, these are the definite costs incurred. They do not include any additional possible costs, such as the number of extra verifications to be completed due to the employment of broadcasting and the inability of a device to be sure it is not the intended recipient of a message. This aspect is not included in the computations as it is impossible to know how many additional verification operations are required for a single run, even when operating in a vacuum. This is due to the unpredictability of the network and its topology, as the location of devices continuously changes and not all Proof Providers will necessarily be in range of each other. Therefore, while the verification costs incurred for extension three appear lower than those for extension two (Figure 4.5), these are a minimum and this relationship cannot be guaranteed.

	Unit size	Speed of cryptographic operation
SHA-256	1 byte	0.000008584 milliseconds
AES	1 128 bit block	0.00014 milliseconds
RSA encryption	1 1024 bit block	0.08 milliseconds
RSA decryption	1 1024 bit block	1.46 milliseconds

Table 4.1: Benchmark Speeds Used in Practical Cost Computation

Component	Size (in bits)
Nonce ($\mathcal{N}$)	64
Timestamp (T)	32
Location (X)	64
Proof/Verdict (L/D_V)	1
Identity	1024
Hash (h)	256
k	16

Table 4.2: Message Component Sizes

4.5.4 Extension Comparison

In order to fully analyse each extension within the SLVPGP and to illustrate the comparative costs vs. the level of security provided, we have computed the practical costs incurred for a single run of each protocol, using 1024 bit RSA [82] keys. These costs have been divided into three categories; *transmission* (Figure 4.3), *generation* (Figure 4.4) and *verification* (Figure 4.5). In the case of *verification*, we focus on the definite costs incurred, i.e. those that will definitely be incurred by the participants. The additional costs incurred by the use of broadcasting in extension three (mentioned earlier in Section 4.5.3) are discussed at the end of the section, but are not included in the point by point comparison due to their unpredictability. Also not included in this section are cost comparisons for alternative protocol designs using symmetric cryptography instead of asymmetric approaches, though these may be addressed in future work.

For each cost category, a graph is included depicting the incurred cost increases for each extension. The breakdown of costs in each graph can be found in Tables 4.3, 4.4, 4.6 and 4.8, based on the component figures shown in Table 4.2.

For the purpose of comparing the cost of each extension, the protocol is run with the Claimant, Verifier and five Proof Providers. Each Proof Provider completes ten distance bounding iterations with the Claimant. We employ benchmark speeds from Crypto++'s

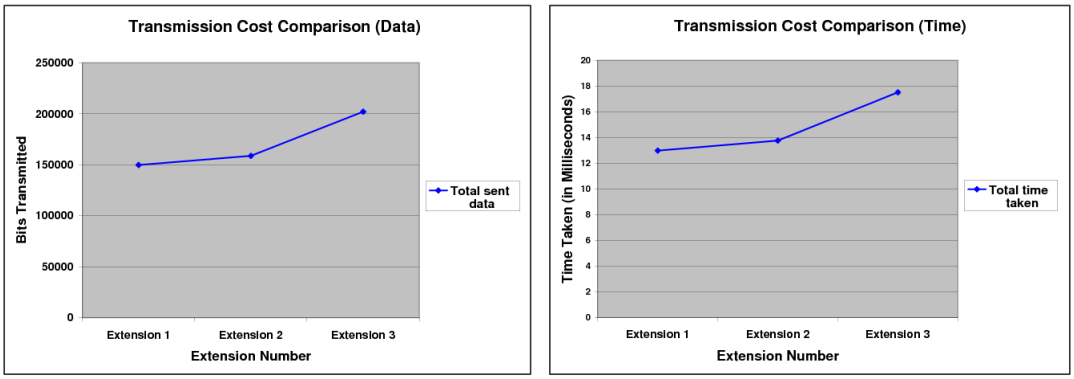


Figure 4.3: Transmission Cost Comparison - Data and Time Costs

benchmarking page[1] to calculate the time required to complete an encryption, decryption or SHA-256 operation. These speeds were generated while running on an Intel Core 2 1.86 GHz processor, with only one CPU core employed during the benchmarking process. The system ran Windows Vista (in 32-bit mode), with the code generating the benchmarks written in C++. Where RSA is employed, the public exponent is set as 17. The benchmark speeds employed are shown in Table 4.1 ("Speed of cryptographic operation"), along with the size of the data being operated on in that time ("Unit size"). We acknowledge that these device specifications are not representative of standard mobile devices, possessing greater processing power than that of a mobile device. However, they do allow a comparison to be made of the relative costs for each extension.

The speed employed in the calculation of transmission costs is based on a IEEE 802.11b wireless network, employing 11mb/s transmission speeds. Based on this setting, the time required to transmit a single bit over this connection is $\frac{1}{11534336}$ seconds. Delays are not factored into the calculation of time costs and the same transmission speeds are applied to those messages sent between the Claimant and Verifier as between the Claimant and Proof Providers. This is done for clarity, as there is no way of gauging the number of hops through which a message from the Claimant to the Verifier would have to travel, though in practice it is assumed that at least part of the journey would be done via an IEEE 802.3 network.

Transmission costs

Graphs (a) and (b) in Figure 4.3 depict the comparison of data and time transmission

[1] Crypto++ `http://www.cryptopp.com/benchmarks.html`

	Extension One	Extension Two	Extension Three
1) C→V:	2112	3072	3072
2) V→C:	6144	7168	19456
3) C→B_i:	20480	20480	25600
4a) B_i→C:	16800	16800	16800
4b) C→B_i:	67200	67200	67200
5) B_i→C:	15845	20480	32640
6) C→V:	19013	20480	34816
7) V→C:	2145	3072	3072
Total	**149739**	**158752**	**202016**

Table 4.3: Transmission Cost Comparison for Claim Involving Five Proof Providers, Each Doing Ten Distance Bounding Exchanges (Quantity of Data Transmitted, in Bits)

	Extension One	Extension Two	Extension Three
1) C→V:	0.183105469	0.266335227	0.266335227
2) V→C:	0.532670455	0.621448864	1.686789773
3) C→B_i:	1.775568182	1.775568182	2.219460227
4a) B_i→C:	1.456520774	1.456520774	1.456520774
4b) C→B_i:	5.826083097	5.826083097	5.826083097
5) B_i→C:	1.373724504	1.775568182	2.82981179
6) C→V:	1.648382707	1.775568182	3.018465909
7) V→C:	0.185966492	0.266335227	0.266335227
Total	**12.98202168**	**13.76342773**	**17.51431552**

Table 4.4: Transmission Cost Comparison for Claim Involving Five Proof Providers, Each Doing Ten Distance Bounding Exchanges (Time Required to Transmit Messages, in ms)

	Extension One	Extension Two	Extension Three
Step 1	$X + 2x$	$3x$	$3x$
Step 2	$xn + x$	$x * (n + 2)$	$3xn + 4x$
Step 3	$4xn$	$4xn$	$5xn$
Step 4a	$n * (y * (k + h + \mathcal{N}))$	$n * (y * (k + h + \mathcal{N}))$	$n * (y * (k + h + \mathcal{N}))$
Step 4b	$n * (y * (h + \mathcal{N} + x))$	$n * (y * (h + \mathcal{N} + x))$	$n * (y * (h + \mathcal{N} + x))$
Step 5	$n * (3x + T + X + L)$	$4xn$	$n * (h + 2\mathcal{N} + 6X)$
Step 6	$3x + T + X + n * (3x + T + X + L)$	$5x + 3xn$	$4x + 6xn$
Step 7	$2x + D_V + X + T$	$3x$	$3x$

Table 4.5: Transmission Cost Comparison Formulae

	Extension 1	Extension 2	Extension 3
SHA-256	0.057490267	0.052683227	0.088215622
RSA Encryption	5.6	10.64	10.64
AES Encryption	0	0	0.00686
Total	**5.657490267**	**10.69268323**	**10.73493562**

Table 4.6: Generation Cost Comparison (in Milliseconds)

costs incurred by each of the three extensions, i.e. the amount of data transmitted over the course of a single protocol run and the time required to transmit that data. The trend lines shown here behave exactly as expected, with the cost of increasing the level of provided security from authentication to authentication, external anonymity and external confidentiality being relatively minor in comparison to that incurred by the addition of complete anonymity and confidentiality. These figures are computed based on the Claimant exchanging messages with five Proof Providers. The figures shown against steps 4a) and 4b) in Tables 4.3 and 4.4 depict the data and time costs for transmitting ten distance bounding exchanges with these five Proof Providers. The formulae for the calculation of the figures shown in Table 4.3 are shown in Table 4.5, with x = the size of the RSA key employed (1024), y = the number of distance bounding exchanges undertaken (10) and n = the number of Proof Providers employed (5). The remaining notation employed is drawn from Table 4.2. Table 4.4 is computed by multiplying the time required to transmit one bit by the number of bits per step (found in Table 4.3).

As shown in Table 4.3, the increase in transmission costs incurred by moving from extension one to extension two is relatively low, at 9013 bits. or a little over one kilobyte. Similarly, Table 4.4 shows the time required to complete a run of extension two to be less than a millisecond more than that required to complete a run of extension one. However, the data and time costs incurred by extension three are a significant increase on those incurred by extension two, with an increase of over 3.5 milliseconds in time costs and 43264 bits in data costs. This makes the third extension of the SLVPGP clearly the most expensive in terms of all transmission costs.

Generation costs

The figures shown in Table 4.6 compare the generation costs (in milliseconds) for each extension, i.e. the data processing time incurred by creating digital signatures and encrypting messages. Digital signature costs are computed based on the time required to hash

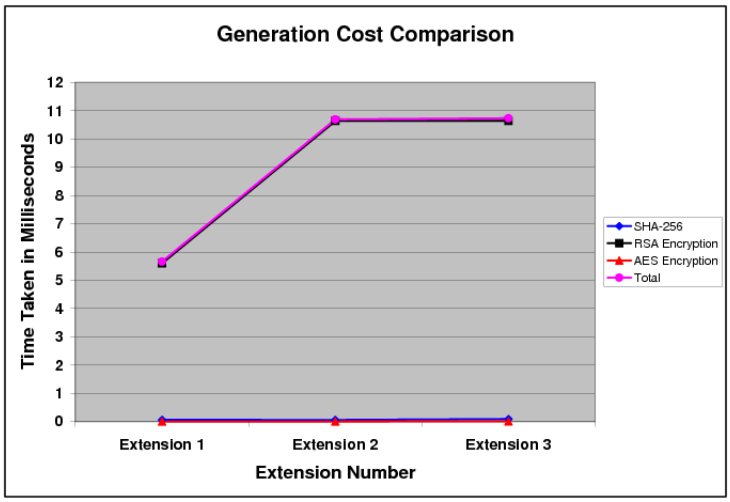


Figure 4.4: Generation Cost Comparison - Practical Costs

Extension #	# Digital signatures	# Encryptions
Extension 1	$5 + 2n + ny$	n
Extension 2	$5 + 2n + ny$	$5 + 2n$
Extension 3	$6 + 2n + ny$	$4 + 4n$

Table 4.7: Generation Cost Comparison (Number of Cryptographic Generation Operations per Extension)

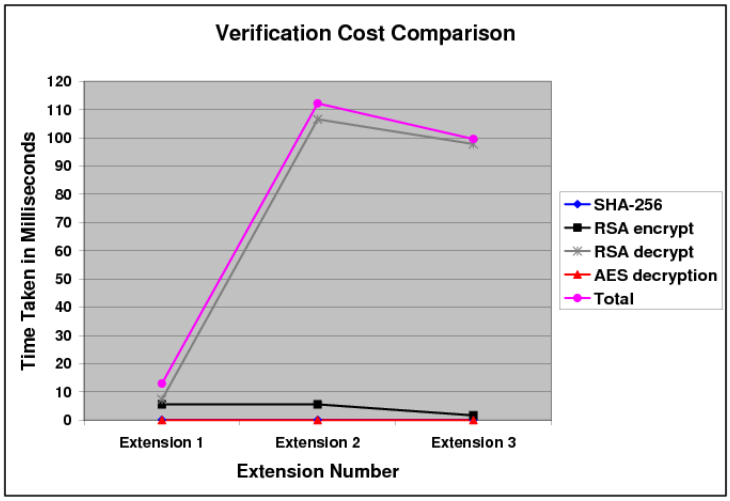


Figure 4.5: Verification Cost Comparison - Practical Costs

a message using SHA-256 and encrypt the hashed data with the applicable public key. Contrary to the situation seen when comparing the transmission costs incurred by each extension, the difference between extensions two and three in generation costs incurred is minimal, with a difference of less than a millisecond. However, there is a drastic decrease in cost when moving from extension two to extension one, illustrated in Figure 4.4, with the cost incurred by extension one equalling just over half that incurred by extension two. This decrease is primarily due to the low level of RSA encryption employed in the first extension, as RSA encryption is the most expensive generation operation in terms of time costs. As seen in Table 4.7, RSA encryption is used in n messages (where $n = 5$, the number of Proof Providers involved in the claim, and $y = 10$, the number of distance bounding exchanges completed) in extension one, less than half the RSA cost of extension 2. However, Table 4.7 does not factor the number of components of the message being operated on into its formulae. It merely indicates the number of cryptographic operations generated within each extension, i.e. the number of digital signatures generated and messages encrypted. This accounts for the apparent extensive difference in encryption costs between extensions two and three, despite having similar time costs in Table 4.6.

Verification costs

Table 4.8 shows the breakdown of definite verification costs incurred by each extension

	Extension 1	Extension 2	Extension 3
SHA-256	0.068998192	0.069684912	0.085125382
RSA encrypt	5.6	5.6	1.68
RSA decrypt	7.3	106.58	97.82
AES decryption	0	0	0.00686
Total	**12.96899819**	**112.2496849**	**99.59184538**

Table 4.8: Verification Cost Comparison (in Milliseconds)

of the SLVPGP, illustrated by the graph in Figure 4.5. It should be noted that the cost of verifying a digital signature is split into the cost of hashing the message using SHA-256 plus the cost of encrypting the hash, using RSA. The trend lines for AES decryption and SHA-256 operations are almost identical across the three extensions, with AES masking SHA's line entirely within the graph. Similarly, the time cost for RSA encryption (in order to verify digital signatures) remains level between extensions one and two, with extension three's RSA encryption costs coming in slightly below these. However, the total cost of verification for extensions two and three is very high, leaving extension one with the only verification cost total below 15 milliseconds. This increase is cost is due to the explosion in RSA decryption costs, a fact shown by the mirroring of the RSA decryption trend line in the overall costs trend.

Within the benchmarks employed (Table 4.1), RSA decryption requires 1.46 milliseconds per key-sized block of data (1024 bits in this example), 18.25 times the time required to encrypt the same quantity. While the method employed for RSA decryption matches that of RSA encryption, the speed at which it is achieved differs greatly, due to the difference in composition between the encryption and decryption exponents employed. The speed at which RSA's modular exponentiation process is completed depends upon the number of bits set to 1 within the exponent. The encryption exponent employed with RSA is traditionally selected from a small pool of options and contains only two bits set, chosen to ensure a speedy encryption process. However, the decryption exponent contains a much larger number of set bits, thus increasing the time required to complete the modular exponentiation process decrypting a message. This spike in costs pushes the overall cost of verification to an extremely high level.

Although extension three possesses the largest quantity of cryptographic content, it does not incur the highest definite verification costs. The reason for this is twofold. The first is that the Proof Providers are unable to verify all digital signatures transmitted during

the distance bounding phase of the protocol. As the Proof Providers no longer possess the Claimant's public key, they can no longer verify the Claimant's digital signature. Instead, only a single verification from this phase is verified per Proof Provider, with the y factor (the number of distance bounding iterations) removed entirely from the cost calculation equation. This is illustrated in Table 4.9, which shows the number of verification operations required for each extension. As in Table 4.7, n = the number of Proof Providers involved in the claim, and y = the number of distance bounding iterations.

The second reason for the decrease in definite verification costs is that the Claimant is unable to decrypt and re-encrypt the final proof messages. This is because the Claimant no longer possesses the identities (and consequentially their public keys) of its Proof Providers. Therefore, it is reduced to forwarding without confirmation of the proof messages' content, thus reducing the number of verifications completed. In addition, this decreases the number of messages to be re-encrypted (thereby also decreasing the generation costs).

A similar contradiction exists in both extensions one and two, between the number of digital signatures generated and the number verified. As seen in Table 4.7, there are $5 + 2n + ny$ signatures generated in both extensions. However, Table 4.9 lists the number of signature verifications as $5 + 3n + ny$, an increase of n signatures. This difference is caused by the verification of the signature of each Proof Provider's proof message twice, once by the Claimant and once by the Verifier. This is not done in extension three, for the reasons discussed above.

Unlike extension three's blind forwarding approach, extension two's design is based around external anonymity, which grants the Claimant knowledge of the contents in each message being forwarded to the Verifier. However, a by-product of this design is the explosion of extension two's costs due to the regular use of RSA encryption and decryption, in addition to the use of the Claimant as a central message hub in the area (a situation which also incurs additional encryption and decryption operations). However, while the difference between extensions two and three is marked, the difference between extension one and the SLVPGP's subsequent extensions is drastic. Extension two costs just over 8.65 times the verifications costs incurred by the first extension. This is due to the low level of RSA operations within extension one, whereas in both extensions two and three, encryption and decryption are used with great frequency.

Extension #	# Digital signature verifications	# Decryptions
Extension 1	$5 + 3n + ny$	n
Extension 2	$5 + 3n + ny$	$5+ 2n$
Extension 3	$6 + 3n$	$4 + 4n$

Table 4.9: Verification Cost Comparison (Number of Cryptographic Verification Operations Per Extension)

	Extension 1	Extension 2	Extension 3
Transmission time	12.98202168	13.76342773	17.51431552
Generation time	5.657490267	10.69268323	10.73493562
Verification time	12.96899819	112.2496849	99.59184538
Total	**31.60851014**	**136.7057959**	**127.8410965**

Table 4.10: Overall Comparison of Time Costs (in Milliseconds)

However, as mentioned above, the cost comparisons discussed here only factor in the definite verification costs incurred by extension three. They do not include the additional costs caused by the employment of broadcasting and the lack of knowledge on the part of the participants. While it is impossible to establish the impact of these additional costs on the overall verification costs for extension three, it can be definitively stated that they will certainly increase them. As increase is relative to the number of Proof Providers active in the area, both on this claim and on others, the verification costs for extension three become unpredictable, with only a minimum value available.

4.5.5 Comparison Summary

Each of the three extensions to the SLVPGP are fully secure against collusion attacks, in addition to maintaining the integrity of the proof messages supplied to the Verifier by the Claimant. While extension one provides only authentication and protection against tampering, extension two introduces anonymity and confidentiality on an external level (protecting private data from those not participating in that particular claim). Extension three then increases these properties from external to complete. Though the complexity of extension three appears vastly greater than that of extensions one and two, it is comprised primarily of forwarding and, as such, is structurally quite simple.

Table 4.10 shows the overall time costs (in milliseconds) for each protocol extension, broken down by category. The data shown here indicates that extensions two and three of the protocol are quite costly, particularly in terms of the verification time required. The

employment of RSA cryptography within the protocol increases the quantity of data to be transmitted (Table 4.3), both due to the size of the identities employed and due to the block size messages are padded to in order for encryption to occur. However, if an alternative form of public key cryptography was employed, these costs could be dramatically reduced, as schemes such as Elliptic Curve Cryptography (ECC) require less time for encryption and decryption, in addition to employing smaller keys, thus reducing the quantity of data to be transmitted.

If the situation in which location verification is to be employed requires a lightweight protocol, due to the employment of smaller devices with lower power and slower transmission speeds, extension one is the most appropriate selection. It incurs the fewest overheads, costing 31.6 milliseconds to complete and requiring the transmission of 149739 bits of data. However, if device privacy is required, then extension one is unsuitable for use and extensions two or three must be employed. Extension two incurs the highest definite overall time costs, due to the high cost of RSA decryption coupled with the quantity of data to be decrypted, though extension three may exceed these if executed in a high-traffic area. However, extension three incurs the highest data transmission cost, requiring the transmission of 43264 bits more than extension two. Therefore, if the deployment scenario places a higher premium on speed, extension three would be most suitable (assuming the area in which it is run does not have high levels of network traffic), while if data transmission is required to be lower, extension two should be employed (assuming full anonymity is not also a requirement).

4.6 Summary

In this chapter, we outlined our design for a secure proof gathering protocol. We discussed the approach taken in creating this design and the security properties required in order to provide a secure and complete proof gathering protocol. These properties have been selected based on the requirements of the system's users and those which protect the integrity of the proofs gathered within the protocol. In order to analyse the basic approach designed, we have outlined the bare-bones protocol, devoid of security measures. From here, we created three variations of the protocol, each with an increased level of security. This was done to provide flexibility in the amount of security relative to its costs. Each

extension has been defined and its security and vulnerabilities discussed. We have designed a single secure lightweight extension which can be used in situations where anonymity and confidentiality are not an issue. This extension protects the integrity of the proofs gathered, but does not protect the private data of the participants. The second and third extensions to the protocol represent two variants of a secure and complete protocol, in which all three security properties are supported in differing degrees. In order to gauge the level of cost incurred relative to each increase in security, we have analysed each extension's generation, verification and transmission costs. This analysis demonstrates the marked increase in costs required to provide complete anonymity and confidentiality, proving the second and third extensions quite similar in costs, despite the appearance of additional complexity and cost in extension three.

Chapter 5

Proving the Security of the Secure Location Verification Proof Gathering Protocol

5.1 Introduction

In Chapter 4, we discussed the need for the process of gathering location proof to be secured in some way. We outlined the design of the Secure Location Verification Proof Gathering Protocol (SLVPGP) and discussed its intended security properties and the reason for their inclusion. In this chapter, we outline the process of converting these protocols from security notation to mathematical models, before formally verifying them using model checking.

The SLVPGP is a complex protocol, designed to protect the distance bounding exchanges between a Claimant and its corresponding Proof Providers in order to ensure that the proofs received at the end of the exchanges are not tampered with in any way. In order to ensure that the proofs received are not vulnerable to tampering or any form of malicious interference, we wish to validate the protocol's correctness through formal verification. While validation of the protocol in this manner does not prove its ability to accurately verify a location claim, it does constitute a valid security investigation and is done to ensure that the protocol does not contain any hidden flaws or vulnerabilities that may leak information or lead to flawed functionality, despite its secure appearance. Even the simplest of protocols may contain some hidden security issue which leaves it vulnerable

to attack. For example, at the time of its publication the Needham-Schroeder Public-Key protocol [67] was considered to be secure. However, in [57] Gavin Lowe proved the protocol vulnerable to a man-in-the-middle replay attack using interleaving when employed outside of its originally intended environment. This attack remained undetected for many years, but with formal verification the attack is discovered immediately. After the publication of Lowe's paper, the formal verification of protocols experienced a surge in research, particularly the verification of authentiation and electronic commerce protocols. The practice of formal verification has become increasingly common as a method of confirming the security of released protocols. Examples of protocols formally verified after their publication include the Secure Electronic Transaction (SET) protocol suite [10, 12], the Generalized Pre-Shared Key method of the Extensible Authentication Protocol (EAP-GPSK) [65] and Kerberos [17, 28].

The formal verification process involves the analysis of the protocol's security in order to ensure that the protocol's intended security properties are achieved and that the protocol is secure from attacks. There are two main approaches to formal verification, model checking and theorem proving. Formal verification through theorem proving and induction is used to verify systems in which there may be an infinite number of possible states. This is due to the fact that states are not enumerated and therefore the number of agents and intruders can be considered to be unbounded. The various possible event sequences that could be done by an agent are defined based on inductive reasoning using a set of rules. These rules correspond to all possible actions that could be taken by agents involved, including hostile agents. Security properties are defined as predicates which must hold over all the traces.

Model checking is used in the verification of systems with a finite number of possible states. It consists of an exploration of all possible states and transitions between states contained within the model, based on the powers and knowledge of the agents involved. This gives a complete check of the system and exposes any potential flaws. The model checking approach to formal verification has previously been used to verify the security of schemes such as SSL 3.0 [66] and NetBill [45]. We have chosen to employ this method for the formal verification of the SLVPGP, as the protocol's mathematical model is finite and can therefore be fully explored. Additionally, during a model checking verification, the model checking tool investigates whether the specifications of the system are upheld

throughout. This aspect makes model checking an especially apt tool for the verification of our claimed security properties, as they can be expressed as specifications on the system.

We employ Casper [58] and the Failures-Divergences Refinement (FDR) [83] tool to accomplish model checking, using the Casper software to generate a Communicating Sequential Processes (CSP) [14] description of the protocol. The CSP description is then model checked by FDR, where the security specifications used to model the properties being checked are confirmed to either be true or false for this system. However, the CSP model generated can be complex, producing a state space which is intractable to many computer systems. Therefore, we require some method of simplifying the protocol's model without losing the correctness of the model. If this is lost, attacks on the protocol could be missed by FDR while carrying out a model check, nullifying the check's results. The issue of transforming the protocol to simplify the state space whilst retaining the correctness of the system is discussed further in section 5.4.

In this chapter, we first outline the process through which the protocol is formally verified (Section 5.3). In Section 5.4, we discuss the simplifications performed on the protocol models. Sections 5.5.3 - 5.5.5 detail the original design for each extension of the protocol, translated into Casper, before applying and explaning the transformations employed. In Section 5.7, we outline our investigation into the security of employing broadcasting as a method of transmitting secure messages. Finally, we summarize the chapter's contents in Section 5.8.

5.2 The Limits of Formal Analysis

Before outlining the formal analysis process employed to verify the security of the SLVPGP, we first outline the limitations of formal analysis in general. As mentioned in the previous section, formal verification of the location verification protocols presented in this work does not prove that the protocols will accurately verify location claims. The purpose of formally verifying a protocol is to confirm that at no point during the run of a protocol will the security specifications laid down within the system's design be violated. This provides supporting evidence of the hypothesis that that particular protocol is secure from any attacks. However, although formal verification is a useful tool for probing the security of a protocol's design, a system can never be proven to be 100% secure [103]. As systems

do not run within a vacuum, the verdict received on a protocol reflects only that within the system described, there does not seem to be any insecurity related to the included security specifications. If formal analysis indicates that the system being verified is secure, it indicates that, within the environment described for analysis, this system is secure from the attacks specified and investigated, based on all specified assumptions.

However, if even one of the assumptions the described system is based upon is inaccurate, an assumption is not included in the model or the protocol is used in an environment different to the one modelled, the results of the verification are no longer valid (c.f. Lowe's discovery regarding the Needham-Schroeder Public-Key protocol [57]). For example, when model checking our protocols, we assume the strength of the cryptography employed to be complete, i.e. that an intruder cannot crack the code. If the encryption employed on the protocol was insufficient, the model would not accurately reflect the real world deployment of the protocol and therefore the verification results could no longer be applied to the protocol.

This weakness also extends to the security specifications employed to model the security properties being checked for a protocol. If the specification is designed poorly and does not accurately reflect the real-world security property, its success or failure in a verification gives no indication of whether that property is upheld by the protocol. Finally, without a sufficiently rich threat model describing the knowledge and powers of the intruder process, the verification verdicts received regarding a protocol could not be applied to the real world protocol, as the intruder was not fully described. We discuss the approach taken to modelling the intruder and its importance in Section 5.3.3.

5.3 The Model Checking Process

A security protocol is defined as a high level description of the steps taken by those devices participating, written using abstract protocol notation. In order for this description to be formally verified, a model must be created which describes the sequence of steps involved in the protocol and the environment in which it operates. This formal representation of the system can then be analysed for vulnerabilities and the intended security properties of the system verified. However, as discussed in Section 5.2, this verification is contingent upon the protocol being used within the specified environment.

As mentioned in Section 5.1, several languages and tools are employed to formally verify the protocols. In Section 5.3.1 we introduce the tools employed in model checking - CSP (Communicating Sequential Processes) and FDR (Failures-Divergences Refinement). Although CSP can be handcrafted, we have opted to employ an automatic generation tool (Casper) to translate a high level description of the system into the low-level CSP description required by FDR. The reasoning behind this approach is discussed in Section 5.3.2. Finally, we discuss the role of the intruder within our model in Section 5.3.3, outlining the assumptions made regarding the capability of this attacker.

5.3.1 What is Model Checking?

As mentioned in Section 5.1, model checking is a method of formally verifying systems with a finite number of possible states. When applied to security protocols, it formally validates the design of a protocol, indicating that the protocol conforms to the specified security properties included in its description. We employ CSP and FDR to accomplish model checking. CSP is a complex notation for describing systems in which messages are passed between agents (processes) operating in parallel [84], thus making it ideal for the verification of this system. It allows the security of concurrent systems to be examined fully, and provides the ability to describe all the possible states in which a system may be. As CSP is such a complex language, we employ a high level modelling language (Casper [58], discussed in Section 5.3.2) to generate a CSP description of the protocol. The resulting CSP description is model checked by FDR. In FDR, the included specifications used to model the protocol's security properties to be checked are confirmed as either true or false for this system. In this section, we outline the basic functionality of CSP and how FDR verifies a supplied CSP model.

CSP

CSP is a mathematical language used to describe the interaction of processes operating in parallel. These interactions occur in the form of communications between processes, referred to as events or actions. CSP describes each process in terms of states and events, with each process beginning in a specific state with a corresponding set of possible actions. When a process runs, it selects an event from its set of all possible performable events for that state, performs it and moves to the next logical state. The continued running of

a process in this manner produces a *trace* or specific sequence of visible communications that can be performed. For each process there exist multiple traces, as due to the process's ability to choose an event to perform, there are multiple possible event combinations. By computing all possible traces produced by a process, a traces model can be built. This model details all possible ways in which a process can behave, allowing that process's behaviour to be thoroughly examined for flaws by tools such as FDR.

Where the traces model concept is extended to include the traces of multiple processes within a system, it creates a model of all possible ways in which an entire system can behave. However, this type of model is far larger than that of a single process, as where only a single process is modelled, only a single state must be considered between each event. When modelling an entire system rather than a single process, the concept of a state is altered slightly. Unlike a process's state, a system's state is composed of one state for each component process. Therefore, as with traces, there are many different possible states within the system, as there are multiple possible state combinations. This causes a dramatic increase in the number of states to be explored (the state-space) when checking the system.

In order to investigate a protocol's security properties, additional processes are created, known as specification processes. These processes are added to the CSP model of the protocol. Each specification process models one or more security properties (multiple secrecy properties may modelled in a single secrecy specification process). These processes model what the system is expected to be capable of at specific points. In order to define what the system is capable of, signals (specification events) are introduced. A signal event is a specification/control event in CSP, used in the checking of security property specification assertions [88]. Signals are sent on a special channel outside the protocol, and indicate a particular claim regarding a security property. For example, if the nonce n was to be kept secret between agents a and b, the signal:

$$signal.Claim\_Secret.a.b.n$$

would be sent on the signal channel, indicating that at that point, a and b are claiming that n is a secret shared by them. As it is sent outside the protocol communications channel, the intruder cannot overhear it and learn any new information. If this claim is made on the signal channel, the intruder should not be able to leak the value of n, i.e. it

should not know n's value.

FDR

FDR's functionality is based on the concept of comparing processes and seeking *refinement*. If one process is a refinement of another, anything permissable to occur in the first process is also allowable in the second process. For example, when given two processes, *Implementation* and *Specification*, FDR checks whether *Implementation* is a refinement of *Specification*, i.e. whether all the traces in Implementation's traces model are contained within Specification's traces model. If so, *Implementation* is said to refine *Specification*. [84]. FDR is also capable of investigating systems regarding vulnerability to deadlock (known as *failures*, in which processes refuse to perform some event and cannot progress) and livelock (known as *divergences*, in which processes continually perform internal actions rather than progressing), however these aspects are not employed in our work.

When refinement checking a complete system, FDR employs the concept of refinement to confirm that the model is a refinement of the CSP specification process supplied. It achieves this through systematically comparing the traces model generated by the system being checked, based on the processes described, with those of the specification process provided. In the case of modelling security protocols, this specification is a process encoding the security property or properties being investigated, e.g. anonymity. If the security properties investigated are not upheld over every trace in the traces model of a system, the model is not a refinement of the specification. In this situation, FDR fails the check and returns an example trace that violates the refinement.

FDR also employs refinement to investigate the security properties of a protocol. When investigating a protocol's security properties, the CSP model of the protocol is refine checked against a set of security specifications (discussed in the previous section) representing each property. If the CSP model refines the security specifications, the security properties represented are upheld. This is proven by the fact FDR cannot find any traces in which the intruder is able to violate the rules of the specification. However, if FDR returns a counter-example, the security properties are not upheld. An example of this would be where a secrecy property is being investigated and FDR returned a counter example in which the intruder could leak the message component being claimed as secret.

FDR's method of refinement checking requires enumerating vast numbers of states and

comparing all possible traces within a model. Therefore, the effort required to check a model depends upon the complexity of the process or processes involved. Due to the possibility that there are a vast number of states to be explored (the state-space) when checking a system, verifying a model can be extremely complex and resource-consuming. In the case of model checking the SLVPGP's extensions to confirm the security properties specified in their design, the models generated are extremely complex. For this reason, a selection of simplification techniques have been performed on the systems checked in this work, in order to reduce the size of the resulting models. These simplification techniques are discussed in Section 5.4.

5.3.2 Modelling a Protocol Using Casper

In order to generate CSP models to be checked by the FDR model checker, we first translate each of the protocols from abstract protocol notation into Casper. Casper provides automatic generation of CSP models based on a high-level Casper description. In this section, we outline the reasons for employing Casper in this process. We then describe the way in which a protocol is translated from abstract notation to Casper. Finally, we outline the modelling of security properties in Casper and how they translate to CSP.

Why Use Casper?

In order to model check a security protocol, it must be translated from its security notation design to a script representing this design in some modelling language. Casper has been designed to emulate the form of a high level programming language, simplifying the process of describing a protocol. Using Casper allows us to automate the production of the protocol's mathematical model (its CSP description), reducing the liklihood of errors occuring. When writing in a language such as CSP, the density and complexity of the code often results in overlooked errors and logic flaws. This leaves the system vulnerable to undetected security issues, as the model being checked does not accurately represent the original protocol and the environment in which it operates. Employing a higher level language such as Casper to generate CSP code allows a clear picture of the system being modelled to be visualised.

Modelling in Casper

When modelling a protocol in Casper, both the protocol and the environment in which it is being formally modelled are described. Details on the environment in which the protocol is to be run are included because all protocols operate based on some set of assumptions. This information is required in order for the model to be accurate. Within a Casper script, the model's description is broken into two parts, each dealing with multiple aspects. The first part of the Casper script contains information on the theoretical system's design - the "agents" (participants) in the protocol (e.g. the Verifier, Claimant, etc) and their initial knowledge (#**Processes**); the sequence of steps to be taken within the protocol (#**Protocol description**); the types of any protocol participants defined in #Processes and message components employed in the protocol (#**Free variables**); and a set of specifications detailing the security properties being investigated (#**Specification**). In essence, this part of the script describes an abstract model of the system being analysed, how it should perform and the specifications regarding its behaviour that it should meet. It results in a CSP description of the messages to be sent (based on the protocol steps) and CSP descriptions of the agents involved, describing the order in which they send and receive those messages.

The second part of the Casper file deals with the system being investigated in this analysis, i.e. it defines a specific instance of the model described in the first part of the file, instanciating it with specific parameters. It contains details on the variables employed in this instance (#**Actual variables**) and the agents taking part in this specific exchange (#**System**). Each agent detailed in the second section of the script is modelled in CSP as a process, parameterised with the variables supplied under the #Active variables heading.

In addition to dealing with the honest participants within the system, this section of the Casper file also includes information regarding the intruder and its knowledge (#**Intruder Information**). This process represents any intruder within the system being checked. The specification of a system are deemed to be upheld if this process is unable to violate them at any point during the check, therefore the intruder's design must be very thorough. When building a CSP model by hand, modelling the intruder process is a very difficult task, as many different possibilities must be included. When translated into CSP by Casper, the intruder is provided with a deduction system which processes any information it overhears

based on its own knowledge, in an attempt to learn new data. This system allows the intruder to deduce all possible information, without requiring a large quantity of complex code to be written by an investigator. This reinforces the argument for employing a higher level language to generate the CSP. Higher level languages are simpler to process and work with, allowing for errors and complexity issues to be discovered and dealt with easily. The powers of the intruder and its impact on the model's accuracy are discussed further in Section 5.3.3.

Modelling Security Specifications

As discussed in Section 5.3.1, the modelling of security properties in CSP is accomplished through the use of security specification processes, built to represent the events allowable by those properties. These specifications are represented in Casper by simple specifications listed in the #Specifications section. Casper translates these security specifications into specification processes during the CSP generation process. When modelling the extensions of the SLVPGP, we employ two Casper specification types: StrongSecret and Agreement. The definitions of these specification types is sourced from [84].

The Agreement specification models authentication in the form of an agreement on the value of a message component, by a specified list of participants in the protocol. It is presented in the form:

- Agreement(A, B, [nA])

where A and B are participants in the protocol and nA is the value being agreed upon. A successful verification of this Agreement specification indicates that if B believes it has successfully completed a protocol run with A, then A has been running the protocol with B. It also indicates that there is a one to one relationship between the number of runs of A and the number of runs of B. Finally, it indicates that A and B agree on the value of nA, i.e. that the value of nA does not change between when it is sent by A and when it is received by B.

The StrongSecret specification is a variant of the Secret specification, which models a requirement of secrecy regarding a specified value. However, in the Secret specification the *Claim_secret* signal event (discussed previously in Section 5.3.1) is performed at the end of the protocol. This leaves the case where a protocol run is started and aborted

prior to the end of the run unchecked, as the signal event is never performed. Where the information being classed as secret has significance outside of the protocol exchange, this level of security is insufficient. As this is the case for the protected information in the SLVPGP extensions, the StrongSecret specification is employed to investigate the secrecy security property.

The StrongSecret specification addresses this shortfall through moving the signal event from the end of the protocol's run to the earliest possible moment, i.e. at the initialisation point of the protocol for the sender and at the point of receiving the value for the receiver. This causes the security specification generated to confirm that the value remains secret throughout the protocol, thus confirming the security of the value even for aborted protocol runs. The StrongSecret specification is presented in the form:

- StrongSecret(A, nA, [B])

where A and B are participants in the protocol and nA is the value being kept secret. A successful verification of this StrongSecret specification indicates that A is correct in believing that nA is a secret known only to itself and B. However, if B is an intruder, the value could be passed to any other device and the protection of the value lost. For this reason, Casper generates a condition within the CSP specification process indicating that this specification is investigating only the case where B is not an intruder. If this specification is upheld, it indicates that if B is not the intruder, the intruder will never learn the value of nA.

5.3.3 The Role of the Intruder

A crucial aspect of model checking is the inclusion of a specific process designed to behave maliciously in order to test the strength of the model's security. This *intruder* process represents any malicious entity, either a member of the system or an outsider, which may attempt to attack the system in any way. When modelling a system in Casper, the intruder process is automatically generated based on a specified set of known facts, such as the identities of the devices it has knowledge of, any public or private keys (or key access functions), etc. Casper employs the Dolev-Yao [24] intruder approach, a worst-case scenario in which the intruder is given complete power to tamper with messages sent within the system, bounded only by the limitations of its own knowledge (both learned and

previously known) and its cryptographic ability. It can obtain any message transmitted within the system and has the ability to replay, fake, redirect and alter messages in an effort to learn protected information and break the security of the system.

However, as the intruder's knowledge is composed only of what it is granted initially and the information it has learned over the course of an exchange, it cannot use a value without having first overheard it in a communication or already being in possession of it. If the intruder overhears said value within a message, it can logically make use of it at a later date. Additionally, the intruder can only decrypt those messages for which it possesses the correct cryptographic decryption key, and vice versa with regard to encryption. This approach is superior to employing an intruder process designed only with specific malicious attacks and devient approaches, as it is less limited. The intruder process is capable of performing any action which falls within the stated limitations at any time and can therefore discover any vulnerability within the described system.

When modelling the SLVPGP in Casper, the intruder process is modelled with the ability to function as each of the players within the system (excluding the process used to model the network, introduced in Section 5.5.5.1). Prior to beginning a run of the protocol, the intruder's knowledge set includes the identity of the Verifier, its own private keys as well as the public key functions for all involved types. These functions allow the possessor to retrieve the public key of a device, if they are in possession of its identity, an aspect of the form of cryptography employed in the real world system. In order to further test the system's security, multiple other checks were conducted, granting the intruder process various pieces of additional information. These checks are further discussed in Sections 5.6.1 - 5.6.3.

5.4 Safely Simplifying the CSP Model

As mentioned in Section 5.1, the CSP model based on the SLVPGP is highly complex. Due to the number of variables included in each message and the large quantity of encryption employed over the course of a run of the protocol, the state space generated is vast. This is particularly true in the case of the second and third extensions of the protocol, where encryption is employed heavily and the number of variables increases drastically. Some method of reducing the model's complexity is required in order to ascertain the correctness

of the protocol, as FDR cannot comfortably handle such a complex model. In order to avoid interfering with the accuracy of the model while reducing its complexity, we have employed several safe simplifying transformations, found in the work of Schneider & Ryan in [84]. The authors prove that if certain aspects of a protocol's complexity are not needed for the security of that protocol, the protocol can be analysed with said complexity removed.

The transformations defined by this work posses the property of preserving attacks on the original protocol, as they satisfy a pair of safe transformation conditions. These conditions state that for a particular simplification *f*, for every message sequence (trace) *tr* that can be generated by P, *f(tr)* is a trace of P's simplified version. If *tr* constitutes an attack on P, then *f(tr)* constitutes an attack on the simplified version. Any information an intruder could gather or deduce based on the original protocol could also be gathered or deduced from the transformed protocol. Therefore, transformations meeting the conditions set out above allow the model's complexity to be decreased without affecting the model's correctness. Three such transformations are employed in this work to reduce the complexity of the SLVPGP. These are message splitting, message redirection and coalescing atoms.

5.4.1 Coalescing Atoms

The simplest transformation applied to the protocol is the coalesing of a pair of atoms into a single atom. In this transformation, a pair of atoms in the protocol is substituted with a single atom representing both, usually a member of the pair. Consider the following simple Casper protocol description:

1. A->B: nA

2. B->A: nA, B, nB, nB1

3. A->B: B, nA1

This basic structure shows an entity A sending a nonce to entity B. It then receives back its own nonce from B, along with two of B's nonces and B's identity. Finally, A replies with B's identity and its own second nonce. When we apply the coalescing simplification to this protocol, we coalesce nB and nB1 into a single value. This results in the protocol description being transformed to this:

1. A->B: nA

2. B->A: nA, B, nB

3. A->B: B, nA1

with nB and nB1 represented solely by nB in the second step.

The transformation process also transforms all traces of the protocol using the same function (in this case the coalescing of a pair of atoms into a single atom) applied to the protocol. Therefore any trace found in the original protocol's traces model will also be produced (in a transformed state, i.e. with the pair of atoms replaced by a single atom) by the transformed model. This connection between the original and transformed designs preserves any possible attacks present in the original model.

As the security specifications employed to model the system's security protocols deal with the protocol in its original state, the coalescing of these atoms into a single atom also has a direct effect on any specifications involving them. In order to investigate the transformed model's security properties, we amend any specifications involving a coalesced pair of atoms to cover only the atom with which the pair is substituted. If the security of this atom is compromised, either through the leaking of the value or an inability to authenticate the value, this flaw also applies to the original pair of atoms. Therefore, the results of the security properties checks on the transformed model can be projected onto the original model.

5.4.2 Message Splitting

In the second safe simplifying transformation, a single message is split into two separate messages. This is done to reduce the size of the message space, i.e. the number of different possible messages in the model. For every message in the model, there are x possible different combinations of atoms which could make up this message. If a message is composed of a large number of atoms, there is a corresponding very large number of possible atom combinations. Splitting the message into two separate smaller messages reduces the number of possible combinations for each of the message halves. However, the resulting drawback is that the state space size is increased, as the number of messages within the model increases. For this reason, message splitting is only applied if a) the number of combinations significantly decreases and b) the decrease is not overpowered by the inclusion of the second message half and all its possible combinations. We illustrate the message

splitting transformation using the same simple Casper protocol description used in the previous section. Initially, the Casper description is as follows:

1. A->B: nA

2. B->A: nA, B, nB, nB1

3. A->B: B, nA1

However, by applying the message splitting transformation to step two of the protocol, the protocol description is transformed to this:

1. A->B: nA

2. B->A: nA, B

2a. B->A: nB, nB1

3. A->B: B, nA1

thus decreasing the number of possible combinations within step two's message and simplifying the state-space to be checked. Note that when applied, the resulting transformed protocol contains all elements from the original protocol.

Similar to the coalescing transformation, performing the message splitting transformation on the protocol does not affect its correctness. Both message splitting and message redirection (discussed below) are examples of *structural transformations* - transformations in which the structure of a message is altered but not the contents of the message itself. The composition of any message being split within the protocol is unchanged. Therefore, if a trace tr exists within the original traces model that violates any of the included security specifications, there exists a corresponding trace tr' in the transformed traces model that also violates that specification. If encryption was present in the original message, the same level of encryption is employed on both message parts. However, unlike in the case of the coalescing atoms transformation, the employment of message splitting within the protocol does not require any of the security property checks to be altered. All atoms within the split messages remain unchanged.

5.4.3 Message Redirection

The final safe simplification employed to reduce the protocol's complexity is message redirection. This transformation replaces two messages with a single message, redirecting a message which was originally sent via a third party so that it is sent direct. This removes a third party from the proxy role, as well as decreasing the number of messages in the model's state space. As no part of the message is renamed or amended by the simplifiction, the signal events produced within the transformed protocol are unchanged from those of the original. If a transformed model produces the same signal events as the original, it must still be performing the same critical steps and therefore any possible attacks are preserved. Consider the following simple Casper protocol description:

1. A->B: A

2. B->C: A

3. C->B: C, nC

4. B->A: C, nC

5. A->B: nA

6. B->C: nA

In this protocol, entity B acts as a proxy, forwarding messages between entities A and C. This process increases the size of the model greatly. However, if the message redirection transformation is applied, the model shrinks noticeably, eliminating half the protocol steps:

1. A->C: A

2. C->A: C, nC

3. A->C: nA

As with message splitting, the application of this transformation to the example protocol does not impact the contents of the messages being transmitted. The only difference is that the steps including B are removed. However, B can learn as much about the contents of the messages as it could while it was involved in the protocol, as the same messages are sent over the network.

As mentioned in the previous section, message redirection is an example of a structural transformation. The alterations are not done through renaming of atoms within a message but through changing the structure of the message itself. Although this appears to alter the model to no longer reflect the protocol being checked, any attacks or vulnerabilities present on the protocol's original structure are preserved in a structural transformation as no signal events are altered. If a trace *tr* exists in the original which violates any included security specification, there exists a corresponding trace *tr'* in the transformed model which violates that same specification. Therefore, an intruder can mount the same attack on the transformed model as the original, as if he can obtain the contents of the original message, he can also obtain the contents of the redirected message.

It is important to note that structural transformations reflect their original model only if the signal events in the transformed model are in the same place as in the original model, even if this is not the natural position for them in the transformed model. This ensures that even in the transformed model, events affecting the security properties being modelled occur in an order reflecting that of the original model.

5.5 Modelling the SLVPGP in Casper

5.5.1 System Assumptions

When designing the Casper models that represent the SLVPGP in the model checking process, a number of assumptions were made, based on both the original protocol and its environment. In this section we clearly outline those assumptions, broken into four categories. They are Encryption, Message Sending, Simplifications and the Intruder.

Encryption

As with other models of real world protocols, assumptions must be made regarding the encryption methods employed. We employ the standard assumption regarding encryption, in that all participants employ a universally known encryption/decryption algorithm. It is also assumed that both the algorithm and the strength of the keys employed are of sufficient strength that devices that do not possess the correct key cannot correctly encrypt or decrypt a message. This means that if an entity within the model receives an encrypted message and does not possess the correct decryption key, it cannot gain access to the

message contents.

Message Sending

When modelling protocols employing message sending, the security of the medium over which a message is sent must be considered. In this case, it is assumed that the transmission medium is open, in that any message sent can be received by any entity within the system and therefore is not secure without the use of reasonable encryption. It is also assumed that once a message is transmitted, it cannot be prevented from reaching its destination, i.e. message jamming, either selective or general, is not taken into consideration.

Simplifications

As discussed previously, the Casper-generated CSP models of the SLVPGP extensions are highly vast and complex. As such they are beyond the limitations imposed by hardware capability at this time. For that reason, the models have been simplified using the simplification approaches discussed in Section 5.4. We assume that the simplifications employed are attack preserving, as shown in [49], and that the resulting simplified models remain true to the original protocol design.

The Intruder

The modelling of the intruder is a key aspect of any formal verification process. If the intruder does not accurately reflect the threats posed by attackers in the real world, the results of that verification are nullified as they do not capture the reality of the system. As discussed in Section 5.3.3, the Dolev-Yao intruder model [24] is employed in the SLVPGP verifications. We assume that this model accurately represents the real-world malicious device threats, with all the powers and abilities of the same.

5.5.2 Proof Provider Representation

Before outlining the Casper models of the SLVPGP extensions and the processes employed in their simplification, we first discuss the representation of Proof Providers within them. Within the various models discussed in Sections 5.5.3, 5.5.4 and 5.5.5, the role of the Proof Provider is represented using only two processes. This limitation essentially reduces the number of Proof Providers participating in a claim to only two per Claimant. While

including a larger number of Proof Providers in the Casper model would more accurately represent the real world situation that will be faced, this is not currently feasible. The size and complexity of a model employing only two Proof Providers is too unwieldly for immediate model checking, and requires the use of simplification techniques before it can be successfully checked. Increasing the number of Proof Providers represented would explode the CSP model's size beyond the limits of the equipment available at the time of writing. Therefore, the models described in this chapter model only a limited version of the system and do not rule out the possibility of collusion attack where more than two Proof Providers are used. Though the fact that the model checks discussed in Section 5.6 succeed provides a positive indication that the system will remain secure with the addition of more Proof Providers to the model, it has not yet been proven.

5.5.3 Formally Modelling Extension One

In this section, we outline the process of translating the first extension to the SLVPGP from security notation to a model for use by FDR. This process is accomplished in two stages: The original design of the protocol is first translated into Casper from abstract protocol notation, with the extension's desired security property modelled as two Casper authentication specifications (Section 5.5.3.1). It is then simplified to reduce its complexity (Section 5.5.3.2).

5.5.3.1 Modelling Extension One - From Abstract Notation to Casper

Before applying the safe simplifications outlined in Section 5.4, we first translate extension one's abstract protocol description into Casper. This process requires not only information regarding the protocol's steps but also details of the environment in which it will run and the knowledge granted to the various participants. The full Casper script is included in Appendix A, with the translated protocol description shown below.

0. -> C: xC

1. C -> V: {C, xC}{SKAgent(C)}

2. V -> C: {P, Pb}{SKVerifier(V)}

3. C -> P: {P, C, {nP, n1P}{PKProver(P)}}{SKAgent(C)}

4. C -> Pb: {Pb, C, {nPb, n1Pb}{PKProver(Pb)}}{SKAgent(C)}

5. P -> C: kP, hP, n2P

6. Pb -> C: kPb, hPb, n2Pb

7. C -> P: h1P, {n2P}{SKAgent(C)}

8. C -> Pb: h1Pb, {n2Pb}{SKAgent(C)}

9. P -> C: {P, tP, xP, lP, C}{SKProver(P)} % mP

10. Pb -> C: {Pb, tPb, xPb, lPb, C}{SKProver(Pb)} % mPb

11. C -> V: {C, tC, xC}{SKAgent(C)}, {mP % {P, tP, xP, lP, C}{SKProver(P)},
mPb % {Pb, tPb, xPb, lPb, C}{SKProver(Pb)}}{SKAgent(C)}

12. V -> C: {dV, xC, C, tV}{SKVerifier(V)}

The role of the Proof Provider has been represented using two processes (P and Pb), behaving identically, in order to model the inclusion of multiple Proof Providers in a single exchange (Section 5.5.2). The role of the Claimant is represented by the process C, with the process V representing the Verifier. Encryption of a message using public key cryptography is done using a particular function to retrieve the key to be employed. The function is supplied with a particular identity and returns that process's key. The encryption function is paired with a second function, which provides the inverse, i.e. signing, key. There is a function pair for each process type, as each requires a different identity parameter. The encryption functions are noted PKAgent() (resp. SKAgent() to retrieve the signature key), PKProver() (resp. SKProver()) and PKVerifier() (resp. SKVerifier()), where Agent, Prover and Verifier are the process types of the key owners. Finally, the % notation allows participants to accept a message without having knowledge of its composition, thus allowing the forwarding of an encrypted message without possession of the appropriate key to decrypt or recreate it. The intruder process is given the ability to participate within the system as any of these process types, thus allowing it to behave as any of the above processes.

In order to verify the security properties believed to be upheld by the protocol, outlined in Section 4.4.1 of Chapter 4, a set of Casper security specifications are employed. These specifications provide FDR with a set of conditions which must remain true for every possible state in the generated state space, i.e. the model being checked must be a refinement of these specifications. If this is not the case, the check fails and the model

is deemed flawed. Details regarding Casper's interpretation of the specifications employed are outlined in Section 5.3.2.

The aim of the first SLVPGP extension is to provide authentication between users, i.e. that a value given by a device cannot be manipulated between its initial transmission and its arrival at its final destination. This is modelled in Casper as a set of agreements; those between the Verifier and each Proof Provider regarding the value of their proof atom and a single agreement between the Claimant and the Verifier regarding the value of its final verdict. These agreements confirm that the value transmitted during an exchange cannot change between its transmission from the source device (either a Proof Provider or the Verifier) and its receipt by the destination device (either the Verifier or the Claimant). If the value could be undetectably tampered with between its initial sending and final receipt, this specification will not receive a pass in a model check.

- Agreement(P, V, [xP])
- Agreement(P, V, [lP])
- Agreement(Pb, V, [xPb])
- Agreement(P, V, [lPb])
- Agreement(V, C, [dV])

More general authentication is provided by the use of digital signatures on messages being exchanged between participants. The presence of a digital signature on a message prevents any intruder from successfully replacing it with a false message or tampering with its contents, without the appropriate key. Authentication is implicitly modelled in this manner, rather than explicitly as a Casper specification. We assume that the cryptographic scheme and key sizes being employed are of reasonable strength, and therefore a key cannot be discovered without the aid of that key's owner. As one of the basic assumptions of this work is the use of tamper-resistant units for the protection of cryptographic material, the owner of a key is prevented from directly divulging his keys to other participants. The use of alternative approaches by the key's owner, e.g. chosen plaintext analysis, may lead to the breaking of its keys, but this is a common issue in the area of cryptography.

5.5.3.2 Simplifying the SLVPGP's First Extension

With the protocol translated from abstract protocol notation into Casper, the complexity of the model may now be reduced using some basic simplifications. These simplifications merely remove some of the more complex aspects which have no bearing on the security properties being checked. Each of the simplifications carried out are outlined below, with the resulting alterations illustrated by updated Casper model descriptions.

The coalescing transformation

The first simplification applied to extension one of the SLVPGP is the coalescing of atoms into a single atom. In this case, three transformations are performed - the location variable *xP* (resp. *xPb*) and the latency variable *lP* (resp. *lPb*) are combined into a single proof value, *proof* (resp. *proofB*). The nonces *nP* (resp. *nPb)* and *n1P* (resp. *n1Pb)* are combined into *nP*. Finally, the values *kP* (resp. *kPb)* and *hP* (resp. *hPb)* are combined into *hP*. This reduces the complexity of the protocol steps through decreasing the number of variables, while retaining the correctness of the model.

0. -> C: xC

1. C -> V: {C, xC}{SKAgent(C)}

2. V -> C: {P, Pb}{SKVerifier(V)}

3. C -> P: {P, C, {**nP**}{PKProver(P)}}{SKAgent(C)}

4. C -> Pb: {Pb, C, {**nPb**}{PKProver(Pb)}}{SKAgent(C)}

5. P -> C: **hP**, n2P

6. Pb -> C: **hPb**, n2Pb

7. C -> P: h1P, {n2P}{SKAgent(C)}

8. C -> Pb: h1Pb, {n2Pb}{SKAgent(C)}

9. P -> C: {P, tP, **proof,** C}{SKProver(P)} % mP

10. Pb -> C: {Pb, tPb, **proofB,** C}{SKProver(Pb)} % mPb

11. C -> V: {C, tC, xC}{SKAgent(C)}, {mP % {P, tP, **proof,** C}{SKProver(P)},
mPb % {Pb, tPb, **proofB**, C}{SKProver(Pb)}}{SKAgent(C)}

12. V -> C: {dV, xC, C, tV}{SKVerifier(V)}

Unlike the transformation outlined by Ryan and Schneider, when transforming the location and latency atoms, the pair is not replaced with the first atom within the pair. Instead, it is replaced with a single combination atom. This combination atom is merely a renaming of the atom employed in Ryan and Schneider's simplification and is used for clarity. The security of the transformation holds as it meets the condition that the removed atom can be deduced based upon the replacement and the intruder's initial knowledge. The security specifications are altered to use the combination atom as a representation of the pair. If the intruder discovers the value of *proof* (resp. *proofB*), it has knowledge of both elements within that pair. This statement is also true for the coalesced nonce pairs, which follow the method outlined by Ryan and Schneider and are replaced with an element of the pair.

Message splitting

Although extension one's design has been greatly simplified, the final step remains an issue. Due to the Claimant forwarding on complete messages from its Proof Providers, the number of possible variable combinations causes the model to retain an unacceptable level of complexity. Therefore, message splitting is employed in an effort to reduce the complexity of the step.

0. -> C: xC

1. C -> V: {C, xC}{SKAgent(C)}

2. V -> C: {P, Pb}{SKVerifier(V)}

3. C -> P: {P, C, {nP}{PKProver(P)}}{SKAgent(C)}

4. C -> Pb: {Pb, C, {nPb}{PKProver(Pb)}}{SKAgent(C)}

5. P -> C: hP, n2P

6. Pb -> C: hPb, n2Pb

7. C -> P: h1P, {n2P}{SKAgent(C)}

8. C -> Pb: h1Pb, {n2Pb}{SKAgent(C)}

9. P -> C: {P, tP, proof, C}{SKProver(P)} % mP

10. Pb -> C: {Pb, tPb, proofB, C}{SKProver(Pb)} % mPb

11. C -> V: {C, tC, xC}{SKAgent(C)}

11(a). C -> V: {mP % {P, tP, proof, C}{SKProver(P)}}{SKAgent(C)}

11(b). C -> V: {mPb % {Pb, tPb, proofB, C}{SKProver(Pb)}}{SKAgent(C)}

12. V -> C: {dV, xC, C, tV}{SKVerifier(V)}

Rather than the Claimant transmitting all proof messages received by its Proof Providers in a single message, the design is restructured to allow each message to be forwarded individually. Although this increases the total number of messages produced within the model, the combined complexity of each individual message is vastly less than that of a single message composed of the concatenation of all proofs.

5.5.4 Formally Modelling Extension Two

In this section, we outline the process of translating the second extension to the SLVPGP from security notation to a model for use by FDR. This process is accomplished in two stages: The original design of the protocol is first translated from abstract protocol notation into Casper, with the extension's desired security properties modelled as Casper security specifications (Section 5.5.4.1). The Casper translation is then simplified to reduce its complexity (Section 5.5.4.2).

5.5.4.1 Modelling Extension Two - From Abstract Notation to Casper

As with the conversion of extension one from abstract protocol notation to a format suitable for model checking, the protocol description is translated into Casper prior to simplification. The Casper description of the second extension is more complex than that of the first extension, due to the inclusion of encryption in this version. The full Casper model of this extension is shown in Appendix A, with the protocol description shown below.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{P, Pb}{SKVerifier(V)}}{PKAgent(C)}

3. C -> P: {{P, C, nP, n1P}{SKAgent(C)}}{PKProver(P)}

4. C -> Pb: {{Pb, C, nPb, n1Pb}{SKAgent(C)}}{PKProver(Pb)}

5. P -> C: kP, hP, n2P

6. Pb -> C: kPb, hPb, n2Pb

7. C -> P: h1P, {n2P}{SKAgent(C)}

8. C -> Pb: h1Pb, {n2Pb}{SKAgent(C)}

9. P -> C: {{P, tP, xP, lP, C}{SKProver(P)} % mP}{PKAgent(C)}

10. Pb -> C: {{Pb, tPb, xPb, lPb, C}{SKProver(Pb)} % mPb}{PKAgent(C)}

11. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}, {{mP % {P, tP, xP, lP, C}
{SKProver(P)}, mPb % {Pb, tPb, xPb, lPb, C}{SKProver(Pb)}}
{SKAgent(C)}}{PKVerifier(V)}

12. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

Due to the increased number of security properties claimed by this extension, outlined in Section 4.4.2 of Chapter 4, the number of included specifications increases from those employed in extension one's script. A second security specification type is employed - StrongSecret - which allows Casper to verify the support for anonymity and confidentiality within the protocol. The failure of this security specification indicates that the supposedly secured value is leaked to the intruder at some point during a protocol run. We employ the StrongSecret specification rather than the Secret specification as the level of security being tested by the condition is much higher. If an intruder learns a value prior to the end of the protocol's run and interrupts the run, the flaw will be detected. This is particularly useful in the case of the properties modelled here as the values being protected are not only generated for use in a specific run and then discarded, but are pieces of long term information regarding the protocol's participants. Details regarding Casper's interpretation of the specifications employed in this model are outlined in Section 5.3.2.

The aim of the second extension to the protocol is to provide authentication, external anonymity and external confidentiality to the participants within the exchange being protected. These conditions build upon the authentication provided within the first protocol extension.

- Agreement(P, V, [proof])
- Agreement(Pb, V, [proofB])
- Agreement(V, C, [dV])

External anonymity is defined as a guarantee that the identities of all devices participating in the exchange are unknown to any external entities, i.e. those not participating in the exchange. This condition is modelled through specifying that these identities are secrets known only to devices participating in the exchange.

However, upon modelling these conditions, it was found that this is too general a guarantee. Although both the Claimant and the Verifier have knowledge of the identities of all devices participating in the exchange, the Proof Providers involved have no knowledge of any identities other than those of the Claimant and Verifier. Therefore the conditions modelled are as follows:

- StrongSecret(C, C, [V, P, Pb])
- StrongSecret(P, P, [V, C])
- StrongSecret(Pb, Pb, [V, C])

External confidentiality is defined as a guarantee that the location information regarding a device must remain secret from any devices external to the exchange. Similar to the modelling of external anonymity, this condition is modelled in Casper as a secret in which the location information is known only to the listed device within the exchange. As discovered with modelling external anonymity, this is too general a guarantee. The use of encryption on protocol messages prevents any devices other than those to whom the message is encrypted for from learning the contents. Therefore, the Claimant and Verifier know all location information passed within the system. However, the Proof Providers involved have knowledge only of their own location information and not those of any other Proof Providers involved. This is modelled as follows:

- StrongSecret(C, xC, [V])
- StrongSecret(P, proof, [V, C])
- StrongSecret(Pb, proofB, [V, C])

5.5.4.2 Simplifying the SLVPGP's Second Extension

In order to reduce the complexity of the second extension's model and therefore decrease the size of the resulting state space, two basic simplifications are applied. These simplifications do not infringe on the completeness of the protocol. They merely remove some of the more complex aspects which have no bearing on the security properties being checked. The simplifications carried out on this extension are outlined below, with alterations to the protocol illustrated by updated Casper descriptions.

The coalescing transformation

The first simplification applied to extension two of the SLVPGP is the coalescing of atoms into a single atom. Three transformations are performed: The location variable *xP* (resp. *xPb*) and the latency variable *lP* (resp. *lPb*) are combined into a single proof value, *proof* (resp. *proofB*). The nonces *nP* (resp. *nPb)* and *n1P* (resp. *n1Pb)* are combined into *nP*. Finally, the values *kP* (resp. *kPb)* and *hP* (resp. *hPb)* are combined into *hP*.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{P, Pb}{SKVerifier(V)}}{PKAgent(C)}

3. C -> P: {{P, C, **nP**}{SKAgent(C)}}{PKProver(P)}

4. C -> Pb: {{Pb, C, **nPb**}{SKAgent(C)}}{PKProver(Pb)}

5. P -> C: **hP**, n2P

6. Pb -> C: **hPb**, n2Pb

7. C -> P: h1P, {n2P}{SKAgent(C)}

8. C -> Pb: h1Pb, {n2Pb}{SKAgent(C)}

9. P -> C: {{P, tP, **proof**, C}{SKProver(P)} % mP}{PKAgent(C)}

10. Pb -> C: {{Pb, tPb, **proofB**, C}{SKProver(Pb)} % mPb}{PKAgent(C)}

11. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}, {{mP % {P, tP, **proof,** C}{SKProver(P)}, mPb % {Pb, tPb, **proofB**, C}{SKProver(Pb)}}{SKAgent(C)}}{PKVerifier(V)}

12. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

Message splitting

With the application of the coalescing transformation, the complexity of extension two's design is slightly reduced. However, as discussed previously in relation to extension one, the complexity of the Claimant's message forwarding the gathered proofs to the Verifier is a crucial issue. In order to address this issue, we employ a structural transformation to the problematic step, dissecting it into multiple individual messages and sending these in place of a single concatenated message.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{P, Pb}{SKVerifier(V)}}{PKAgent(C)}

3. C -> P: {{P, C, nP}{SKAgent(C)}}{PKProver(P)}

4. C -> Pb: {{Pb, C, nPb}{SKAgent(C)}}{PKProver(Pb)}

5. P -> C: hP, n2P

6. Pb -> C: hPb, n2Pb

7. C -> P: h1P, {n2P}{SKAgent(C)}

8. C -> Pb: h1Pb, {n2Pb}{SKAgent(C)}

9. P -> C: {{P, tP, proof, C}{SKProver(P)} % mP}{PKAgent(C)}

10. Pb -> C: {{Pb, tPb, proofB, C}{SKProver(Pb)} % mPb}{PKAgent(C)}

11. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}

11(a). C -> V: {{mP % {P, tP, proof, C}{SKProver(P)}}{SKAgent(C)}} {PKVerifier(V)}

11(b). C -> V: {{mPb % {Pb, tPb, proofB, C}{SKProver(Pb)}}{SKAgent(C)}} {PKVerifier(V)}

12. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

As in the case of extension one, the Claimant sends a message containing its own information before proceeding to send each of the proof messages gathered to the Verifier seperately. Although this message splitting does alter the structure of the protocol, it does not alter its functionality. As the order of the messages being sent is preserved, the signals generated by the model remain in their original order and no values within the messages are altered in any way. Therefore, the traces produced remain faithful to those of the original model and the simplification does not remove any possible attacks.

5.5.5 Formally Modelling Extension Three

In this section, we outline the process of translating the third extension to the SLVPGP from security notation to a model for use by FDR. This process is accomplished in two stages; the original design of the protocol is first translated from abstract protocol notation into Casper, with the extension's desired security properties modelled as Casper security specifications (Section 5.5.5.1). It is then simplified to reduce its complexity (Section 5.5.5.2). However, where the earlier protocol extensions could be directly translated into

Casper, the third extension to the protocol requires an additional alteration, due to its reliance on broadcasting to support anonymity.

5.5.5.1 Modelling Extension Three - From Abstract Notation to Casper

As with the conversion of the previous extensions, the protocol description is translated into Casper prior to simplification. The Casper description of the third extension is the most complex of the three, due to the inability of the participants to see the contents of the encrypted messages they receive. This inability is a by-product of retaining anonymity and confidentiality, as encryption is heavily employed to protect the private data of all devices involved. In this section, we model the third extension in Casper and detail the security specifications included in this description, used to model the extension's claimed security properties. We then discuss the need for an additional adaptation to the protocol's design in order to create a fully functional Casper model - the addition of an Oracle.

Modelling extension three in Casper

The full Casper model of this extension is shown in Appendix A, with the protocol description shown below.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)} % mP,
{{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)} % mPb}{SKVerifier(V)},
{{nP, n1P, kCP, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKAgent(C)}

3. C -> P: mP % {{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)},
{C}{PKVerifier(V)} % mCP

4. C -> Pb: mPb % {{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)},
{C}{PKVerifier(V)} % mCPb

5. P -> C: kP, hP, n2P

6. Pb -> C: kPb, hPb, n2Pb

7. C -> P: h1P, {n2P}{SKAgent(C)} % mCPN

8. C -> Pb: h1Pb, {n2Pb}{SKAgent(C)} % mCPbN

9. P -> C: h(kCP), {nP, {{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, xP, lP,
mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}{PKVerifier(V)} % mPV}{kCP}

10. Pb -> C: h(kCPb), {nPb, {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, xPb, lPb, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}} {PKVerifier(V)} % mPbV}{kCPb}

11. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}, {mPV % ({{P, mCPN1 % {n2P} {SKAgent(C)}, tP, xP, lP, mCP1 % {C}{PKVerifier(V)}}{SKProver(P)}} {PKVerifier(V)}), mPbV %, ({{Pb, mCPbN1 % {n2P}{SKAgent(C)}, tPb, xPb, lPb, mCPb1 % {C}{PKVerifier(V)}}{SKProver(Pb)}}{PKVerifier(V)})}{SKAgent(C)}

12. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

However, while the previous extensions to the protocol could be directly translated following the simplification process, the increased level of anonymity provided by the third extension requires that the design undergo a special adaptation prior to its translation into Casper notation. This is due to Casper's requirement that a device know the identity of any message's recipient. Therefore, the Casper description shown above will not function. In order to address this issue, we must include an additional device to model the underlying network over which the system communicates. The addition of this "Oracle" process is discussed in the next section.

Addressing the issue of broadcasting - the introduction of an Oracle

In practice, the Claimant is capable of broadcasting messages to the area surrounding it, removing the need for knowledge of the identities of the devices receiving them. However, this functionality is not modelled directly in Casper. An agent must have knowledge of the identity of its intended message recipient in order to communicate with them. In order to address this issue, a new device is added to the exchange, which models an Oracle or all-knowing entity. This Oracle process receives the message to be sent to be specific devices in the area and has knowledge of the identities of these devices.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)} % mP, {{Pb, nPb, n1Pb, kCPb} {SKVerifier(V)}}{PKProver(Pb)} % mPb}{SKVerifier(V)}, {{nP, n1P, kCP, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKAgent(C)}

3. C -> O: mP % ({{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)} % mP1), {C}{PKVerifier(V)} % mCP

3(a). O -> P: mP1 % {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)}, mCP % ({C}{PKVerifier(V)} % mCP1)

4. C -> O: mPb % ({Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)} % mPb1), {C}{PKVerifier(V)} % mCPb

4(a). O -> Pb: mPb1 % {{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)}, mCPb % ({C}{PKVerifier(V)} % mCPb1)

5. P -> O: kP, hP, n2P

5(a). O -> C: kP, hP, n2P

6. Pb -> O: kPb, hPb, n2Pb

6(a). O -> C: kPb, hPb, n2Pb

7. C -> O: h1P, {n2P}{SKAgent(C)} % mCPN

7(a). O -> P: h1P, mCPN % ({n2P}{SKAgent(C)} % mCPN1)

8. C -> O: h1Pb, {n2Pb}{SKAgent(C)} % mCPbN

8(a). O -> Pb: h1Pb, mCPbN % ({n2Pb}{SKAgent(C)} % mCPbN1)

9 P -> O: h(kCP) % mCPh, {nP, {{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, xP, lP, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}} {PKVerifier(V)} % mPV}{kCP} % mCPk

9(a). O -> C: mCPh % h(kCP), mCPk % {nP, mPV % ({{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, xP, lP, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}} {PKVerifier(V)} % mPV1)}{kCP}

10. Pb -> O: h(kCPb) % mCPbh, {nPb, {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, xPb, lPb, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}} {PKVerifier(V)} % mPbV}{kCPb} % mCPbk

10(a). O -> C: mCPbh % h(kCP), mCPbk % {nPb, mPbV % {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, xPb, lPb, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}{PKVerifier(V)} % mPbV1}{kCPb}

11. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}, {mPV1 % ({{P, mCPN1 % {n2P}{SKAgent(C)}, tP, xP, lP, mCP1 % {C}{PKVerifier(V)}{SKProver(P)}} {PKVerifier(V)}), mPbV1 % ({{Pb, mCPbN1 % {n2P}{SKAgent(C)}, tPb, xPb, lPb, mCPb1 % {C}{PKVerifier(V)}}{SKProver(Pb)}}{PKVerifier(V)})}{SKAgent(C)}

12. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

However, although a new process has been introduced to the model, the underlying system being modelled remains the same. The Oracle process is an artefact of employing the Casper language, but does not impact the security of the system being modelled. This is because the Oracle process does not alter or create messages within the system, but merely acts as a forwarder. In essence, the Oracle process represents the real world technology to which a device sends its message for broadcasting. It receives a message and transmits it to a final destination, without performing any action upon the message.

In the Casper model of this extension, the Oracle receives a message from one device and transmits the message to the correct recipient. Whilst this does increase the complexity of the model to be checked, due to the creation of a new agent and the addition of another process, it allows the condition of anonymity to be met. Additionally, as no alteration is performed to the message, no extra security is added to the content and no new information is made available to the intruder or any other device, the model's integrity remains intact. Therefore, as with extensions one and two, the verification results for this model are representative of the underlying protocol extension upon which it is based.

Modelling the security specifications for extension three

The aim of the final SLVPGP extension is to extend the security properties of confidentiality and anonymity to a complete level, while retaining the property of authentication. The security properties of extension three are discussed in Section 4.4.3 of Chapter 4, with details regarding Casper's interpretation of the specifications employed in this model outlined in Section 5.3.2. As in the first extension of the protocol, authentication of proof content is achieved through the need for the Verifier and each Proof Provider to agree on the values for the corresponding proof, while authentication of the Verifier's verdict is confirmed through a similar agreement between the Verifier and the Claimant.

- Agreement(P, V, [proof])
- Agreement(Pb, V, [proofB])
- Agreement(V, C, [dV])

The third extension introduces another process into the protocol, the Oracle, in order to model the functionality of broadcasting a message to an unknown recipient. This allows

devices to pass messages without knowing the intended recipient's identity, thus preserving anonymity. While the previous extension provided external anonymity, this final extension provides an even greater level of security. The level of anonymity provided is now complete, rather than external. The StrongSecret specification is again employed to model this condition. However, this time the only devices with knowledge regarding the data being kept secret are the Oracle, the Verifier and the device to which the data pertains. This differs from the anonymity specifications employed in extension two of the protocol, where the Claimant also had knowledge of the identities of all devices involved.

- StrongSecret(C, C, [O, V])
- StrongSecret(P, P, [O, V])
- StrongSecret(Pb, Pb, [O, V])

Similar to the increase in security regarding anonymity, the third SLVPGP extension increases the security property regarding location information from external confidentiality to complete confidentiality. This is also modelled using the StrongSecret specification.

- StrongSecret(C, xC, [V])
- StrongSecret(P, proof, [V])
- StrongSecret(Pb, proofB, [V])

In this case, the only devices allowed to be in possession of information regarding location are the Verifier (the all-knowing entity within the system) and the device to whom the location information pertains.

5.5.5.2 Simplifying the SLVPGP's Third Extension

The SLVPGP is at its most complex after the application of the third security extension, particularly with the inclusion of an Oracle to facilitate the modelling of broadcasting. The model representing this version of the protocol generates an extensive state space to be checked by FDR. In order to reduce this state space to a more manageable size, a number of simplifications are applied. As with those mentioned previously, these simplifications

do not infringe on the correctness of the protocol. We outline below each of the simplifications carried out on the protocol, illustrating the alterations to the protocol with the corresponding Casper descriptions.

The coalescing transformation

The first simplification applied to final extension of the SLVPGP is the coalescing of atoms into a single atom. The location variable *xP* (resp. *xPb*) and the latency variable *lP* (resp. *lPb*) are combined into a single proof value, *proof* (resp. *proofB*). Finally, the values *kP* (resp. *kPb)* and *hP* (resp. *hPb)* are combined into *hP*. Notably, we do not combine the nonces *nP* (resp. *nPb)* and *n1P* (resp. *n1Pb)* into *nP*, as *nP* (resp. *nPb*) also appears alone within the protocol.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)} % mP,
{{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)} % mPb}{SKVerifier(V)},
{{nP, n1P, kCP, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKAgent(C)}

3. C -> O: mP % ({{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)} % mP1),
{C}{PKVerifier(V)} % mCP

3a. O -> P: mP1 % {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)},
mCP % ({C}{PKVerifier(V)} % mCP1)

4. C -> Pb: mPb % ({Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)} % mPb1),
{C}{PKVerifier(V)} % mCPb

4a. O -> Pb: mPb1 % {{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)},
mCPb % ({C}{PKVerifier(V)} % mCPb1)

5. P -> O: **hP**, n2P

5a. O -> C: **hP**, n2P

6. Pb -> O: **hPb**, n2Pb

6a. O -> C: **hPb**, n2Pb

7. C -> O: h1P, {n2P}{SKAgent(C)} % mCPN

7a. O -> P: h1P, mCPN % ({n2P}{SKAgent(C)} % mCPN1)

8. C -> O: h1Pb, {n2Pb}{SKAgent(C)} % mCPbN

8a. O -> Pb: h1Pb, mCPbN % ({n2Pb}{SKAgent(C)} % mCPbN1)

9. P -> O: h(kCP) % mCPh, {nP, {{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, **proof**, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}{PKVerifier(V)} % mPV}{kCP} % mCPk

9a. O -> C: mCPh % h(kCP), mCPk % {nP, mPV % ({{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, **proof**, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}{PKVerifier(V)} % mPV1)}{kCP}

10. Pb -> O: h(kCPb) % mCPbh, {nPb, {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, **proofB**, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}{PKVerifier(V)} % mPbV}{kCPb} % mCPbk

10a. O -> C: mCPbh % h(kCP), mCPbk % {nPb, mPbV % {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, **proofB**, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}{PKVerifier(V)} % mPbV1}{kCPb}

11. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}, {mPV1 % ({{P, mCPN1 % {n2P}{SKAgent(C)}, tP, **proof**, mCP1 % {C}{PKVerifier(V)}{SKProver(P)}}{PKVerifier(V)}), mPbV1 % ({{Pb, mCPbN1 % {n2P}{SKAgent(C)}, tPb, **proofB**, mCPb1 %{C}{PKVerifier(V)}}{SKProver(Pb)}}{PKVerifier(V)})}{SKAgent(C)}

12. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

Message splitting (A)

In order to facilitate the employment of the redirection transformation on the structure of the modelled protocol, the messages being redirected must first be broken into their individual components. These include the initialisation messages sent to the Claimant for the Proof Providers and the additional information sent by the Claimant to each Proof Provider when forwarding these messages.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. **V -> C: {{nP, n1P, kCP, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKAgent(C)}**

3. **V -> C: {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)} % mP}{SKVerifier(V)}**

4. **V -> C: {{{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)} % mPb}{SKVerifier(V)}**

5. C -> O: mP % ({{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)} % mP1),

5a. O -> P: mP1 % {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)}

6. C -> O: {C}{PKVerifier(V)} % mCP

6a. O -> P: mCP % ({C}{PKVerifier(V)} % mCP1)

7. C -> Pb: mPb % ({Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)} % mPb1)

7a. O -> Pb: mPb1 % {{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)}

8. C -> O: {C}{PKVerifier(V)} % mCPb

8a. O -> Pb: mCPb % ({C}{PKVerifier(V)} % mCPb1)

9. P -> O: hP, n2P

9a. O -> C: hP, n2P

10. Pb -> O: hPb, n2Pb

10a. O -> C: hPb, n2Pb

11. C -> O: h1P, {n2P}{SKAgent(C)} % mCPN

11a. O -> P: h1P, mCPN % ({n2P}{SKAgent(C)} % mCPN1)

12. C -> O: h1Pb, {n2Pb}{SKAgent(C)} % mCPbN

12a. O -> Pb: h1Pb, mCPbN % ({n2Pb}{SKAgent(C)} % mCPbN1)

13. P -> O: h(kCP) % mCPh, {nP, {{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}{PKVerifier(V)} % mPV}{kCP} % mCPk

13a. O -> C: mCPh % h(kCP), mCPk % {nP, mPV % ({{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}{PKVerifier(V)} % mPV1)}{kCP}

14. Pb -> O: h(kCPb) % mCPbh, {nPb, {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}{PKVerifier(V)} % mPbV}{kCPb} % mCPbk

14a. O -> C: mCPbh % h(kCP), mCPbk % {nPb, mPbV % {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}{PKVerifier(V)} % mPbV1}{kCPb}

15. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}, {mPV1 % ({{P, mCPN1 % {n2P}{SKAgent(C)}, tP, proof, mCP1 % {C}{PKVerifier(V)}{SKProver(P)}}{PKVerifier(V)}), mPbV1 % ({{Pb, mCPbN1 % {n2P}{SKAgent(C)}, tPb, proofB, mCPb1 %{C}{PKVerifier(V)}}{SKProver(Pb)}}{PKVerifier(V)})}{SKAgent(C)}

16. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

Message splitting in this manner clearly distinguishes the different message components and their end destinations, allowing for an easier understanding of the process behind the protocol's simplification. As with all other message splitting transformations described within this work, the order of the messages is preserved and the values within the messages are not altered. This ensures the traces of the transformed protocol remain faithful to the protocol's original model and any possible attacks on said model are also applicable to the transformed version.

Redirection of the Proof Provider identity messages

The next simplification applied is the redirection of the proof messages from the Proof Providers. In the original design, the initialisation message for each Proof Provider is conveyed to that Proof Provider via the Claimant. This is seen more clearly after the application of message splitting, shown above. The Claimant acts as a transparent relay, passing on the message without altering it in any way. It does not provide any additional security, nor does its removal impact the security being checked. For this reason, employing a redirection simplification does not alter the security of the protocol, but merely reduces the model's complexity. In addition to this, message redirection is an attack-preserving simplification, as outlined in Section 5.4.3.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{nP, n1P, kCP, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKAgent(C)}

3. **V -> P: {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)}}{SKVerifier(V)}**

4. **V -> Pb: {{{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)}}{SKVerifier(V)}**

5. C -> O: {C}{PKVerifier(V)} % mCP

5a. O -> P: mCP % ({C}{PKVerifier(V)} % mCP1)

6. C -> Pb: {C}{PKVerifier(V)} % mCPb

6a. O -> Pb: mCPb % ({C}{PKVerifier(V)} % mCPb1)

7. P -> O: hP, n2P

7a. O -> C: hP, n2P

8. Pb -> O: hPb, n2Pb

8a. O -> C: hPb, n2Pb

9. C -> O: h1P, {n2P}{SKAgent(C)} % mCPN

9a. O -> P: h1P, mCPN % ({n2P}{SKAgent(C)} % mCPN1)

10. C -> O: h1Pb, {n2Pb}{SKAgent(C)} % mCPbN

10a. O -> Pb: h1Pb, mCPbN % ({n2Pb}{SKAgent(C)} % mCPbN1)

11. P -> O: h(kCP) % mCPh, {nP, {{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}{PKVerifier(V)} % mPV}{kCP} % mCPk

11a. O -> C: mCPh % h(kCP), mCPk % {nP, mPV % ({{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}} {PKVerifier(V)} % mPV1)}{kCP}

12. Pb -> O: h(kCPb) % mCPbh, {nPb, {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}{PKVerifier(V)} % mPbV}{kCPb} % mCPbk

12a. O -> C: mCPbh % h(kCP), mCPbk % {nPb, mPbV % {{Pb, mCPbN % ({n2P} {SKAgent(C)} % mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)} {SKProver(Pb)}}{PKVerifier(V)} % mPbV1}{kCPb}

13. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}, {mPV1 % ({{P, mCPN1 % {n2P} {SKAgent(C)},tP, proof, mCP1 % {C}{PKVerifier(V)}{SKProver(P)}}{PKVerifier(V)}), mPbV1 % ({{Pb, mCPbN1 % {n2P}{SKAgent(C)}, tPb, proofB, mCPb1 % {C}{PKVerifier(V)}}{SKProver(Pb)}} {PKVerifier(V)})}{SKAgent(C)}

14. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

The signals produced by the transformed model shown above remain in the same order as those produced by the original model's design The values contained within the messages are committed to at the same point during the exchange, therefore the transformed model meets the requirements for a valid redirection transformation as defined by Ryan and Schneider. Any attacks present on the original design will be preserved in the transformed model and the results of any specification check completed on the transformed model can be projected onto the original.

Message splitting (B)

Although the application of the above simplifications does greatly reduce the complexity of the protocol to be modelled, the message in which the Claimant's gathered proofs are forwarded to the Verifier remains an issue. In order to address the level of complexity within this message, we again employ the message splitting transformation. In this instance, the the Claimant's proof forwarding message is dissected into several individual messages and these are sent in place of a single concatenated message.

0. -> C: xC

1. C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}

2. V -> C: {{nP, n1P, kCP, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKAgent(C)}

3. V -> P: {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)}}{SKVerifier(V)}

4. V -> Pb: {{{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)}}{SKVerifier(V)}

5. C -> O: {C}{PKVerifier(V)} % mCP

5a. O -> P: mCP % ({C}{PKVerifier(V)} % mCP1)

6. C -> Pb: {C}{PKVerifier(V)} % mCPb

6a. O -> Pb: mCPb % ({C}{PKVerifier(V)} % mCPb1)

7. P -> O: hP, n2P

7a. O -> C: hP, n2P

8. Pb -> O: hPb, n2Pb

8a. O -> C: hPb, n2Pb

9. C -> O: h1P, {n2P}{SKAgent(C)} % mCPN

9a. O -> P: h1P, mCPN % ({n2P}{SKAgent(C)} % mCPN1)

10. C -> O: h1Pb, {n2Pb}{SKAgent(C)} % mCPbN

10a. O -> Pb: h1Pb, mCPbN % ({n2Pb}{SKAgent(C)} % mCPbN1)

11. P -> O: h(kCP) % mCPh, {nP, {{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}{PKVerifier(V)} % mPV}{kCP} % mCPk

11a. O -> C: mCPh % h(kCP), mCPk % {nP, mPV % ({{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}} {PKVerifier(V)} % mPV1)}{kCP}

12. Pb -> O: h(kCPb) % mCPbh, {nPb, {{Pb, mCPbN % ({n2P}{SKAgent(C)} % mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}{PKVerifier(V)} % mPbV}{kCPb} % mCPbk

12a. O -> C: mCPbh % h(kCP), mCPbk % {nPb, mPbV % {{Pb, mCPbN % ({n2P} {SKAgent(C)} % mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)} {SKProver(Pb)}}{PKVerifier(V)} % mPbV1}{kCPb}

13. C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}

13a. C -> V: {mPV1 % ({{P, mCPN1 % {n2P}{SKAgent(C)}, tP, proof, mCP1 % {C}{PKVerifier(V)}{SKProver(P)}} {PKVerifier(V)})}{SKAgent(C)}

13b. C -> V: {mPbV1 % ({{Pb, mCPbN1 % {n2P}{SKAgent(C)}, tPb, proofB, mCPb1 % {C}{PKVerifier(V)}}{SKProver(Pb)}} {PKVerifier(V)})}{SKAgent(C)}

14. V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}

This transformation could be taken a step further by also splitting out the Claimant's encrypted identity, contained within each proof. However, it was discovered that rather than simplifying the design, employing this level of splitting increases the protocol's complexity, resulting in an overly complex design and a state space explosion. This is because the complexity cost in creating an extra message with that level of encryption and digital signing is far greater than including it within the original message. Upon removing this additional split in the message, the generated state space becomes tractable and the model check can be completed.

5.6 Results of Model Checking

After the protocol designs have been simplified to a reduced level of complexity and translated into Casper notation, the resulting descriptions are compiled and CSP scripts describing each design are generated. These CSP scripts can then be fed to the FDR model checker and their security specifications validated or failed. We have also created seven alternate models for each extension which, in addition to the knowledge already possessed, grant the intruder knowledge of at least one participating device's private cryptographic key. These models allow us to investigate the impact of the compromising of keys on the protocol's security. The models created include: **Known C**, where the Claimant's keys are compromised; **Known P/Pb**, where a Proof Provider's keys are compromised; **Known CP/CPB/CPPB**, where both the Claimant's keys and either one or both of the Proof Providers keys are compromised; and finally **Known PPb**, where the keys of all Proof

Known	Agreement(Proof)	Agreement(ProofB)	Agreement(D_V)
None	√	√	√
C	√	√	√
CP	Fail	√	√
CPb	√	Fail	√
CPPb	Fail	Fail	√
P	Fail	√	√
Pb	√	Fail	√
PPb	Fail	Fail	√

Table 5.1: Verification Results for Extension One of the SLVPGP

Providers involved are compromised. While we do not envision that these cryptographic keys can be compromised in this manner, due to the system's requirement that all keys be stored within a tamper-resistant unit, we wished to investigate the impact of such an event occurring on the security of the system. We do not model a scenario where the Verifier's keys are compromised, as the Verifier is a trusted device, and should that set of keys be compromised, the system would cease to be viable.

It should be stated that each of the uncompromised extension models received successful verdicts, indicating that the protocols do not contain security vulnerabilities in their described environments. The only model checking failures were those models in which keys were compromised, with not all of these failing and no check failing on all specifications.

5.6.1 Results for Extension One

As mentioned previously, the SLVPGP's first extension is the least secure, providing only the minimum security: a single security property (authentication), which is split into three security specifications; one investigating the security of the Verifier's verdict and one for each Proof Provider modelled. These Agreement specifications check that the transmitted value of proof, proofB and D_V cannot be manipulated during the course of the protocol exchange. As shown in Table 5.1, the verification results received on the verdict specification is successful for all compromised keys. This is due to the value being digitally signed using the Verifier's private key, a key unknown to all devices other than the Verifier. As discussed previously, we do not model the compromising of the Verifier's keys, as the Verifier is a trusted entity.

Known	A(proof)	A(proofB)	A(D_V)	SS(C)	SS(P)	SS(Pb)	SS(X_C)	SS(proof)	SS(proofB)
None	√	√	√	√	√	√	√	√	√
C	√	√	√	Fail	Fail	Fail	Fail	Fail	Fail
CP	Fail	√	√	Fail	Fail	Fail	Fail	Fail	Fail
CPb	√	Fail	√	Fail	Fail	Fail	Fail	Fail	Fail
CPPb	Fail	Fail	√	Fail	Fail	Fail	Fail	Fail	Fail
P	√	√	√	Fail	Fail	√	√	√	√
Pb	√	√	√	Fail	√	Fail	√	√	√
PPb	√	√	√	Fail	Fail	Fail	√	√	√

Table 5.2: Verification Results for Extension Two of the SLVPGP

The verification results received on both Proof Provider specifications by the uncompromised and Known C models are successful. This indicates that the compromising of a Claimant's keys does not impact the extension's support of the authentication property. These results are expected, as within this extension, the Claimant is not involved in protecting the proofs gathered. However, the results received by the remaining six models reflect an obvious insecurity within the extension. These results illustrate that any compromise of a Proof Provider's key leads to that Proof Provider's proof being left vulnerable to tampering. This is because the only protection applied to this value is the digital signature of the Proof Provider. This vulnerability is addressed in extensions two and three through the inclusion of encryption.

5.6.2 Results for Extension Two

Building on the first extension's security property of authentication, extension two increases the number of security properties supported to include anonymity and confidentiality of location. These properties are modelled using the StrongSecret (SS) security specification. The number of specifications is increased to include a StrongSecret specification for each of the values being protected, i.e. the identities of each of the participants, the Claimant's location and the proof values of both Proof Providers modelled. These specifications check that the values being protected are not leaked at any point during the exchange, which would allow an intruder to gain knowledge of them.

As shown in Table 5.2, the verification results for the uncompromised model of the extension are all successful. Additionally, as in the verification results for extension one, authentication on the value of the Verifier's final verdict is supported in all situations. However, when investigating the security of all other specifications, the compromising of

Known	A(proof)	A(proofB)	A(D_V)	SS(C)	SS(P)	SS(Pb)	SS(X_C)	SS(proof)	SS(proofB)
None	√	√	√	√	√	√	√	√	√
C	√	√	√	Fail	√	√	Fail	√	√
CP	√	√	√	Fail	Fail	√	Fail	√	√
CPb	√	√	√	Fail	√	Fail	Fail	√	√
CPPb	√	√	√	Fail	Fail	Fail	Fail	√	√
P	√	√	√	√	Fail	√	√	√	√
Pb	√	√	√	√	√	Fail	√	√	√
PPb	√	√	√	√	Fail	Fail	√	√	√

Table 5.3: Verification Results for Extension Three of the SLVPGP

the Claimant's keys has an enormous impact. Where the Claimant's keys alone are compromised, all anonymity specifications fail, along with all confidentiality specifications. This is an expected outcome, as the Claimant is used within this extension of the protocol as a hub, through which all messages pass and with all messages encrypted for the Claimant. This provides any device in possession of the Claimant's keys with the values all supposedly protected information. Authentication, unlike anonymity and confidentiality, is still believed to be supported, with the authentication results for both Proof Provider values receiving positive results in addition to those received by the Verifier's verdict specifications.

However, due to the presence of digital signatures on the proof values from each Proof Provider, though an intruder may gain knowledge of their values, but it cannot amend them. This is because it does not possess the Proof Provider's keys to forge a new signature. Where a Proof Provider's key is compromised, either alone or in conjunction with the Claimant's, the compromised Proof Provider's agreement specification no longer passes the check. Similarly, the anonymity property is no longer supported, as the identities of the Claimant and any compromised Proof Provider are leaked. However, unlike where the Claimant's key is compromised, the compromising of only a Proof Provider's key does not cause support for the confidentiality property to be lost, as the values protected to give confidentiality are never encrypted for those keys.

5.6.3 Results for Extension Three

The third extension to the protocol upgrades the level of anonymity and confidentiality being tested from external to complete through amending the security specifications employed in extension two's verification. In the amended StrongSecret (SS) specifications, the number of devices allowed knowledge of the values being protected is greatly reduced to

include only the participant the value concerns, the Verifier and in the case of anonymity StrongSecret specifications, the Oracle. However, similar to the previous two extensions, authentication is modelled as three Agreement specifications, one concerning the Verdict supplied by the Verifier and one for each of the proofs transmitted within the model.

The improvement in verification results for this extension compared to those of the previous extensions demonstrates the increased level of security employed. As shown in Table 5.3, the verification results for the uncompromised model are again all successful. There is a vast decrease in the number of fails received where the keys of the Claimant are compromised in comparison to the verification results for the second extension, with only the Claimant's identity and location consistantly vulnerable in this situation. This leakage occurs when the Verifier sends its verification message to the Claimant. The only other vulnerable information within the protocol is the identity of a compromised Proof Provider. This vulnerability is due to the identity's presence within the only message containing sensitive information encrypted using a key other than the Verifier's.

5.7 Verifying the Security of Broadcasting Messages

In practice, the third extension to the SLVPGP makes use of broadcasting to protect the anonymity of the participants. However, one concern when using this approach is the possibility that supplying devices other than the intended recipient with a message inadvertantly causes a security breach within a protocol. In order to confirm that this is not the case, we designed a trivial system in which broadcasting is used. The system was then model checked to verify its secrecy and authentication properties. The results of this process support the hypothesis that when messages are encrypted using uncompromised keys, their integrity is unaffected by broadcasting

5.7.1 System Description

1. S →A: $\{M_B\}_{K_B^+}, \{M_C\}_{K_C^+}$
2. A →B $\{M_B\}_{K_B^+}$
3. A →C: $\{M_C\}_{K_C^+}$

The basic design of the example system is composed of a Server device (S), which wishes to send a message to two devices, Bob and Celine. However, the Server has no method of sending the message directly to these devices. Bob and Celine are in the vicinity of a third device (Alice), which is in contact with the Server. In order to transmit the messages to Bob and Celine, the Server employs Alice as a proxy. However, Alice cannot know the identity of the devices she forwards the messages to. Therefore she must broadcast the messages to all devices in the vicinity. Broadcasting in a wireless network is defined as transmitting a single message to all devices within transmission range of the source device, without requiring knowledge of the addresses or identities of the recipients. It is accomplished through the use of the standard MAC broadcast address, FF:FF:FF:FF:FF:FF.

5.7.2 Casper Notation

In Casper, the act of broadcasting cannot be directly modelled, as each message must have a specific recipient and this recipient must be known to the sending agent (removing the ability to retain anonymity). Therefore another process known as the Oracle (O) is introduced. This alters the system's description to:

1. S →A: $\{M_B\}_{K_B^+}, \{M_C\}_{K_C^+}$
2. A →O: $\{M_B\}_{K_B^+}$
3. O→B: $\{M_B\}_{K_B^+}$
4. A →O: $\{M_C\}_{K_C^+}$
5. O→C: $\{M_C\}_{K_C^+}$

The primary function of the Oracle is to act as third party proxy, allowing Alice to transmit S's messages to Bob and Celine. The Oracle is modelled as a special type within the system, outside the control of any adversary and without impact on the system as a whole. Therefore, this adjustment does not impact the model checking results on the system. The casper translation of the protocol's security notation is shown below, with the complete script shown in Section A.2 of Appendix A.

1. -> S: mB, mC

2. S -> A: {mB}{PK(B)} % nMb, {mC}{PK(C)} % nMc

3. A -> O: nMb % ({mB}{PK(B)} % nMb1)

4. O -> B: nMb1 % {mB}{PK(B)}

5. A -> O: nMc % ({mC}{PK(C)} % nMc1)

6. O -> C: nMc1 % {mC}{PK(C)}

In the Casper model, the proxied messages are sent only to the intended destinations, with the intruder also receiving a copy of all messages. The intruder can therefore represent any device within range of Alice when she broadcasts the message being forwarded. If the intruder can successfully break any of the security conditions being checked, then broadcasting can be regarded as an insecure approach in this situation.

Security specifications

In order to confirm that a message cannot be undetectably interfered with by any intruder once sent by the Server device, an agreement specification is employed. This specification checks that the message committed to by the Server is the same message received by the intended recipient (either B or C) at the end of the protocol's run. If an intruder can undetectably insert their own value for this message, the system's security is flawed.

- Agreement(S, B, [mB])
- Agreement(S, C, [mC])

The system described above conforms to the agreement specifications. Therefore, broadcasting messages over an open network does not affect the authentication aspect of the messages assuming the messages are properly secured, i.e. that a message is digitally signed by its creator.

The second aspect of a message's security is that its contents remain secret. Broadcasting messages allows for all devices within range of the sender to receive a copy. Therefore it is vital that this secrecy is maintained even when the message is received by many devices and not just the intended recipient. In order to confirm that a message's secrecy is not affected by the act of broadcasting, secrecy specifications are employed for each of the messages investigated.

- Secret(S, mB, [B])
- Secret(S, mC, [C])

These specifications check that the value of the message in question is not leaked at any point during the protocol's run. If an intruder can obtain the value of the message, the property of secrecy does not hold and the protocol is flawed.

This system conforms to these specifications, as messages are encrypted to protect their contents. Therefore, the verification results suggest that broadcasting messages over an open network does not affect the secrecy of their contents. This is based on the assumption that the messages are properly secured.

5.7.3 Modelling Colluding and Compromised Agents

In the previous section, the implications of employing broadcasting within a system were outlined and its limitations discovered. However, those limitations are only proven to apply where an intruder has knowledge only of its own keys - i.e. no devices are compromised - and where no devices involved are colluding. In order to investigate the effects of compromised and colluding devices, multiple versions of the system described in Section 5.7.1 have been modelled. In each version, the intruder process' knowledge is amended. Each amendment represents either the compromising of a device or the collusion of a number of devices. The intruder's knowledge defined within the Casper script is altered to include information regarding the keys of one or more devices. The altered intruder knowledge definitions are shown in Section A.2.2 of Appendix A.

In the case of modelling a single compromised device, the intruder process is given knowledge of the public and private keys of this device. In addition to this, if the compromised device being modelled is a message end-recipient, i.e. a device receiving one of the messages being transmitted (e.g. devices B or C), the security specification regarding the secrecy of that device's message is removed. This is to allow the system to be fully checked for security flaws, rather than failing its model check on a specification that is intuitively known to fail. With this specification removed, the model check is successful, proving that the system's integrity remains intact for messages secured with unknown keys. However, if the specification is re-inserted, the check fails as the secrecy of this message is no longer provided through encryption.

This approach is also employed in the modelling of colluding devices. We consider two collusion scenarios for modelling, where device A colludes with end-recipient device B (collusion AB) and where device A colludes with end-recipient device C (collusion AC). We do not investigate the impact of devices B and C colluding as the secrecy of both messages being transmitted would immediately be compromised in this scenario. Therefore, any checks on the secrecy of those messages would fail.

Similar to the checks performed on the compromise of a single device, the various checks on the colluding device combinations are all successful where the model does not include a specification regarding the message intended for the end-recipient involved. When a specification of this nature is included, the check fails due to an obvious flaw in the security of the system, i.e. that the intruder can gain access to messages encrypted for access using the end-recipient's secret key. As this situation is designed to model the collusion of an end-recipient and the intruder, an unsuccessful check is the expected outcome. FDR's returning of positive verdicts on the remaining checks supports the hypothesis that broadcasting a message does not impair the security of the message or the system. However, the receipt of negative verdicts on those checks where keys have been compromised shows that this hypothesis only holds assuming that the message being transmitted is secured with uncompromised encryption.

5.8 Summary

In this chapter, we discussed the approach used to formally verify the Secure Location Verification Proof Gathering Protocol through model checking. We have outlined the steps involved in readying the protocol for verification, from reducing the complexity of the protocol using safe simplifying transformations to translating the simplified design into Casper notation. The verification of the CSP models generated from these Casper notations indicates the absence of flaws with the security properties claimed for each extension within the systems described. This supports our conclusion that the SLVPGP is secure and provides these security properties.

In order to further analyse the system's security, we have carried out additional verifications where the integrity of various keys has been compromised. This analysis indicates that the impact of compromising a key's integrity is at a minimum in the final extension,

due to the layering of encryption and digital signatures. This extension was also the only version of the SLVPGP not to receive a fail on either agreement during the compromised-key analysis, indicating that the described system is not vulnerable to proof tampering. The analysis also indicates the vulnerability of the middle extension to manipulation if the integrity of the Claimant's key pair is compromised, due to the use of the Claimant as a central entity in that extension's design. However, as discussed in Chapters 3 and 4, the storing of cryptographic keys in tamper-resistant modules prevents their sharing and compromise. Additionally, we have analysed the security of broadcasting sensitive messages over a network, designing a trivial system employing broadcasting and investigating its weaknesses through model checking. Our analysis supports the hypothesis that broadcasting as a technique does not impact the security of a sensitive message, assuming the message is adequately encrypted.

Chapter 6

The Verification System

6.1 Introduction

In the previous chapters, we have discussed the technology upon which this system is based, from the method by which a verdict is determined (Chapter 3) to the protocol employed to protect the determination and gathering processes and their security (Chapters 4 and 5). In this chapter, we discuss the method by which these verdicts are employed in order to arrive at a final decision on the possibility of a Claimant's location claim.

In order for a verification system to function without implicit trust in those devices providing proof, a method of extracting a verdict must be found which relies purely on evidence from random devices. Through this approach, the Verifier may reach an overall verdict regarding a claim through placing its trust in the voice of the majority within a randomly selected group of devices. While individual devices may lie, the probability that the majority of a group would all make the decision to lie about a specific claim is low, and decreases as the size of the group employed increases.

In addition to the verification of claims, the Verifier has the additional task of computing up to date trustworthiness values regarding each of the devices registered within the system. We have designed an approach that employs existing reputation techniques to calculate the trustworthiness or believability of a device based on its prior actions within the system. This is done through recording of each device's behaviour when involved with a claim, using binary notation to indicate positive or negative behaviour. Armed with this information, the *Verifier* can calculate the probability of a device behaving honestly during the current claim, thus providing the system with an idea of the relative trustworthiness of a device

at any time.

The Verifier has three main areas of responsibility; the reputation system, the provision of Proof Providers and finally the verification of claims. The reputation system deals with the calculation of a device's trustworthiness, based on previous events within the system. This trustworthiness may be employed as a factor when dealing with the provision of Proof Providers, a process that begins with gathering a pool of volunteers. These volunteers are examined and the most suitable are selected for use in the proof gathering protocol. The protocol's resulting proofs are then employed by the Verifier in the verification of claims.

In this chapter, we discuss related work in the area of reputation computation (Section 6.2) and outline the reputation system and its facets (Section 6.3). We then discuss two approaches to the gathering of volunteers for use as Proof Proviers (Section 6.4) before comparing several different criteria considered for use when selecting which volunteers to use as Proof Providers (Section 6.5). In Section 6.6 we outline the process of verifying a location claim based on the evidence provided. We then discuss the possibility of extracting a more specific location for inclusion in the Verifier's verdict message if a positive verdict is awarded, based on the evidence available regarding the claim (Section 6.8). Finally, we summarize the chapter's contents and discuss open questions in Section 6.9.

6.2 Related Work

Many reputation computation engines have been proposed to solve the problem of differenciating between those devices that usually behave honestly and those that tend towards dishonest behaviour. A number of detailed surveys, such as [51] and [60] have been published, discussing these engines and the methods by which trustworthiness and reputation are calculated. The reputation system employed in this work is based on Josang and Ismail's beta reputation system [50], in which a form of probability density function (PDF) is employed to calculate the probability of a device behaving honestly based on its past behaviour. The strength of this approach lies in its employment of statistics and probabilities based upon prior actions, grounding the end result in factual events.

In [15], the authors upgrade their CONFIDANT [16] protocol to include a Bayesian-based reputation system, thus removing its exclusive reliance upon negative ratings. The Buchegger-LeBoudec reputation system contains the concepts of both trust and reputation,

with reputation evaluating how well a node participates in the CONFIDENT system and trust evaluating how honest the information published by that node is. The approach taken to reputation is particularly advantageous as greater weight is allocated to first-hand experience than that of published reputation records. Similar to the case of the beta reputation system, this reputation system also includes a method of fading, allowing less weight to be given to those events observed further into the past, a technique which we have also employed (see Section 6.3.4).

In CORE [63], the system employs reputation values regarding nodes to detect and prevent selfish behaviour, thus ensuring cooperation within the mobile ad-hoc network (MANET). In this system, reputation is derived through summations of observed behaviour, combined with reported reputation values, in a weighted calculation. The concept of reputation is split into several variants; subjective, calculated based on first-hand observations; indirect, where the positive observations of others are reported (thus preventing negative attacks on other nodes); and finally functional, which is a combination of subjective and indirect reputations based on a node's performance of a specific function. Within this system, the concept of fading is flipped, with greater weight given to those events further in the past and less weight to those that have recently occurred. The authors reason that this reduces the influence of sudden strange behaviour. However, we believe this to be an indication of the current trustworthiness of the device. Also, while this system does prevent negative attacks on a node, it does not rule out the possibility of a positive attack, where nodes are talked up. This form of attack increases the difficulty of detecting malicious nodes.

Conversely to CORE, Aberer and Despotovic [3] have proposed a distributed reputation-based trust management scheme employing P-Grids [2], in which only negative information is spread. In their approach, the information shared regarding a participant (p) takes the form of a complaint, which is used to calculate the probability that p will cheat again in the future. Unlike previously mentioned systems, this approach takes into account both the complaints made against p and the complaints made by p to extract a reputation value. The final reputation is derived as the product of the number of complaints made about p and the number of complaints p makes about other devices.

6.3 The Reputation System

Within the scope of the Verifier's processes, we employ a reputation system composed of pre-existing techniques to represent the concept of trustworthiness in a device based on past actions. These past actions are recorded to form an event history, tracking a device's behaviour over time. By recording the sequence of events in this manner, a history for each device in the system is produced, giving the Verifier an indication of how it is likely to behave during a verification. These event histories form the basis of trust calculation and are discussed in Section 6.3.2. The reputation system employed in this work is based on statistical theory, utilising an existing form of probability calculation to extract trustworthiness levels for a device from its recorded past behaviour. We discuss this further in Section 6.3.3. In order to ensure the weighting system awards greater weight to recent activity, a fading factor has been included in the system. In addition to this, the ability to fade during quieter periods of activity has also been included, to allow device trustworthiness to decay not only with interactions between a device and the system, but also over time, independant of activity. This process is discussed in Section 6.3.4.

6.3.1 What is Trust?

The notion of trust has been well explored in many academic arenas, including economics [62, 8], sociology [34, 21] and, more recently, computing. This move within computing circles stems from the development of decentralised and Peer to Peer (P2P) systems [71, 68, 91, 9], in which nodes require a method of distinguishing other participants which are more reliable, in order to facilitate their own needs. Numerous trust management and reputation schemes have been developed to solve this problem, such as [40, 15, 100, 3, 52]. In these schemes, the concepts of trust and reputation have been discussed in depth, resulting in several slight variations on their definitions within the field. In this section, we define what trust and reputation mean within this work and how it is employed to facilitate the system's needs.

Within this work, we refer to the procedure through which we calculate a device's trustworthiness as the "reputation" system. Traditionally, reputation and trustworthiness are considered distinct concepts, with one stemming from the public perception of an agent/entity based on second-hand experiences and the other based upon direct interac-

tion [51]. However, this work mixes both of these concepts when calculating a device's trustworthiness. A device's trustworthiness is calculated based on direct interaction between that device and the Verifier, but the outcome of this interaction is based upon calculating a concensus from the second-hand evidence provided. For this reason, we have merged the two concepts, rendering the terms interchangeable. However, for continuity's sake, we use "trustworthiness/trust value" to refer to the trustworthiness of a device, and use "reputation" only when referring to the calculation system. The concept of trust is also extended to apply to relations between malicious devices. We assume that if one device is willing to collude with another, they are "friendly" devices and share a mutual level of trust. This allows them to cooperate in their attempts to deceive the Verifier and location verification system.

When the Verifier selects Proof Providers, it may wish to employ only those that are likely to behave in an honest fashion. In order to facilitate this, we designate each participating device with a measure of trustworthiness, based upon its prior actions within the system. The notion of trustworthiness here represents the level of credibility a device has earned, given its behaviour in the past. It is a measure of how much the system trusts a device to perform its role honestly. This follows Gambetta's definition of trust, which states trust to be: "a particular level of the subjective probability with which an agent assesses that another agent ... will perform a particular action" [34]. The more a device shows itself as an entity to be believed in, the greater its trustworthiness value grows. We have designed the reputation system to calculate the appropriate level of trustworthiness to allocate to a device, based on these past actions. The "shadow of the future" [7] provided by each device's history allows the Verifier to calculate its trust value, a procedure which is discussed further in Section 6.3.3.

However, a device's behaviour may differ depending upon the role played in an interaction. When acting as a Claimant, a device may behave perfectly, as it wishes to profit from the situation. However, when the same device acts as a Proof Provider, it may behave maliciously as it receives no benefits from the interaction and therefore has no obvious incentive not to. The reverse is also possible, where a device behaves honestly as a Proof Provider and only acts maliciously when it requires a false location proved by the system. Therefore, we have designed our system to include the concept of multi-faceted trust,

Algorithm 1 Building Event Histories

Device D joins location verification system, $E_C \leftarrow <>$and $E_{PP} \leftarrow <>$

```
while D_in_System do
  if D_makes_claim then
    if verdict == positive then
      E_C ←< 1 > ∧E_C
    elseif verdict == negative then
      E_C ←< 0 > ∧E_C
    elseif verdict == unsure then
      E_C ← E_C
  if D_is_ProofProvider then
    if verdict == unsure then
      E_PP ← E_PP
    elseif evidence_given == positive then
      if verdict == positive then
        E_PP ←< 1 > ∧E_PP
      elseif verdict == negative then
        E_PP ←< 0 > ∧E_PP
    elseif evidence_given == negative then
      if verdict == positive then
        E_PP ←< 0 > ∧E_PP
      elseif verdict == negative then
        E_PP ←< 1 > ∧E_PP
```

where each device in the system possesses two trustworthiness values, one representing its behaviour as a Claimant and the other its behaviour as a Proof Provider. This concept of multi-faceted trust is seen in many other works, including [40] and [100].

The division of trustworthiness values in this manner allows the different trust facets to be employed at different points within the location verification engine. A device's Proof Provider trustworthiness may be taken into account during the Proof Provider selection process, in order to select only the most trustworthy volunteers from the volunteer pool. It may also be employed during the verification process, depending on the method of verification utilised. This calculation process is dealt with in Section 6.6.3. Currently a device's Claimant trustworthiness is not employed in the verification process. However, by recording the trust values for both aspects, a device's Claimant behaviour may be factored into other areas at a later date.

6.3.2 Tracking Behaviour - Event Histories

As discussed in Section 6.3.1, a device's trustworthiness is split into two facets within this system; trust based on behaviour as a Proof Provider and trust based on behaviour as a Claimant. Therefore, each device involved in the system has associated with it a pair of event histories, one reflecting prior Claimant behaviour and one for its behaviour as a Proof Provider. These event histories are each composed of a sequence of 0s and 1s, reflecting the actions of a device over time. They are used to calculate the trustworthiness of the device when playing a specific role.

When the Verifier reaches a verdict on the possibility of a location claim, it records the outcome in the appropriate event history of each device involved. If the device is determined to have behaved honestly, i.e if a Proof Provider's vote mirrors the verdict issued by the Verifier or if a Claimant's claim is verified, then the event history entry is set as 1, otherwise it is set as 0. The Verifier will update the event history of the device with this entry if the result is deemed to be in the "sure" portion of the possibility scale. However, where the overall result of the claim falls in the unsure portion of the scale, the Verifier refrains from updating the event history, as the outcome of the process is not clear. (See Section 6.6.4 for an explanation of the "unsure" portion of the possibility scale.) The process of building event histories is described formally in algorithm 1.

An example of the event history building algorithm is as follows: Device D joins the system with a pair of empty event histories (as it has not yet behaved either positively or negatively in any role within the system). D makes a location claim, for which it receives a "possible" verdict. This verdict impacts D's Claimant event history, causing it to be updated with a 1, representing a recorded event of honest behaviour. D then plays the role of Proof Provider in another device's location claim, giving that device a negative piece of evidence. However, the Verifier provides that Claimant with a "possible" verdict, thus causing D's Proof Provider event history to be updated with a 0, indicating an event of dishonest behaviour. D again makes a location claim to prove its own location, but this time incurs an "unsure" verdict. In this situation, the Claimant event history is not updated, as the verdict is neither a 0 or a 1. D proceeds to behave as a Proof Provider for two more claims. In the first, D provides positive evidence for the Claimant, who receives a "possible" verdict from the Verifier. This causes D's Proof Provider history to

be updated with a 1, representing its honest behaviour. In the second, D again provides positive evidence, but this time the Claimant in question receives an "unsure" verdict. Due to the overall verdict in this claim being neither 1 nor 0, D's Proof Provider history is not updated. D then takes the role of a Proof Provider and provides negative evidence regarding the presence of the Claimant. The Claimant receives a negative verdict from the Verifier and D's Proof Provider event history is updated with a 1. Finally, D makes a location claim of its own and receives a "not possible" verdict, causing its Claimant event history to be updated with a 0. At this end of this period within the system, D's event histories stand as:

- Claimant event history (E_C): <0, 1>
- Proof Provider event history (E_{PP}): <1, 1, 0>

Note that although D participated in a total of seven claims, three as a Claimant and four as a Proof Provider, only five entries are found in its event histories. This is because "unsure" verdicts do not impact the event histories of devices participating in claims where they are awarded.

As all devices have an empty pair of event histories associated with them when they join the system due to their lack of past behaviour, it is initially difficult to distinguish which devices can be trusted and which have a tendancy to behave badly. For this reason, one area of possible interest for future work is that of "bootstrapping", or building up, a device's event histories, without needing to participate in location claims. This issue is under consideration and is discussed in further detail in Section 6.3.5.

6.3.3 Trust Calculation

The core of any reputation system is its approach to trust calculation. There are a great many of these approaches to choose from. They range from summations/averaging, a principle employed by eBay [81], to discrete trust models in which the trustworthiness of a device is given in verbal statements, rather than on a numeric scale [51]. An example of this is found in the Abdul-Rahman and Hailes model [1], where a device's trustworthiness can be ranked from *Very Trustworthy* to *Very Untrustworthy*. We have chosen a reputation system employing a Bayesian approach to trust calculation. This approach has been chosen

based on its statistical nature, as the Bayesian method works on the basis of computing the trustworthiness of a device through the updating of a beta Probability Density Function (PDF) [29], an approach which also forms the basis of Josang and Ismail's reputation system [50]. As mentioned earlier, we believe the basis of trust lies in the actions which occurred previously, with these actions indicating how a device is likely to behave - a belief which is firmly based in statistical theory. The beta PDF calculates the probability density of events with a binary outcome, making it a perfect candidate for use within this system.

Trust can be represented in two ways, either by the result of a probability expectation value of the beta PDF [51] or by a pair of numbers, (α, β), representing the beta PDF parameter tuple. Within this system, α represents the number of honest behaviour observations for a specific device, while β represents the number of dishonest behaviour observations. The parameter tuple is used to calculate the beta PDF, $f(p|\alpha, \beta)$, which can be expressed using the gamma function Γ as :

$$f(p|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}, \text{ where } 0 \leq p \leq 1,\ \alpha > 0,\ \beta > 0,$$

with the restriction that the probability variable $p \neq 0$ if $\alpha < 1$ and $p \neq 1$ if $\beta < 1$. However, the beta PDF merely represents distributions of binary events (i.e. a positive or negative event). Instead, we employ the probability expectation calculation to calculate the probability of a device behaving honestly during the next encounter, as a value lying in the range of 0 and 1. Where h is the number of observed honest behaviours and d is the number of dishonest, the probability expectation formula is as follows:

$$E(P) = \frac{\alpha}{\alpha+\beta} \text{ where } \alpha = h+1 \text{ and } \beta = d+1, \text{ where } h, d \geq 0.$$

Consider a situation where a Proof Provider has behaved honestly once, and has no dishonest behaviour, the trustworthiness value for the next encounter will be

$$E(P) = \frac{2}{2+1} = 0.666667, \text{ as } \alpha = 1+1 \text{ and } \beta = 0+1$$

while if it behaves dishonestly once, with no honest behaviour, the trustworthiness value will be

$$E(P) = \frac{1}{1+2} = 0.333334, \text{ as } \alpha = 0+1 \text{ and } \beta = 1+1$$

Within the reputation system, h and d are calculated using a device's event histories, with h equalling the number of positive entries and d equalling the number of negative. Using the example event histories generated for device D in algorithm 1, the probability expectation value for D acting as a Claimant is calculated as follows: D's Claimant event history is {0, 1}, with 0 representing a negative event and 1 a positive. As D's Claimant event history contains one positive event, α equals 1 + 1, the number of positive events in the event history plus the possibility of the next event being positive. As D's Claimant history also contains one negative event, β is also equal to 1 + 1. Therefore:

$$E(P) = \frac{1+1}{(1+1)+((1+1))} = \frac{2}{4} = \frac{1}{2}$$

giving D a Claimant probability expectation, or trustworthiness, value of 0.5.

6.3.4 Reputation Fading

The principle of reputation fading is based on the theory that as events recede further into the past, their importance to the calculation of trustworthiness levels of a device decreases. This theory stems from the assumption that older events may no longer accurately represent how a device is likely to behave in the next exchange. The most recent events in a device's event history give the best idea of how it is likely to behave during its next encounter. Therefore, more weight is given to these events than to those further in the past. This approach also allows a device to redeem itself over time, as its bad behaviour, intentional or accidental, decays in importance in the calculation of its trustworthiness.

In a standard Bayesian system, this notion of device fading is not present. Bayesian systems give equal weight to all entries in a device's history, thereby removing the need to retain the order in which those entries occurred. In order to achieve fading within our own system while utilitising this method, we employ the approach used by Buchegger and Le Boudec [15] and Josang and Ismail [50]. In this approach, each entry in the computation of h and d is multiplied by a weight, or fading factor (u), based on its position in the computation sequence, thus discounting older entries in a device's event history. Using this approach, where $0 \leq u \leq 1$ and $S_n, S_{n-1}, ...S_1$ is the event history of a device, the calculation of h and d becomes

$$h = (S_n) + (S_{n-1})u + (S_{n-2})u^2 + ... + (S_1)u^{n-1}$$

$$d = (1 - S_n) + (1 - S_{n-1})u + (1 - S_{n-2})u^2 + ... + (1 - S_1)u^{n-1}$$

Note the use of $(1 - S_n)$ in d's calculation sequence. As the values to be included in the d sequence are "0"s, the event history values cannot be inserted in the same way as for "1"s. $1 - S_n$ removes the "1" events and retains the "0"s, allowing for the accurate calculation of d. As evidenced above, the addition of a new event to the event history fades those already within the sequence. Also, in this approach, when calculating a value, the occurence of events of the opposite kind (i.e. where a negative event occured within the timeline during the calculation of h) is accounted for, allowing events to be faded in the order of their occurence.

However, when computing the values of h and d in this manner, the complete event histories of every device must be retained forever. This is impractical, both in terms of storage and when computing the trustworthiness of a device. These issues can be avoided through re-defining the computation formula for h and d to:

$$h = h * u + S_n$$

$$d = d * u + (1 - S_n)$$

where $0 \leq u \leq 1$ and S_n is the entry being added to the sequence. This approach removes the need to retain the complete sequence of events, as each event is now multiplied by u the appropriate number of times without being explicitly recomputed at each iteration. The new values of h and d can then be used to calculate the trustworthiness of a device using the probability expectation formula shown in Section 6.3.3.

Rather than employing the fading factor suggested in [15] ($u = \frac{1}{m}$, where m = the order of magnitude of the number of entries in the sequence), we have followed the example set by Josang and Ismail and leave u as a customisable variable to be set at implementation. This allows the level of fading implemented to be set based on the requirements of the system in which it is being employed, increasing its flexability. However, when selecting a fading factor, it should be noted that where $u = 1$, the fading factor has no impact, as it multiplies every value by 1, while where $u = 0$, only the newest entry to the sequence is valued, with the rest set to 0.

In addition to the fading approach discussed above, we also include a simple method to fade events over time. Two methods have been considered to address this issue - time-

dependent fading and the employment of a cut off value. Where a cut off value is employed, if the final value in the h or d calculation falls below a certain threshold, (for example - the final value becomes so small that it does not have much impact on the end result of the calculation) then it is removed from the event history completely. However, fading in this manner is flawed in two ways - the value of the final entry in the sequence must fall below a specified threshold and the method described above does not provide for fading without system interaction. Both of these flaws stem from the same requirement: a device must achieve above a certain level of activity before its behaviour is subject to fading.

In order for a device's event history to be eligible for fading, the final entry in the sequence must fall below a set threshold. This is achieved through the sequence filling with a certain number of additional entries, pushing the final entry back far enough to reduce its value to less than the threshold. However, in order for this to occur, a device must participate in enough claims and receive enough behaviour entries in their history to cause the final entry to fall below the threshold. If a device seldom participates in claims, its event history would not fill with enough entries to cause any fading to occur, allowing for its behaviour to remain untouched. Conversely, devices with a high level of participation within the system would incur heavy fading as their event history would fill to the required level repeatedly. This imbalance reduces the accuracy of the final value generated from the event history, fading active devices too harshly while ignoring the behaviour of those more passive devices.

In addition to a lack of fading of the behaviour of less active devices, the above approach also allows no provision for the fading of behaviour without any system interaction. A device may interact with the system many times over a short time frame, building up its event history, before withdrawing from active involvement. Using the above method of fading, during this idle period of time a device's event history would remain untouched, despite the passage of time, as no new entries would be added to the history to cause the value of the final entry to decrease. Therefore, though a device's nature may have altered over time, the history accrued previously remains intact and unaffected.

Rather than compromising between participation levels and fading impact, we have chosen to employ a time-dependent fading mechanism suggested in [15], in which we let $d = u * d$ and $h = u * h$ at the end of every y period of time. This mimics the decay of h

and d's old values where a new entry is added, without the addition of a new entry to the sequence. Through fading based on the passing of time, rather than on history entries or falling below a certain value, fading can occur without devices entering into exchanges, a situation which may easily occur. Also, fading is guaranteed to be equally enacted over all devices involved with the system, whether they play an active role or are mostly dormant.

6.3.5 Initialisation of the Reputation System

An outstanding issue when discussing the reputation system is that of the its initialisation and the initialisation of any new devices when they join the system. As mentioned in Section 6.3.2, all devices join the system with a blank pair of event histories. This causes a problem, as all devices appear equal and distinguishing trustworthy devices from those which have a malicious tendancy is impossible. When performing our own verification simulations (discussed fully in Section 6.7), it was discovered that attempting any approach employing a device's trustworthiness does not provide accurate results without first evolving the trustworthiness of that device to a reasonable level. For this reason, it would be preferrable to provide some method of "bootstrapping" trustworthiness to increase the speed at which a device's trustworthiness values become defined. This bootstrapping concept was also used by Srinivasan, Teitelbaum and Wu in their Distributed Reputation-based Beason Trust System (DRBTS) [90]. At present, no bootstrapping approach is employed within our location verification system. However, future work in this area may provide a solution to the bootstrapping problem without resorting to infrastructure reliance.

As the system currently stands, there exists no trusted infrastructure. This environment makes bootstrapping trustworthness values difficult, as there are no reliable units available to judge the trustworthiness of the participating devices. However, if some completely trusted known entities, such as roadside units (RSUs), were to be introduced to the system, some methods of bootstrapping both Claimant and Proof Provider trust values would become feasible. One such possible method of bootstrapping Claimant trust values is for a Claimant, who is unknowingly in the vicinity of an RSU, to make a location claim. The Verifier, noting that the Claimant is in the area of an RSU, includes the RSU in the claim's pool of Proof Providers. If the Claimant receives a positive verdict from the RSU, it, along with all Proof Providers which agreed with the RSU, receives a positive mark

in the relevant behaviour history, no matter what the outcome of the verification process. This approach to bootstrapping is also suggested in [1].

In order to prevent abuse, any method of bootstrapping employed within the system should not be advertised as such to participating devices. An approach such as the one described in the paragraph above meets this requirement, as the Claimant is unaware of its possible trust gain through the inclusion of a roadside unit in the pool of Proof Providers. If a device was aware that the exchange it was participating in was not a regular claim, its behaviour for that exchange would not be representative of its true nature, making the results possibly deceptive. A malicious device could not only undo any damage to its trust value but could earn a high trust value through repeatedly engaging in bootstrapping. An example of an unsuitable approach would be the broadcasting of bootstrapping "beacons" by RSUs, calling for devices in the area to distance bound with them and receive a positive mark in their behaviour history upon successful completion of the exchange.

However, bootstrapping in this form requires additional equipment and removes one of the system's main advantages: its non-reliance upon infrastructure. Therefore, while it would allow the reputation system's calibration process to become faster and more accurate, it would be preferable to employ an alternative initialisation approach.

6.4 Gathering Proof Provider Volunteers

The main concern when selecting devices is to ensure protection against tampering with the location verification process, as if devices are chosen poorly, the system is left vulnerable to collusion attacks. If collusion is possble, then the proofs supplied could easily be fabricated and not accurately represent the real world situation. The Claimant could be in a completely different location to where it has claimed, yet based on the proofs gathered from collaborating "friendly" Proof Providers, the Verifier would be unable to detect the deception.

The information the system receives during a device's active participation is not sufficient for tracking devices. This is because the last known sighting of a device is not necessarily where it is currently located, nor does it indicate where the device is likely to currently be located. Additionally, storing non-essential information after the system's functionality has been completed leaves the privacy and security of those devices involved

vulnerable if the central database was to be compromised. Therefore, as the system does not have access to information of this nature, we require devices in the area to "volunteer" to act as Proof Providers, based on the receipt of a request message. In order to prevent tampering and collaboration, we have devised two different approaches to locating volunteers, detailed below. Once volunteers have been located, the Verifier then selects the most suitable from the candidate pool based on a set of specific criteria. This selection process and the criteria employed are further discussed in Section 6.5.

6.4.1 Claimant-based Volunteer Gathering

In order for a Verifier to pass judgement on the possibility of a location claim, it first requires volunteer devices from which it can select the Proof Providers to be employed. In the Claimant-based volunteer gathering approach, these volunteers are provided to the Verifier by the Claimant at the time of initiating the claim. The Claimant gathers volunteers from its pool of neighbouring devices through issuing a volunteer request and forwarding the responses to the Verifier. This process is shown below, using abstract protocol notation defined at the beginning of this work, with X's digital signature key noted as K_X^- and its public key noted as K_X^+.

1. C$\rightarrow \mathcal{D}$: F_{Vol},$\mathcal{N}_\mathcal{C}$
2. $\mathcal{D}_\mathrm{i}$ $\rightarrow$C : $\mathcal{N}_\mathcal{C}$, $\left\{ \left\{ |\mathcal{D}_\mathrm{i}, T_{\mathcal{D}_\mathrm{i}}, Loc_{\mathcal{D}_\mathrm{i}}, R_{D_i}| \right\}_{K_{\mathcal{D}_\mathrm{i}}^-} \right\}_{K_V^+}$
3. C$\rightarrow$V : $mesg1$, $\{| \left\{ \left\{ |\mathcal{D}_1, T_{\mathcal{D}_1}, Loc_{\mathcal{D}_1}, R_{D_1}| \right\}_{K_{\mathcal{D}_1}^-} \right\}_{K_V^+}$,.., $\left\{ \left\{ |\mathcal{D}_\mathrm{N}, T_{\mathcal{D}_\mathrm{N}}, Loc_{\mathcal{D}_\mathrm{N}}, R_{D_\mathrm{N}}| \right\}_{K_{\mathcal{D}_\mathrm{N}}^-} \right\}_{K_V^+} |\}_{K_C^-}$

The Claimant broadcasts a message to all devices $\mathcal{D}$ within its range, containing a request flag (F_{Vol}) and an identifying nonce ($\mathcal{N}_C$, a random integer). The request flag informs the recipient that a device in the area (whose identity remains unknown) is seeking volunteers to function as Proof Providers in its location claim, while the nonce allows the Claimant to match any messages it receives to the request. Those devices which wish to participate as a Proof Provider ($\mathcal{D}_\mathrm{i}$) then respond to the Claimant's request with a message containing the identifying nonce received in the Claimant's message and a second message to be forwarded

on to the Verifier. This second message is digitally signed with the volunteer's private key and encrypted with the Verifier's public key, thus protecting the message's contents from eavesdropping devices and preserving the anonymity and confidentiality of the volunteer. The message contains the device's identity ($\mathcal{D}_{\text{i}}$), transmission range (R_{D_i}), a timestamp ($T_{\mathcal{D}_{\text{i}}}$) and the device's current location ($Loc_{\mathcal{D}_{\text{i}}}$).

Once the Claimant has received response messages from the N volunteering devices, it initiates its location claim with the Verifier using an amended version of mesg1 of the SLVPGP extension being employed. The Claimant digitally signs the volunteer responses to prevent undetectable alteration in transit and appends this to mesg1 of the SLVPGP. The nonce portion of the response message is not included in this forwarding process, as this is not required by the Verifier, and the inclusion of a timestamp prevents the Claimant from including older response messages in the volunteer pool without detection.

6.4.2 Verifier-based Volunteer Gathering

An alternative approach to Claimant-based gathering is shown below, where the Verifier is employed in the role of volunteer gatherer. Unlike in Claimant-based gathering, when a Claimant wishes to make a location claim, it is not required to secure a pool of volunteers before initiating the process. Instead, the Claimant initialises the location verification process with an unaltered mesg1 from the SLVPGP, as shown in Chapter 4. When the Verifier receives the Claimant's initialisation message, it creates a volunteer request message composed of a request flag and an identifying nonce (the same volunteer request message sent by the Claimant in Section 6.4.1) and sends it to the area of the location claimed by the Claimant using geographical routing [61]. This approach to routing ensures that the message can be routed to a specific area, rather than a specific recipient, allowing the Verifier to reach multiple devices in a specific area with the same message.

1. V$\rightarrow \mathcal{D}$: F_{Vol},$\mathcal{N}_{\mathcal{V}}$
2. $\mathcal{D}_{\text{i}} \rightarrow$V: $\left\{ \left\{ |\mathcal{N}_{\mathcal{V}}, \mathcal{D}_{\text{i}}, T_{\mathcal{D}_{\text{i}}}, Loc_{\mathcal{D}_{\text{i}}}, R_{D_i}| \right\}_{K^-_{\mathcal{D}_{\text{i}}}} \right\}_{K^+_V}$

When a device in the vicinity of the claimed location receives the volunteer request message and wishes to volunteer, it creates and digitally signs a reply, encrypted using the Verifier's public key, containing the identifying nonce, a timestamp and its identity and current

location. It transmits this message back to the Verifier, who then selects Proof Providers from the pool of volunteers and continues with mesg2 of the SLVPGP as normal.

6.4.3 Claimant vs Verifier - Which Method is Best?

The process of location verification is a sensitive one. As devices are not trusted, the process must prevent as much abuse as possible through its design, denying the ability to tamper with information once it has been sent and preventing deception by malicious devices. This prevention through design must begin with the selection of Proof Providers, otherwise however secure the location proving protocol may be, colluding devices may circumnavigate the security measures and deceive the system. Therefore, the selection of Proof Providers must not be left open to abuse.

Claimant-based volunteer gathering is far more timely than Verifier-based, as it is performed locally, rather than a request being routed over some distance before volunteers can begin responding. It is also performed prior to the initialisation of the claim, reducing the amount of time required to complete the claim. This is of significant advantage when dealing with mobile nodes as depending on their nature, devices can move location with great speed (e.g. devices in VANETs). However, this is a double-edged issue, as if a Claimant recruits volunteers in advance of the claim's initiation, those volunteers may not still be in the area when the proof gathering process begins.

Although Claimant-based volunteer gathering is faster than Verifier-based, it is extremely vunerable to Claimant abuse, as it cannot be guaranteed that the Claimant has followed the volunteer gathering protocol and broadcast a request message to all devices in the surrounding area. This vulnerability leaves the Claimant free to do several things to influence the request's outcome: it can behave honestly and broadcast the request; it can send the request message only to devices in its area that it knows will collaborate with it; and finally, it can send the request message only to devices in a different area (in which it claims to be) which it knows will collaborate with it. This final choice of action would render the location proving system powerless to detect cheating, and would result in false positive verdicts on location claims. A cheating Proof Provider could ignore the distance bounding aspect of the protocol and fabricate proof to be sent to the Verifier. This would result in the Verifier being deceived into returning a falsely positive verdict, with no way

of detecting the fraud due to a lack of dissenting devices. Therefore, the Claimant-based approach to volunteer gathering is not considered suitable for use. Although the Verifier-based approach is slower due to the distance the request message and its responses must travel, in addition to being vulnerable to lying volunteers claiming to be located in the correct area (as in theory, any device that receives the request at some point along the request's route to the location could claim to be located in the correct area and fraudulently volunteer), it is outside the control of the Claimant and is therefore a more secure and suitable solution.

6.5 Selecting the Proof Providers

There can be many devices in the area of the Claimant at the time of a location verification, and so many devices volunteering to act as Proof Providers. Once a pool of volunteers has been established for each claim, Proof Providers can then be selected. However, this process must be designed to minimise the effect of manipulation on a claim's eventual outcome. A malicious Claimant will attempt to manipulate the process in any way it can, including the selection of locations near which only friendly devices lie. It may also attempt to select a location in which the surrounding key locations for minimal range overlap are occupied by friendly devices, or even one near only friendly devices with high levels of trust (in the event that trust levels are leaked from the Verifier's system). In addition to this, a minimum level of Proof Provider participation must be known. This is done to prevent situations where the word of malicious devices can overwhelm that of honest participants with ease. In this section we first discuss the minimum number of Proof Providers that should be required for a claim to occur. We then outline a variety of possible Proof Provider selection criteria considered and the possible vulnerabilities they possess. Each of these approaches have been employed within the verification simulations discussed in Section 6.7 and their effectiveness is compared there.

6.5.1 Minimum Proof Provider Requirements

As this location verification system is designed for use with mobile nodes rather than an infrastructure with fixed deployment levels, the environment in which a Claimant makes a claim can vary extensively in population level. It may be within the vicinity of many

# Proof Providers	Honest positive	Honest negative	Honest inconclusive	Lying positive	Lying negative	Lying inconclusive
4	86.77%	3.04%	10.19%	4.23%	85.71%	10.06%
5	95.55%	0.19%	4.26%	4.08%	87.30%	8.62%
6	96.54%	0.15%	3.31%	3.21%	83.97%	12.83%
7	97.89%	0.15%	1.96%	2.48%	90.99%	6.52%
8	99.56%	0%	0.34%	1.78%	89.91%	8.31%

Table 6.1: Comparing Minimum Proof Provider Requirements

other devices or only a handful of devices scattered over a wide area. The aim of this system is to verify location claims based upon evidence gathered from neighbouring devices. However, this process relies on a majority ruling to provide a final verdict. If the Claimant is operating within a sparsely populated area, it may have only a small group of Proof Providers available for use. This reduces the number of positive verdicts required to reach a majority, raising the issue of minimum Proof Provider requirements - at what point does the system deem there to be too few within a group to extract a reliable verdict from the majority?

With the threat of malicious devices ever-present, a limit is required to distinguish between acceptable and unacceptable levels of participation, in order to prevent malicious colluders from influencing the overall verdict with only a minimal number. While the odds of gaining a single willing colluder in a random area are reasonable, the odds of gathering several willing colluders decrease as the number of colluders needed increases. By including a requirement detailing the minimum number of Proof Providers allowed for a claim to proceed, the system forces a Claimant to have colluders numbering at least 60% of that value (rounded to the nearest whole number) in order to successfully manipulate a claim through to a positive verification. For example, where a minimum of two Proof Providers are required for a claim, a single piece of positive evidence may be all that is required in order to achieve positive verification. Where a minimum of three are required, that number increases to two pieces. Both of these values are extremely low, and therefore neither minimum is considered sufficient for use. Using simulations similar to those described in Section 6.7, we investigated the accuracy of minimum Proof Provider requirements from four up to eight in the verification of honest claims and the detection of false claims.

The results of these simulations are shown in Table 6.1. Based on these simulations,

it is recommended that a minimum of five Proof Providers or higher be employed. This value has shown to be sufficient to provide an acceptable rate of verification of honest claims (95.55%) and detection of false claims (87.30%), with only 4.08% verification of false claims. Requiring higher than five Proof Providers as a minimum may remove the ability of many devices to verify their location due to a low number of available volunteers in their area.

However, while five Proof Providers is recommended for a high level of performance, employing a minimum of four Proof Providers has also proven to have reasonable rates of verification and detection. If the system is employed using a minimum of four Proof Providers, then it should be noted that the actual value computed using the upper threshold (Sections 6.6.3 and 6.6.4) is equal to that computed using the lower threshold. This means that the same value is required to achieve both an "unsure" and a "possible" verdict. In order to address this, the upper threshold employed for values of four or lower is increased to 70%, thus requiring an additional piece of positive evidence to prove a claim's validity.

6.5.2 All Available Volunteers

The simplest approach to selecting Proof Providers is to employ every device within the volunteer pool for that claim. This approach does not limit the number of Proof Providers involved in the claim, thus increasing the amount of possible evidence to be employed in the claim's eventual calculation. This benefits the Verifier as it provides redundancy in numbers, reducing the risk of a claim failing due to the non-participation of selected Proof Providers. The greater the number of Proof Providers employed within a claim, the less impact the non-participation of one. In addition to this, if all volunteers are employed in a claim there is no real method that a malicious Claimant can attempt to manipulate. The only real manipulation that it can attempt is to select a location in which a large number of friendly devices are located. This approach would be employed by most malicious Claimants, no matter what the situation (unless attempting to prove a specific location and not any location other than its own). If the malicious Claimant possesses the location of all devices within the system, the manipulation may be extended as it can then attempt to select a location near only those devices which are friendly to it. However, the possibility of this occurring is extremely unlikely. A Claimant is likely to

possess the locations of those devices it is friendly with, but it would be infeasible for a Claimant to possess the location of all devices within the system.

One potentially problematic disadvantage to employing an "all-volunteers" approach to Proof Provider selection is the risk that the number of volunteers which respond to a claim's volunteer request is unreasonable, due to the Claimant being in a densely populated area. The larger the list of Proof Providers employed within a claim, the more the SLVPGP overhead costs are adversely affected. With transmission and computation overheads related to the number of Proof Providers participating, an overly large number of Proof Providers would increase the load on both the Claimant's equipment and its local network to an unreasonable level. As yet, no work has been done to calculate the maximum viable number of Proof Providers to be employed for any one claim. This issue is not pressing at this time, with the number of volunteers for a claim not envisioned to be overly high. However with the increased adoption of mobile technology in devices, the number of volunteers is expected to grow. A study of the impact of Proof Provider participation levels on SLVPGP overhead costs would allow a reasonable limit to be calculated.

6.5.3 Most Trustworthy Volunteers

Where the number of Proof Providers employed must be limited, we have considered a number of possible selection criteria for use to decide which volunteers are to be included. The first of these is to select based on a volunteer's trustworthiness level. If the selection of Proof Providers is based upon how trustworthy they are, then intuitively the evidence supplied for use will be the most trustworthy evidence possible. This increases the likelihood of the Verifier accurately assessing the claim and producing a correct verdict.

However, this intuitive belief is incorrect on a number of levels. A device may generally behave in a trustworthy manner, supplying accurate evidence to the best of its knowledge. However this does not rule out the possibility of said device being "friendly" with the Claimant and willing to behave maliciously in this instance. Nor does it rule out the possibility of the opposite situation occurring, where the device in question has some form of grudge against the Claimant (assuming in this case that the device has knowledge of the Claimant's identity). Finally, if it gains access to the device trust values, a malicious Claimant may attempt to manipulate the selection process through claiming a location

where the group of most trusted local devices is primarily composed of friendly devices. While the possibility of any device gaining access to the Verifier's list of trust values is slim, as this requires the Verifier to be compromised, we believe that the possibility of this attack should be noted.

6.5.4 Most Geographically Suited Volunteers

A second form of criteria considered factors the location of the volunteers into the Proof Provider selection process. Location-based selection allows the Verifier to select those volunteers located such that the intersection of their transmission areas is the smallest area possible, maximising the information extracted through triangulation and thus providing a much more accurate end location for the Claimant. This concept was previously discussed in Chapter 3.

However, this approach is also vulnerable to attack by a malicious Claimant. Similar to the trust-based approach, location-based selection is done based on targeting a specific attribute and is therefore predictable. This leaves the process vulnerable to manipulation. A malicious Claimant could target a specific location for a claim, knowing the location of its friends in the area and the size of their resulting area of intersection. The Claimant could further include its friends in the deception of the Verifier through having them falsify their locations within the volunteer response, adjusting them slightly to increase the likelihood of selection. Although this attack does capitalise on a vulnerability in the selection approach, it does not guarantee that the Claimant's friends will be selected. Unless the Claimant possesses knowledge of the location of all devices, it cannot defend against the possibility that other, non-friendly, devices may be in more suitable positions, which would render the Claimant's attempted manipulations ineffectual.

6.5.5 Randomly Selected Volunteers

The final approach to Proof Providers selection considered in this work is the random selection of volunteers from the pool, rather than attempting to employ a targeted approach. This approach has one main disadvantage in that it loses any possible advantages gained through targeting certain attributes and strengths of the volunteers. However, the advantage of employing this selection method over targeted methods is clear. Random selection

has a similar advantage to that of employing all volunteers - a malicious Claimant cannot manipulate the selection process any further than the selection of a location in proximity to a number of friendly devices.

6.6 Verification of Location Claims

The Verifier's principle task is to assess the possibility of a device's location claim, based on information gathered from Proof Providers. However, a single proof merely contain that device's location and a verdict regarding the visibility of the Claimant (a binary result). This does not provide the Verifier with much detail to base its assessment upon. For this reason, the Verifier requires some method of extracting a final verdict based on the combined value of the proofs supplied. We outline two extraction approaches for use here. The first employs a simple summation, extracting a final verdict based on the majority view of the claim. The second employs the trustworthiness of those Proof Providers providing supporting (positive) evidence regarding the claim.

The verification process runs as follows: the Verifier receives the proofs and checks them to confirm their applicability to the claim in question (Section 6.6.1). After this, it evaluates each verdict provided (Section 6.6.2) and calculates the overall value of the claim, before comparing this value to the already computed possibility scale (Section 6.6.3). Finally, having made its decision on the possibility of the claim, the Verifier then updates the event sequences of all devices involved in the computation to reflect the events of the verification.

In order to minimise the amount of information retained on devices, particularly regarding their interactions with others, the verification process does not employ details from previous claims, other than those filtered into the reputation system. Though the knowledge that a device may previously have participated in multiple false claims with another device may aid in the identifying of colluding partnerships, possessing this knowledge would require the retention of large quantities of information on every device. This situation may cause privacy concerns with participants and would represent a danger to the privacy of every device within the system should the information's security be compromised.

6.6.1 Pre-Verification - Validating the Proofs

Before the Verifier attempts to determine a claim's verdict, the proofs supplied must first be validated. This process ensures that the Claimant cannot pass off invalid proofs, such as those referring to other devices or those from an older exchange, without detection. If the Claimant is discovered to have attempted this, the claim is rejected and a negative entry is added to the Claimant's event history.

To check a proof's validity, the Verifier first decrypts the message and checks its integrity, to ensure that no tampering has occurred, by checking the digital signature. It then checks that the Claimant identity contained within the proof matches that of the Claimant in question. Once this has been confirmed, it then checks the timestamp, to ensure that it does refer to this exchange and has not been reused from an older verification attempt.

Finally, the Verifier checks the identity of the Proof Provider that created the proof to confirm that the Proof Provider is a member of the group selected by the Verifier during the device selection stage. Without this check, a Claimant could again cheat the system, through ignoring the supplied list of Proof Providers and simply gathering proofs from colluding devices.

6.6.2 Evaluating a Verdict

After the proof gathering process has been concluded and pre-verification has been completed, the overall possibility of the claim is calculated by summing up the values of the verdicts received. The value of a verdict depends upon the verification approach in use. If the Verifier is employing the "Summation" approach, the verdicts are valued as 1 for a positive and 0 for a negative. If the Verifier is employing the "Trustworthiness" approach, the verdicts are valued at 0 for a negative and the trustworthiness of the proof's provider for a positive. If a Proof Provider with a trustworthiness of 0.85 supplies a positive verdict, that verdict is worth 0.85. However, if the same Proof Provider supplies a negative verdict, it is worth 0.

Both approaches to verification value negative verdicts at 0. Employing this form of valuation places emphasis on positive verdicts, while removing the negative values from the computation. This approach is taken due to the fact that a negative verdict may be one of three types: (i) a true negative situation, (ii) an accidental false negative or (iii) a

malicious negative. These situations are also discussed in Chapter 3.

(i) Where a negative verdict represents a true negative situation, it is caused by the Claimant not being present in the area, or not within distance bounding range of that Proof Provider. It is an accurate reflection of the situation, indicating that either the Claimant is lying or mistaken in its location. If a true negative was the only possible negative verdict, they would not need to be removed from the calculation of a claim's possibility. However, this is not the case, with two additional types of negative existing.

(ii) In an accidental negative situation, the Proof Provider supplies a negative verdict despite the fact that the Claimant is in the area of the Proof Provider and within reasonable distance bounding range. This is due to either malfunctioning/underperforming equipment, or performance issues on the local network. If negative verdicts were included in the calculation of a claim's possibility, a Claimant could be unfairly punished for the failure of equipment or network issues, as accidental negatives are not distinguishable from true or malicious negatives.

(iii) The final possible type of negative verdict, a malicious negative, is caused by a Proof Provider knowingly supplying false evidence against a Claimant. In this case, the Proof Provider is attempting to reduce the chances of the Claimant's location claim being verified. It performs distance bounding and participates in the Claimant's exchange honestly so as not to arouse suspicion. However, despite calculating that the Claimant is in the vicinity, it supplies evidence to the contrary of this. As with accidental negatives, there is no method of distinguishing malicious from true negatives. As its inclusion would again unfairly punish the Claimant, this reinforces the case for the removal of all negative verdicts.

The reasoning behind the removal of negative verdicts from the possibility calculation may seem as though it also applies to positive verdicts, thus rendering the entire approach moot. This is not the case, as mistaken or accidental positive verdicts are not possible. Therefore, a Claimant cannot accidentally benefit due to flawed equipment or issues with the communication medium. This is also the case for malicious (or colluding) false positives, giving the system a slight skew towards the cause of the Claimant. The trustworthiness approach to verification seeks to address this this bias, reducing the value of evidence supplied by those that are known to provide false information to reduce the gain received

by a Claimant by its inclusion.

6.6.3 The Possibility Scale

In order to reach a final verdict regarding the overall possibility of a claim within the verification system, we employ a measuring standard known as the possibility scale. This scale allows for all claims to be compared to an equal scale, despite differences between the number of Proof Providers participating. The scale has a minimum of 0, with the maximum re-computed for each claim at the Proof Provider selection stage, based on the number of devices involved in the claim. The scale is split into three bands based on set percentage thresholds (discussed further in Section 6.6.4), each of which reflects a different level of trust in the claim and earns an equivalent overall verdict (possible/unsure/not possible).

After the evidence-containing verdicts have received valuation, the overall value of the claim is extracted. This is done by summing together the verdicts. The overall value is then compared to the claim's possibility scale in order to reach a final verdict. The maximum value on the scale can be calculated in one of two ways, (i) based on the number of devices participating (summation approach) or (ii) based on the trustworthiness values of the Proof Providers (trustworthiness approach).

(i) Where the calculation of the possibility scale is done based solely on the number of devices involved in the claim, the scale is calculated as follows: when the Verifier selects the devices for use as Proof Providers from the volunteer pool, it computes the maximum possible value of the claim through summing the number of devices involved (as each piece of evidence has a maximum possible value of 1 for a positive verdict and a minimum value of 0 for a negative). This is done as follows:

$$Maximum\ value = \#ProofProviders\_Employed$$

(ii) Where the possibility scale is based on the trustworthiness values of those Proof Providers that have provided supporting evidence, the maximum value on the scale is calculated after the evidence-containing verdicts have been received. The calculation adds the number of positive verdicts received and employs this as the maximum value on the scale, as follows:

$$Maximum\ value = \#Positive\_Proofs$$

Algorithm 2 Setting the Thresholds

```
set lower threshold:
  lower_threshold = (maximum_possible_value * 0.4)
set upper threshold:
  if minimum_Proof_Providers > 4 then
    upper_threshold = (maximum_possible_value * 0.6)
  else
    upper_threshold = (maximum_possible_value * 0.7)
if using summation approach then
  lower_threshold = lower_threshold rounded to nearest whole number
  upper_threshold = upper_threshold rounded to nearest whole number
```

By including the trustworthiness of the Proof Providers involved in the claim within the scale's calculation, the credibility of the devices involved has a direct impact on the outcome of the claim. If the positive proofs are provided by very trustworthy devices, they will be of higher value than those provided by less trustworthy devices. Therefore, even if a large group of untrustworthy devices attempts to collude to provide proof of a claim, their trustworthiness will prevent the claim from receiving a positive verdict.

6.6.4 Verdict Calculation Thresholds

Just as there are degrees of trustworthiness in a device, there are also degrees of trustworthiness, or possibility, in a claim. A claim may be possible, not have enough proof to be convincingly possible, or it may be inconclusive. These inconclusive claims lie in the grey area of not outrightly convincingly possible but also not quite at the other end of the possibility scale. Therefore, rather than using only a single threshold distinguishing between a possible and not convincing verdict, we employ two thresholds. This divides the possibility scale into three distinct bands. The thresholds have been set at 40% and 60% of the initially possible trustworthiness respectively. If the summation approach to is employed, the resulting threshold values are rounded to the nearest whole number.

The threshold percentages were arrived at through completing an initial series of verification simulations, using a minimum of five Proof Providers. In these simulations, various percentage values were employed as thresholds to examine their suitability and accuracy. It was discovered that this combination of values (40% & 60%) was the most suitable for distinguishing between lying and honest claims, with the smallest amount of false posi-

Algorithm 3 Computing a Verdict - Summation Approach

claim_value = Σ (*positive_evidence*)
if claim_value $\geq$ upper_threshold **then**
  overall_verdict = *possible*
elseif claim_value $\geq$ lower_threshold **then**
  overall_verdict = *unsure*
else
  overall_verdict = *not possible*

tives/negatives. These thresholds have been employed within the verification simulations discussed in Section 6.7, and have shown to be reasonable limits for claims with more than four Proof Providers, based upon empirical data collected from simulations. In cases where four or fewer Proof Providers have been employed for a claim, the upper threshold is increased to 70%. This prevents a situation where the same number of positive verdicts are required to meet both thresholds, an issue observed in early simulations. The computation of these thresholds is described in algorithm 2.

A separate calculation is undertaken to validate each claim, as the number of Proof Providers included may differ between claims. If this was not the case, the required value to achieve a possible verdict for one set of Proof Providers may be unattainable for another due to an insufficient number of Proof Providers, thus leaving the claim unverifiable. However, the approaches described here recompute the scale for each claim and operate using fixed percentages as thresholds, rather than fixed values. Therefore they are completely adaptable to any device combination which may occur, while remaining consistant with previous evaluations.

6.6.5 Computing the Verdict

Once the maximum value on the possibility scale has been computed and the verdict thresholds have been set, the Verifier completes the verification process by extracting a final verdict on the possibility of the claim being examined. This extraction is done in one of two ways, depending upon the verification approach in use.

The extraction process for summation-based verification is described in algorithm 3. If a value lies within the first band, between the starting value and first threshold, it is deemed not to be possible based on the proof supplied. In this case, a "not possible"

Algorithm 4 Computing a verdict - Trustworthiness Approach

claim_value = Σ (*positive_evidence_values*)
if #Positive_Proofs >= 3 **then**
  if claim_value $\geq$ upper_threshold **then**
    overall_verdict = *possible*
  elseif claim_value $\geq$ lower_threshold **then**
    overall_verdict = *unsure*
  else
    overall_verdict = *not possible*
else
  overall _verdict = *unsure*

verdict is returned. If a value is within the second band, lying between the first and second thresholds, the claim is deemed to be inconclusive based on the proof supplied and a verdict of "unsure" is returned. Finally, if a claim's overall value is found to be between the second threshold and the maximum value, then the claim is deemed to be possible and a verdict of "possible" is returned.

The extraction process for trust-based verification is described in algorithm 4. This process is similar to that of summation based, with two basic differences. The first is the method by which the claim's value is computed. In this approach the value of the claim is the sum of all evidence values for that claim. As discussed in Section , when employing trustworthiness-based verification the value of a verdict is calculated:

$$trust\_of\_supplier * x$$

where x is either 0 or 1. If the verdict is positive, $x = 1$ and if the verdict is negative, $x = 0$. Therefore, the summation of all evidence values for a claim becomes

$$\Sigma\,(positive\_evidence\_values)$$

The second difference between summation-based and trust-based verdict computation is the introduction of a condition requiring a minimum number of positive proofs to have been received for a claim. This prevents a claim from being verified on the strength of a single proof, even if that proof is from an extremely trustworthy Proof Provider. If this requirement is not met, the claim receives an "unsure" verdict, thus preventing the parties involved from being punished without cause.

With the claim's value calculated and the minimum number of positive proofs met, trust-based verdict computation continues in the same manner as summation based. The claim's value is compared to the upper threshold and if it is found to be greater than or equal to this value, it receives a "possible" verdict. If it is less than the upper threshold, it is compared to the lower threshold. If the claim's value falls between the two thresholds, it receives a verdict of "unsure". However, if the claim's value is below that of the lower threshold, a verdict of "not possible" is returned.

After the final possibility verdict has been reached, the event histories of the devices involved in the claim are updated, based upon the claim's received verdict. If the final result is either "possible" or "not possible", the event histories of the Claimant and those Proof Providers which responded are updated. An update of "0" indicates that they did not agree with the calculated verdict, with "1" indicating that they were in accord. However, in the case of an "unsure" verdict, the event histories of the devices involved are not updated as the results of the claim were inconclusive and are neither positive nor negative, therefore cannot be represented by either of the two possible options. This process is described in algorithm 1.

The nature of the verification process and its connection to the reputation system in this manner leaves the system open to abuse, whereby colluding devices can attempt to reduce the trust in others through consciously returning the opposite verdict. The simplest example of this is where a group of Proof Providers seeks to harm the trustworthiness of their Claimant through all providing a negative verdict, even if the Claimant has proven itself to be within range. If the majority of the Proof Providers participating in the claim provide a negative verdict, the claim will result in an unsure verdict at best, rendering the efforts of the Claimant wasted. This attack can be guarded against through cautious and unpredictable Proof Provider selection processes (Section 6.5) but cannot be completely prevented, particularly in the case of Proof Providers attacking a Claimant. This is because the Proof Providers are all focused on a single device, and are aware that the Claimant is (usually) seeking a positive response from them. However, without knowing that the other Proof Providers involved are also acting maliciously in this manner, a malicious Proof Provider risks its own trustworthiness rating. If Proof Providers remain unaware of their counterparts within the protocol (such as in extensions two (Section 4.4.2) and three

(Section 4.4.3) of the SLVPGP), they cannot guarantee that their malicious attempt will be supported by others and may not want to risk their own trust level.

6.7 Verification Simulations

In order to confirm the effectiveness of the described verification system and the different approaches that may be employed, we have performed multiple system simulations. These simulations test the performance of the different approaches in increasingly hostile environments, in addition to testing the effectiveness and resistance to manipulation of the four considered criteria for Proof Provider selection, outlined in Section 6.5. In this section, we describe the system (Section 6.7.1) before outlining the simulations themselves (Section 6.7.2) and analysing their results (Section 6.7.3).

6.7.1 Verification Simulation System Outline

The simulations outlined here have been implemented in Java and employ no existing simulation software. Devices are created from an XML file and modelled as threads, with each device spawning a single Claimant thread and creating Proof Providers as required by the Verifier. An algorithmic description of the Verifier's functionality is provided in algorithm 5, with descriptions of the Claimant and Proof Provider agents provided in algorithms 6 and 7. The Claimant threads live for the duration of the simulation run and randomly attempt to have a location verified. A single Verifier thread also runs for the duration of the simulation, dealing with verification requests and keeping a count of the number of claims initiated. Once the required number of claims have been completed, the Verifier sends a terminate command to each device and waits for their confirmation. It then writes the updated device information to the XML file and completes its run. It is assumed that connectivity is bi-directional, i.e that if device A can see device B, then B can also see device A. It is also assumed that if a device is in the area of the location being claimed, then that device will automatically volunteer to act as a Proof Provider.

In order to ensure that the devices are representative of real world entities and do not follow a pre-programmed pattern, an element of choice has been included in their design. Claimants have the ability to decide to lie regarding a location claim, while Proof Providers have a choice of three options in their approach to providing evidence regarding

Algorithm 5 Description of Simulated Verifier Behaviour

```
while verification_count <= total_to_be_performed
  if contacted_by_Claimant then
    check ProofProvider_availability
      count devices in region of location provided by Claimant
    if(num_devices < minimum_ProofProviders_Required) then
      reject claim request
    else
      select ProofProviders [selection algorithms available in Appendix M] and set thresholds
        if verification_approach is summation-based then
          maximum_value = number of ProofProviders
        else
          maximum_value = number of positive proofs
        lowerThreshold = maximum_value * 40%
        if number of ProofProviders > 4 then
          upperThreshold = maximum_value * 60%
        else
          upperThreshold = maximum_value * 70%
      if verification_approach is summation-based then
        upperThreshold = upperThreshold rounded to nearest whole number
        lowerThreshold = lowerThreshold rounded to nearest whole number
      supply ProofProvider list to Claimant
      receive Proofs and calculate verdict
        if verification_approach is summation-based then
          claim_value = total_evidence_values
        else
          for evidence in Proofs
            claim_value = claim_value + (evidence_value*trust_of_supplier)
        if verification_approach is trust-based and maximum_value < 3 then
          verdict = "unsure"
        elseif claim_value >= upperThreshold then
          verdict = "possible"
        elseif claim_value >= lowerThreshold then
          verdict = "unsure"
        else
          verdict = "not possible"
      return verdict and update event histories of devices involved
        if verdict == possible then
          claimant_history_value == 1
        elseif verdict == "not possible" then
          claimant_history_value == 0
        for all ProofProviders in claim
          if verdict == ProofProvider_verdict then
            ProofProvider_history_value == 1
          elseif verdict != ProofProvider_verdict and verdict != "unsure" then
            ProofProvider_history_value == 0
      increment verification_count
for all devices in world
  issue terminate command
```

Algorithm 6 Description of Simulated Claimant Behaviour

```
while not terminated by Verifier
  initiate claim
    if trusted_selection employed then
      recompute influence map
    set location to be provided
      generate random behaviour_decision value
      if behaviour_decision > internal_honesty then
        set location to optimal location computed by influence map
      else
        set location to current location
    provide location to Verifier
    if location is within range of sufficient Proof Providers then
      for all supplied Proof Providers in list
        request proof
      supply proofs to Verifier
      receive verdict from Verifier
    else if not enough Proof Providers then
      decrement value for location in influence map
  sleep for randomly generated quantity of time
```

a claim. Once queried for a verdict, a Proof Provider can choose to either answer honestly, lie or remain unresponsive (i.e. give no verdict). This ability is modelled using random number generation. Each device possesses two internal_honesty measures, one for use as a Claimant and one for use as a Proof Provider. These values, set between 0 and 100, are not related to the trustworthiness values computed by the Verifier within the system. They represent a device's innate nature and not its pattern of previous behaviour. They dictate the odds that a device will behave trustworthily in a particular role, although a device may defy the odds and produce a pattern of behaviour in opposition to its internal_honesty value.

Before a device acts, either by making a location claim as a Claimant or by providing a verdict as a Proof Provider, it generates a random number between 0 and 100, known as its behaviour_decision. If this number is less than or equal to its internal_honesty for that role, the device proceeds honestly. However, if the behaviour_decision is greater than the device's internal_honesty, the device proceeds dishonestly. In the case of the Claimant role, the device in question makes a fraudulent location claim. In the case of the Proof Provider role, the device will lie about its verdict. Lying in this situation does not automatically guarantee a favourable response for the Claimant. This approach is also em-

ployed to dictate a device's responsiveness. A device possesses an internal_responsiveness value for use in the Proof Provider role. If a Proof Provider generates a random number (its responsiveness_decision value) greater than the internal_responsiveness value, it will refuse to provide a verdict for that Claimant, subject to some exceptions caused by the existance of friendships.

All devices with an internal_honesty value below a specfied "dishonest" threshold (i.e. those that are considered to possess a truly malicious nature and are not simply opportunistic) possess a list of "friends". This list is comprised of those devices that have previously been lied in favour of or have lied to benefit that device. It is assumed that friendship is bi-directional, if device A considers device B a friend, then device B also considers device A a friend. These friendships are used to dictate exceptions to normal behaviour. If a Claimant is considered a friend of a Proof Provider, it will receive a positive verdict, regardless of a Proof Provider's behaviour and responsiveness decisions. It is assumed that devices have access to all information regarding their friends, including personal information such as their identity and current location.

As discussed in Section 6.5, four different criteria have been considered within this work for use to select Proof Providers from the pool of available volunteers. These are: all available volunteers, the most trustworthy of volunteers, the most geographically suited of volunteers and finally, random selection. The algorithms employed by the Verifier to filter Proof Provider selection based on each of these can be found in Section B.1 of Appendix B. Upon initiation of the simulation system, the Verifier thread is given a parameter indicating which of these criteria is to be employed in its selection of Proof Providers. This parameter is also passed to all devices, modelling the worst case scenario in which malicious agents within the system possess privileged information regarding the functionality of the Verifier and exploit this information to their advantage. The Claimant employs this information within an influence map [92], which is designed to calculate the optimal location for use within the world when lying regarding its location. For each algorithm employed by the Verifier to select Proof Providers from the available volunteers, the Claimant's influence map possesses a counter-approach which attempts to calculate the location within range of the fewest number of non-friend devices, based on that selection criteria. These counter-algorithms are shown in Section B.2 of Appendix B. As there is no real method of defending

Algorithm 7 Description of Simulated Proof Provider Behaviour

```
when ProofProvider requested by Verifier
  if verdict requested by Claimant then
    generates random behaviour_decision value
    generates random responsiveness_decision value
    if internal_honesty <= dishonest_threshold and Claimant == friend then
      verdict = positive
    elseif responsiveness_decision <= internal_responsiveness then
      set verdict = negative
     if Claimant_x_coordinate == within range and Claimant_y_coordinate == within range then
        verdict = positive
      if behaviour_decision > internal_honesty then
        invert verdict
          if internal_honesty <= dishonest_threshold and Claimant is dishonest and verdict ==
positive then
          save Claimant as friend
    else
      set as unresponsive, give no verdict
    provide verdict
```

against the Verifier's random selection approach, the influence map employs the same counter-algorithm to defend against random selection as it does to defend against the employment of all available volunteers.

The Verifier thread has been equipped to simulate both trust-based and summation-based verification approaches. Upon initiation of the simulation system, the Verifier thread is given a parameter indicating which approach is to be employed in the verification process. The algorithm outlining the Verifier's functionality (algorithm 5) shows the way in which this parameter is employed to differenciate between verification approaches.

6.7.2 Verification Simulation Outlines

In this section, we outline the specifications employed within the simulations. The XML files employed within the simulations each contain a total of 200 devices, made up of x "trustworthy" devices and y "untrustworthy" devices. These 200 devices exist within a 100x100 grid. Where a device is classed as trustworthy, its internal_honesty values are set between 95 and 100. Where a device is classed as untrustworthy, its internal_honesty values are set between 0 and 20. For simplicity's sake, the same value was used as the Claimant internal_honesty and Proof Provider internal_honesty. All devices are allocated an internal_responsiveness value of between 90 and 100. A naming scheme based on the number of honest and dishonest devices within the XML file has been employed. The

	200_0	190_10	180_20	160_40	150_50	140_60	120_80	100_100
Honest verified	100.00	99.35	98.41	89.37	85.63	77.97	69.66	49.61
Honest failed	0.00	0.00	0.23	0.00	0.28	3.19	1.38	13.18
Honest inconclusive	0.00	0.65	1.36	10.63	14.08	18.84	28.97	37.21
Lying verified	0.00	1.47	1.67	1.68	4.79	8.97	9.48	30.86
Lying failed	100.00	96.95	96.67	82.35	69.86	52.56	45.97	25.93
Lying inconclusive	0.00	1.58	1.67	15.97	25.34	38.46	44.55	43.21

Table 6.2: Performance Comparison of Summation-based Verification Across Increasingly Hostile Environments Using a Minimum of Five Proof Providers

number of honest devices appears first, followed by an underscore and finally the number of dishonest devices, e.g. 180_20, indicating 180 trustworthy devices and 20 untrustworthy devices.

We have employed eight different XML files in these simulations, each of which is composed of a different percentage of trustworthy devices. These range from 50% trustworthy to 100% trustworthy. The devices within these files have been pre-trained in order to "bootstrap" their reputations, thus providing the Verifier with an indication of their trustworthiness. For each XML file, we have carried out eight simulation runs, the total number of different parameter combinations. A run's parameters are composed of the minimum number of Proof Providers required to have a claim verified (set at five), the verification approach to be employed by the Verifier and the selection method to be employed by the Verifier and Claimants. Where applicable, the maximum number of Proof Providers employed has been set at seven. Each simulation run terminates after 500 verifications have been completed.

6.7.3 Verification Simulation Results

In order to analyse the simulation results using different criteria, this section has been broken into two subsections; the overall functionality of the verification approaches outlined and the method of Proof Provider selection employed.

Overall system functionality

The graph shown in Figure 6.1 depicts the percentage of honest claims made that received a possible result (shown by the graph's top line) within each of the XML environ-

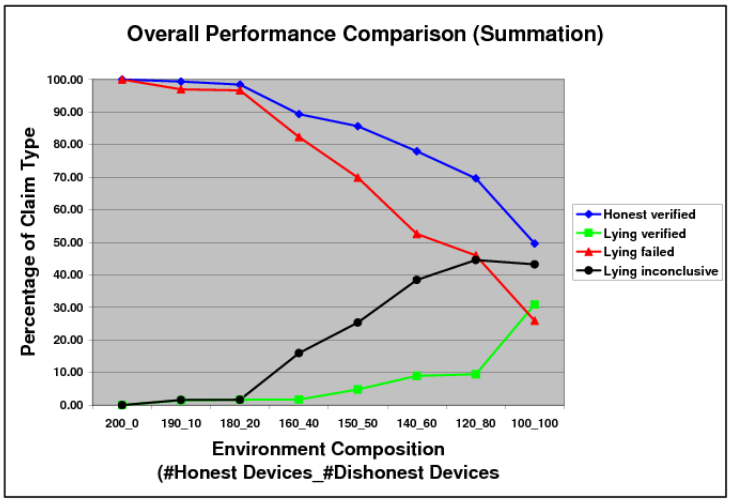


Figure 6.1: Performance Comparison of Summation-based Verification Across Increasingly Hostile Environments, Using a Minimum of Five Proof Providers

	200_0	190_10	180_20	160_40	150_50	140_60	120_80	100_100
Honest verified	100.00	99.35	98.41	89.37	85.63	77.97	69.66	49.61
Lying verified	0.00	1.47	1.67	1.68	4.79	8.97	9.48	30.86

Table 6.3: Comparison of Verification Rates Across Increasingly Hostile Environments

ments, when employing **summation-based verification** using a minimum of five Proof Providers. The remaining lines within the graphs indicate the percentage of false (lying) claims that received a possible (verified), unsure (inconclusive) or not possible (failed) result. The numbers used to generate Figure 6.1 are shown in Table 6.2. As expected, the number of honest verifications trends downwards as the number of honest devices (i.e. those with a trustworthiness level of greater than 95) within the environment decreases, as does the number of failed false claims. However, despite this downward trend in the number of failed false claims, the number of verifications received for false claims does not greatly increase as the level of dishonest devices increases within the environment, with the exception of the most dishonest environment, 100_100. Instead, there is a marked increase in the number of inconclusive verdicts given. This indicates that the upper threshold employed to confirm the veracity of a claim is set at a sufficient level to protect against inaccurate positive verdicts in marginally to moderately hostile environments. The figures shown in Table 6.2 further demonstrate this trait. The number of lying claims verified remains below 10% in every environment bar the most hostile, 100_100.

When examining the numbers generated by the most hostile environment, 100_100, the

	200_0	190_10	180_20	160_40	150_50	140_60	120_80	100_100
Honest verified	98.38	96.65	94.25	93.27	81.09	72.28	70.14	30.48
Honest failed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.37
Honest inconclusive	1.62	3.35	5.75	6.73	18.91	27.72	29.86	68.15
Lying verified	0.00	1.17	2.82	0.00	0.87	6.02	5.00	28.23
Lying failed	0.00	0.00	0.00	0.00	0.00	3.76	9.38	0.96
Lying inconclusive	100.00	98.83	97.18	100.00	99.13	90.23	85.63	70.81

Table 6.4: Performance Comparison of Trust-based Verification Across Increasingly Hostile Environments Using a Minimum of Five Proof Providers

results for verification of honest claims are unsurprisingly low at 49.61%. This indicates the need for a higher percentage of honest devices to be present in the environment in order for verification to correctly function. Additionally, the percentage of false claims that received a positive verdict is 30.86%, giving a high possibility that a false claim would be undetected. This figure is dramatically reduced when the environment's ratio of honest to dishonest devices moves from 100:100 to 120:80. The results generated by simulations employing this environment show a false positive percentage of 9.48%. Similarly, the percentage of honest claims verified increases from 49.61% to 69.66%. This improvement is charted in Table 6.3, which compare the percentage of total honest and false claims verified. The figures shown in this table indicate that once the level of honest devices present within an environment rises to 75%, the system's performance reaches a reasonable level, with 85.63% of all honest claims verified and 4.79% of all false claims verified.

Overall, the results generated from simulation of a summation-based approach to verification demonstrate a high level of system performance, even in situations with high percentages of dishonest devices. This proves that the designed approach can both verify an honest claim and detect a false claim. The system successfully verifies honest claims up to 100% of the time, depending on the percentage of honest devices within the environment. In the case of false claims, the system provides either a rejection or inconclusive result between 69.14% (in an environment where 50% of the devices are highly malicious) and 100% of the time.

The graph shown in Figure 6.2 depicts the percentage of honest claims verified when assessed using **trust-based verification**. As with the investigation of summation-based verification, the claims employ a minimum of five Proof Providers. The remaining lines within the graphs indicate the percentage of false (lying) claims that received a possi-

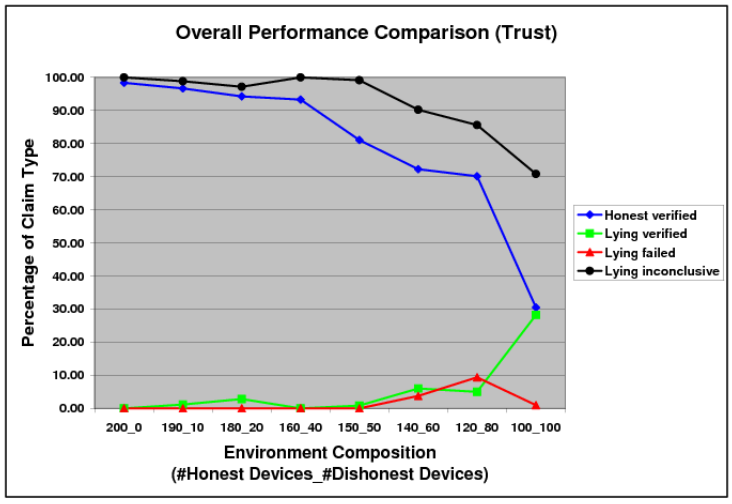


Figure 6.2: Performance Comparison of Trust-based Verification Across Increasingly Hostile Environments, Using a Minimum of Five Proof Providers

Approach	200_0	190_10	180_20	160_40	150_50	140_60	120_80	100_100
Summation-based	100	96.95	96.67	82.35	69.86	52.56	45.97	25.93
Trust-based	0.00	0.00	0.00	0.00	0.00	3.76	9.38	0.96

Table 6.5: Comparing Failure Rates of Lying Clams Between Verification Approaches

ble (verified), unsure (inconclusive) or not possible (failed) result. The numbers used to generate this graph are shown in Table 6.4.

As with summation-based verification, the number of honest verifications trends downwards as the number of honest devices within the environment decreases. Figure 6.1 shows a relatively smooth descent, but the trend seen in Figure 6.2 is more dramatic. The initial figures for trust-based verification are relatively similar. However, the number of honest claims verified quickly drops below those of summation-based, dropping as low as 30.48% for the 100_100 environment (almost 20% below summation-based's minimum). Meanwhile, the number of failed false claims remains below 10% in all environments, with an average of 1.76%. Table 6.5 provides further illustration of this, comparing the failure detection rate within each environment. This table clearly demonstrates that summation-based verification is more suited to the detection of false claims, with an average detection rate of 71.29%. Trust-based verification achieves only a fraction of this, with an average of 1.76% claims outrightly rejected.

However, despite the low number of detected false claims, trust-based verification is more adept at preventing the verification of false claims. While summation-based verifica-

	200_0	190_10	180_20	160_40	150_50	140_60	120_80	100_100
Honest verified	98.38	96.65	94.25	93.27	81.09	72.28	70.14	30.48
Lying verified	0.00	1.17	2.82	0.00	0.87	6.02	5.00	28.23

Table 6.6: Comparison of Verification Rates Using Trust-based Verification, Across Increasingly Hostile Environments

tion incurred a average false verification rate of 7.37% across all environments, trust-based verification incurred an average of only 5.51%. The figures shown in Table 6.6 compare the number of honest verifications against the number of lying claims verified for trust-based verification. The number of lying claims verified remains below 6.5% in every environment bar the most hostile, 100_100.

When examining the numbers generated by the most hostile environment, 100_100, the results for verification of honest claims are again low at 30.48%. This supports the hypothesis that a higher percentage of honest devices is needed within the environment in order for verification to correctly function. As with summation-based verification, the percentage of false claims that received a positive verdict is high, at 28.23%. However, the reduction when moving from 100_100 to 120_80 is far more dramatic than that of summation-based, with a false positive percentage of only 5%. Similarly, the percentage of honest verifications increases from 30.48% to 70.14%, a far greater increase than that seen in the summation-based results. This improvement is charted in Table 6.6, which indicates that once the level of honest devices present rises to 75%, verification performance reaches a reasonable level, with 81.09% of all honest claims verified and 0.87% of all false claims verified.

The results of these simulations assessing the performance of trust-based verification indicate that while the approach performs admirably when verifying honest claims (verifying an average of 79.57% of all honest claims), it does not possess the ability to clearly identify a false claim. While the number of false claims verified is lower then that achieved by summation-based verification, there is an extremely high number of inconclusive results, reaching 100% of the attempted false claims (within the 200_0 environment). However, the accuracy of the system is designed to increase over time, with more information becoming available regarding the trustworthiness of the Proof Providers involved. Therefore, further simulation time would be beneficial in order to study the effectiveness of trust-based verification on a more long term scale.

Figure 6.3: Selection Method Comparison Within the 160_40 Environment

	All Available	Most Trustworthy	Most Suitable Location	Randomly Selected
Lying verified	1%	4.05%	19.82%	4.78%
Lying failed	73%	86.49%	67.57%	72.11%
Lying inconclusive	26%	9.46%	12.61%	23.11%

Table 6.7: Selection Method Comparison Within the 160_40 Environment - Lying Claims

Method of Proof Provider selection

As mentioned in Section 6.5, the issue of Proof Provider selection is a delicate matter, as it is at this point in the verification process that the Claimant has the most potential for influence. We introduced four possible Proof Provider selection methods in Section 6.5; *all available options, most trustworthy devices, most suitable location* and *randomly selected*. For each of these methods, we discussed the possible exploitations a malicious Claimant could possibly employ in order to manipulate them into selecting Proof Providers of their choosing. In this section, we analyse the differences in performance demonstrated in simulation results. We discuss these differences within the scope of a single environment: 160_40, the 100x100 world containing 160 trustworthy devices and 40 untrustworthy devices. For the sake of clarity, we discuss only results computed using the summation-based verification approach, as this has demonstrated a higher level of accuracy.

The graph shown in Figure 6.3 display the results of all lying (false) claims, broken into the number of positive, negative and inconclusive results received using each method of selection, as a percentage of the overall number of lying claims. The figures corresponding

	All	Trustworthiness	Geographic	Random
Honest verified	91.52%	95.32%	92.29%	91.00%
Honest failed	0.50%	0.23%	0.51%	1.22%
Honest inconclusive	7.98%	4.45%	7.20%	7.79%

Table 6.8: Selection Method Comparison Within the 160_40 Environment - Honest Claims

to this graph are shown in Table 6.7. In this set of simulations, a minimum of five Proof Providers was required in order for verification to occur. While it was expected that the two uncontrollable methods, *random* and *all*, would be the most successful at detecting false claims, this is not the case. Instead, it was discovered that *trustworthiness* performs the most accurately, at a rate of 86.49%. Though a malicious Claimant may attempt to influence the selection process through selecting a location near devices with a lower trustworthiness than the friendly devices also in that area, it would require the majority of the devices selected to be friendly. Otherwise, the reliability of the honest devices selected will outweigh the impact of any malicious friendly devices. As the devices involved in this situation have been bootstrapped to a point where a reliable trustworthiness level has been established, selection based on this factor appears highly accurate, even in a 20% hostile environment. It should also be noted that using trustworthiness as the selection method for Proof Providers results in the highest verification rate and lowest inconclusive rate for honest claims (Table 6.8), making it the most accurate approach within this environment.

Though selection based on trustworthiness proved to be far more reliable than first believed, this is not the case for geographic selection. As expected, the selection method most vulnerable to manipulation is selection based on this criterion. The results gathered within the 160_40 environment show that geographic selection has the highest rate of false claims verified (19.82%). Geographic selection also resulted in the second lowest number of inconclusive verdicts (12.61%) for lying claims, less than half the number where all volunteers were employed (26%). The majority of geographically selected false claims were rejected, at 67.57%. However, this is the lowest number of failures received any of the selection methods, reinforcing the hypothesis that geographic selection is the most vulnerable to manipulation.

6.8 From Verification to Localization

Though a device's general location claim can be verified, the accuracy of the claim is unknown. If a Claimant proves itself to be at a claimed location, it Claimant may be located anywhere in the vicinity of that location. Without including a refinement process after a positive verification, the location contained within the Verifier's final message to the Claimant ties that Claimant only to the last location provided (X_C). The location at which the Claimant must have been while completing the evidence gathering process is never calculated. In order to address this issue and improve the functionality of the system, it is important to investigate the actual location of the Claimant and provide this location within the Verifier's proof message, rather than that dictated by the Claimant. This is done through multilateration, using those Proof Providers that provided supporting evidence as starting points.

6.8.1 Finding the Location

Multilateration is the overlapping of device transmission ranges in order to extract a small area of intersection. The process of multilateration employs parameter pairs, comprised of the location of each device being used in the process and that device's transmission range. The information contained within these parameter pairs is collected during the normal progression of the SLVPGP and volunteer gathering process. The device location information is provided within the final proof messages of the Proof Providers and the range information is included within the volunteer response messages.

The information contained within the parameter pairs is employed as follows: the location provided for a device becomes the centre of a circle, with the transmission range providing the circle's radius. With these two pieces of information, an area corresponding to the transmission capability of each included device is found. Through overlapping these areas, the area of intersection is discovered. In order to be within range of all included devices to earn the received positive evidence, the Claimant must have been located within this area of intersection during the evidence gathering stage. Therefore, by discovering the area of intersection of the supporting Proof Providers, a more accurate location for the Claimant is found.

However, while this approach is appropriate for devices with a low level of motion in a

short space of time, such as devices carried by pedestrians, it does not extend to those that travel large distances in a short time. For example, within a VANET, a device may travel over 30 metres in a single second. This means that the location at which it distance bounded with one Proof Provider may be quite different from that at which it distance bounded with a second. While the transmission range of an IEEE 802.11 device may theoretically reach 180 metres (IEEE 802.11n), this does not guarantee an overlap between locations. As mentioned in Chapters 1 and 4, no field work has been done to confirm the ability of this location verification system to function in VANET environments, with mobility remaining a critical question mark over its applicability in the area. Therefore, further work is required to investigate the system's suitability for highly mobile devices and to adequately address the issue of overlapping locations.

6.8.2 Improving Location Accuracy

Although the area in which the Claimant may be located is improved through the multilateration process, some steps exist which can further increase the accuracy of this area. In situations where only a limited number of Proof Providers may be used, the effectiveness of the multilateration process may be improved through employing the geographic criterion when selecting Proof Providers for use in a claim (Section 6.5.4). By selecting devices with a very small area of intersection, a more precise location may be found. Where an over-abundance of Proof Providers is not a danger, selecting all volunteers for use (Section 6.5.2) provides the best level of accuracy, as it gives the largest number of locations and transmission areas possible for that claim, and thus, the smallest possible region of overlap in which the Claimant could be located.

6.9 Summary

In this chapter, we presented the location verification engine, which takes the Proof Provider verdicts supplied by the SLVPGP and extracts a possibility value for the location claim being verified. We have presented two methods of verification, neither of which require trusted infrastructure to instill trust in the evidence gathered for use in a claim. Instead, the evidence is judged based upon either the number of concurring devices or the trust placed in those devices.

In order to provide a method of gauging the trustworthiness of a device, we outlined a combination of existing techniques employed to create a reputation system. With the functionality of the reputation system established, we proceeded to discuss the remaining responsibilities of the location verification engine, namely the provision of Proof Providers and the calculation of a claim's possibility. We put forth two basic approaches to the gathering of volunteers for use as Proof Providers, Claimant-based and Verifier-based, but concluded that a Verifier-based approach should be employed to reduce the likelihood of collusion attacks. We then discussed a number of possible selection criteria to select Proof Providers from the volunteer pool.

We proceeded to outline the two processes by which the Verifier can extract a possibility value for a given location claim, based on its supplied evidence. We discussed the concept of the possibility scale, along with the use of flexible thresholds instead of fixed grading values. We demonstrated the performance and accuracy of both verification approaches through performing java simulations. These simulations show that trust-based verification has potential but may require a high level of training to be truly effective, while summation-based verification performs at reasonably high levels of detection with up to 25% of the devices within the environment internally malicious (i.e. biased to behave maliciously in at least 80% of their interactions). We also investigated a number of different criteria for use in Proof Provider selection. We discovered that although selection based on the trustworthiness of a device is predictable, it is also the most adept at detecting false location claims. We then closed the chapter with a brief proposal of how location accuracy can be improved within the system.

Chapter 7

Conclusion and Future Work

7.1 Review

With the growth of mobile technology into an affordable and accessible market, localization has become increasingly important to extract the maximum information about a user's situation. Within this area, location verification is increasingly appealing, in order to ensure an accurate reflection of the user's situation. However, a common weakness in both localization and location verification systems is reliance upon the presence of infrastructure in order to locate a device.

In this thesis we have presented the design for a novel location verification system that does not require the use of infrastructure or implicitly trusted devices. This work extends previously existing location verification approaches and incorporates work from previously unincluded research fields into the area of location verification to remove this reliance. We have addressed this weakness through employing an ad-hoc approach to the problem, enlisting neighbouring devices to provide supporting evidence of a specific device's presence in a claimed area. The research in this thesis has:

- established that the time required to complete a direct (unamplified) message exchange over an IEE802.11 network greatly differs from that required to complete a proxied exchange.
- proposed a new metric to take advantage of this distinction, allowing for the detection of proxy attacks in distance bounding.
- developed a pair of algorithms to compute the bounds for this metric on the fly,

allowing the bounds to reflect the state of the network at the current time.

- developed a set of secure protocols (the SLVPGP hierarchy) for use to protect the process of gathering evidence supporting a device's location claim.
- compared the incurred costs of these protocols against the gain in security when moving from one level to another.
- verified the security claims of the SLVPGP hierarchy through model checking using a combination of Casper and the FDR model checker.
- established a method of gathering neighbouring devices to act as Proof Providers for a specific claim.
- discussed the benefits and drawbacks of a number of selection criteria for use when choosing Proof Providers from a pool of gathered volunteer devices.
- developed a method of verifying location claims using one of two weighting systems, one of which includes the concept of a device's behaviour within the system creating a reputation for that device.
- established the accuracy of the verification method through multiple system simulations.

We have enhanced the reliability of the evidence provided by a device through the development of a novel metric for use with distance bounding. This metric allows a device to distinguish between (unamplified) direct and proxied communication in an IEEE 802.11 network, an advance that is made use of in the evidence gathering stage of location verification within this work. The ability of the metric to distinguish between direct and proxied exchanges limits the possible distance between the evidence gathering device and claiming device to a single (unamplified) network hop range. We investigated the concept of a window of acceptance in time, within which a direct communcation could be completed while excluding any proxied communications. This was extended through the proposal of a pair of protocols to compute the window of acceptance for a particular area in real time, allowing the window to remain relevant despite network issues or heavy population.

Distance bounding using the binary metric has been employed to establish a verdict regarding the presence of a device in a particular area, according to a neighbouring de-

vice. As mentioned previously, by using the binary metric in this step, the proximity of a Claimant to its evidence (proof) provider is limited to only a single hop range, where the range is not amplified by specialist equipment. Anything above this is detected by the metric and flagged as a proxied exchange, essentially preventing undetectable proxy attacks. We have presented an approach to gathers this proximity evidence from multiple pre-selected devices in a Claimant's area. In order to protect the evidence gathering process, we have developed the Secure Location Verification Proof Gathering Protocol (SLVPGP). This protocol has been designed to protect the evidence supplied and any participants in the location verification process, i.e. the claiming device and any neighbouring devices supplying evidence. We developed this protocol in three layers, each with an increased level of security. Through designing the protocol in this manner, we have provided an increased level of flexibility regarding security properties and overhead costs. To our knowledge, this is the first location verification protocol designed in such a manner.

By developing such a protocol to protect the evidence gathering process, the integrity of the evidence provided is protected, as tampering can be detected and the compromised evidence discarded. Additionally, by protecting the confidential information pertaining to participants in the protocol, it is more difficult to identify who is participanting. Malicious participants are prevented from tying the identity of other participants to their unique hardware identifiers (MAC addresses) through the use of changing pseudonyms, while the SLVPGP protects against external observers. Though local observers could deduce that a device is making a location claim based on traffic and message content analysis, they are not privy to the official identity of the participants. Similarly, non-local devices that receive forwarded protocol messages are unable to distinguish who the sender and receiver are, or where they are located. Unless they are capable of tying a device's unique hardware identifier (MAC address) to a system identity, they cannot employ the information gathered without being present to analyse the situation adequately. For example, if a remote device (i.e. a device not local to the Claimant) was to discover that devices A and B were functioning as Proof Providers, it could not extract the area in which the Claimant is located as they cannot tie identities A and B to specific devices.

Using formal analysis techniques, we then investigated the security of each of the SLVPGP extensions. This analysis allowed us to investigate the possibility of vulnera-

bilities in the design of the protocols and remedy them before they could be discovered and exploited. The results of this formal analysis indicate that the security properties stated for each protocol layer are upheld, where the protocol is employed within the specified environment. Additionally, we formally analysed the security of broadcasting messages over an open network. This analysis supports the assumption that broadcasting a message is secure, provided the message contents are adequately protected.

The protection afforded to the evidence gathering process by the protocol presented in this work is based upon a fundamental assumption that a claiming device is unable to select the devices from which it gathers evidence. If this were possible, the Claimant would be capable of completely circumventing the security measures built into the protocol by controlling the evidence being supplied at the source, rather than through detectable tampering. Additionally, honest volunteers would be vulnerable to targeting for an attack on their reputation. If a malicious device wished to decrease the reputation of a specific device it knew to be willing to act as a Proof Provider, it could manipulate the selection process in some manner to ensure said device was selected as a Proof Provider for a claim. The malicious device could then work with a group of friendly devices (also selected as Proof Providers) to intentionally decrease the honest participant's reputation, thus damaging its credibility within the system. The assumption that it is not possible for a device to select those devices from which it collects evidence protects against manipulation through denying the Claimant the ability to employ only devices willing to collude against the system as Proof Providers. We have discussed a number of vulnerabilities to this form of manipulation in the SLVPGP's initialisation process. These include the gathering of volunteers for a claim and the selection criterion employed in the Proof Provider selection process. Though the possibility of Proof Providers knowingly supplying false evidence cannot be fully prevented, by addressing these vulnerabilities, the integrity of the evidence provided within the SLVPGP is further protected, as are the reputations of those devices participating in a claim.

We have presented a pair of protocols describing two approaches to volunteer gathering: Claimant-based and Verifier-based. We discussed the vulnerabilities inherent in the inclusion of the Claimant in this process. Based on this discussion, we have advocated the use of Verifier-based gathering to limit the ability of the Claimant to manipute the

process. By opting to employ this approach to volunteer gathering, the underlying system assumption regarding the Claimant's inability to select Proof Providers is reinforced.

We discussed four possible approaches to the selection of Proof Providers from a pool of volunteer devices: employing all available, selecting the most trustworthy, selecting those with the smallest overlap of overlap and finally, selecting Proof Providers at random. The selection criterion employed when choosing Proof Providers from a list of available volunteers has a great impact on the vulnerability of the verification process to manipulation. We demonstrated through simulation that due to the Claimant's inability to manipulate the selection process, the employment of all devices is most advantageous, followed by the use of random selection.

Finally, we have developed two verdict extraction methods based on the combining of evidence gathered from the selected Proof Providers using the SLVPGP. These methods, summation-based and trust-based, allow the location verification system to extract a final verdict regarding the veracity of a device's location claim. Percentage-based thresholds and a three-part scale have been employed to allow for the provision of three different verdict classifications: "possible", "unsure" and "not possible". We utilized these verdicts as inputs into a reputation system, allowing for the computation of a device's trustworthiness based on previous behaviour within the system. The accuracy of both judgement approaches presented has been demonstrated through simulations within increasingly hostile environments.

7.2 Future Work and Open Issues

The main goal of this research has been to produce a novel approach to location verification that does not rely on a pre-existing infrastructure of devices. This goal has become of significant interest in the area of location verification, with recent research turning towards ad-hoc systems in an effort to move away from infrastructure dependance. While the work presented in this dissertation is comprehensive, there remain a number of avenues for future research.

As we outlined briefly in Chapter 6, there is a possibility that the number of volunteers available for a claim may prove to be too large to utilize the "employ all available" selection approach. With the ever-increasing popularity of mobile networking devices, there may

simply be too many devices in a single area to allow for distance bounding at a reasonable speed. We intend to investigate the impact the number of Proof Providers has on the accuracy of a verdict, in order to ascertain a reasonable limit on the number to be employed on any one claim. This investigation will be an extension of the experiments described in Section 6.5.1 of Chapter 6. While those experiments established the minimum number of Proof Providers required to produce a reliable verification, continuing these experiments will investigate the point at which increasing the number of Proof Providers involved in a claim does not increase the accuracy of the verification.

This issue also ties into the impact of a heavily populated network on the accuracy of distance bounding, an issue mentioned in Chapter 3. We intend to emulate distance bounding in a more densely populated network, to investigate the accuracy of the binary metric in this situation. This will be achieved initially through the deployment of multiple other active devices within the testing network when conducting emulations, and extended to include emulations of multiple devices attempting location verification simultaneously on the same network. Additionally, we intend to establish the accuracy of the binary metric where a more aggressive proxy attack is launched. At present, the applicability of the binary metric has been proven where a proxy attack is comprised of an additional IEEE 802.11 network hop in each direction. In a more aggressive proxy attack, however, these additional hops may not incur such high time costs due to the equipment employed. While this form of proxy attack is not currently as common as the simpler proxy we have discussed, we believe that its existance should be addressed, ideally through emulations including higher speed equipment using high-speed connections. We also intend to establish the ability of the binary metric to detect the difference between direct exchanges and amplified direct exchanges, as this has not yet been addressed. This will ideally be done through emulations similar to those presented in Chapter 3. Should these investigations lead to the discovery that the binary metric is not as accurate when amplification equipment is involved, the existance of a cut off point will be investigated. By establishing a cut off point, the "acceptable" area outside the normal transmission range is limited and a new, larger acceptance area is established. This issue was not previously investigated due to a lack of equipment, a problem which also prevented the investigation of other issues over the course of this project.

A lack of available equipment has also prevented the system from being field tested on a functional VANET, in order to establish its ability to accurately verify claims in this environment. Though the system was initially designed for VANETs, the high mobility rate of nodes within these networks means that connectivity between devices may not be sustained long enough to complete evidence gathering. Therefore it is unknown whether VANETs are a suitable environment in which to deploy the system. Future investigations will deploy the system in a working VANET at various speeds and node densities to investigate its functionality.

The approach to verification described in this work relies on a central entity in order to verify a location claim. While this design does provide a high level of security and functionality, it is also structurally problematic. The Verifier represents a single point of failure within the design. If the Verifier is unavailable or overloaded, no verifications can occur. This issue can be addressed through the implementation of a hierarchy of Verifiers, each managing a specific catchment area. However, reliance upon a device not local to the claim is not ideal. Future work on this issue would seek an alternative approach to this design, allowing for an untrusted local device to provide appropriate verification of a claim.

As discussed in Chapter 6, the initialisation of the verification system's reputation engine remains an open issue. Currently there exists no method of distinguishing honest devices from dishonest devices at the point of initialisation. When a reputation engine is initialised, all participants appear equal, providing any dishonest devices with an advantage over the system. While "bootstrapping" approaches have been proposed in order to address this, they are less than ideal where trusted devices are not desired or available as part of the system. Research is ongoing in the area and while no solution has been discovered yet, work is continuing in an attempt to address the issue.

One final open issue is the extraction of a Claimant's specific location at the time of proving its claim, where the Claimant has a high level of mobility, e.g. within a VANET. The method we have discussed in Chapter 6 is suited to a low mobility situation, such as a pedestrian's handheld device. However, where a device can travel large distances in short spaces of time, the location at which it contacts one Proof Provider for distance bounding may differ greatly from its location when it contacts a second. This means

that simple multilateration cannot be relied upon to extract an accurate location. Future work to remedy this issue would result in an increased level of accuracy within the location verification system, enabling the verification and provision of a device's specific location and not merely the area within which it is currently located. The most logical adaptation of this work to achieve a solution to this issue would be the employment of simultaneous multilateration, such as that employed in the work of Capkun et al [95, 94]. The SLVPGP's current approach to proximity establishment would need to be altered in order to allow the same response to answer the distance bounding request of all Proof Providers involved, rather than each Proof Provider individually distance bounding with the Claimant. Through employing an adaptaton of this approach, proximity to all Proof Providers would be established simultaneously, removing the issue of movement between interactions.

Bibliography

[1] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, volume 1, pages 4–7, January 2000.

[2] Karl Aberer. P-grid: A self-organizing access structure for P2P information systems. In *Proceedings of the 9th International Conference on Cooperative Information Systems (CoopIS 2001)*, volume 2172, pages 179–194, September 2001.

[3] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317, 2001.

[4] Claudio A. Ardagna, Marco Cremonini, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Supporting location-based conditions in access control policies. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 212–222, 2006.

[5] IEEE Standards Association. IEEE standards for information technology – telecommunications and information exchange between systems – local and metropolitan area network – specific requirements – part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Published online, 1999.

[6] IEEE Standards Association. IEEE standards for information technology - telecommunications and information exchange between systems - local and metropolitan area network - specific requirements - part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications. Published online, 2005.

[7] Robert Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984.

[8] Abhijit V. Banerjee. The economics of rumours. In *Review of Economic Studies*, volume 60, pages 309–327, 1993.

[9] Salman A. Baset and Henning Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, April 2006.

[10] Giampaolo Bella, Fabio Massacci, and Lawrence C. Paulson. Veryifying the SET purchase protocols. Technical report, University of Cambridge, 2001.

[11] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *12th USENIX Security Symposium*, pages 15–28, August 2003.

[12] Dominique Bolignano. Towards the formal verification of electronic commerce protocols. In *CSFW '97: Proceedings of the 10th IEEE workshop on Computer Security Foundations*, page 133. IEEE Computer Society, 1997.

[13] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *Theory and Application of Cryptographic Techniques*, pages 344–359, 1993.

[14] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[15] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for mobile ad-hoc networks. Technical report, 2003.

[16] Sonja Buchegger and Jean yves Le Boudec. Performance analysis of the confidant protocol. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 226–236, 2002.

[17] Frederick Butler, Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov, and Christopher Walstad. Formal analysis of kerberos 5. *Theoretical Computer Science*, 367(1):57–87, November 2006.

[18] Stefania Cavallar, Bruce Dodson, Arjen K. Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy, Herman te Riele, Karen Aardal, Jeff Gilchrist, Gérard

Guillerm, Paul Leyland, Jöel Marchand, François Morain, Alec Muffett, Chris Putnam, Craig Putnam, and Paul Zimmermann. *Factorization of a 512-Bit RSA Modulus*, volume 1807/2000 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2000.

[19] Jolyon Clulow, Gerhard P. Hancke, Markus G. Kuhn, and Tyler Moore. So near and yet so far: Distance-bounding attacks in wireless networks. In *Security and Privacy in Ad-Hoc and Sensor Networks*, pages 83–97. Springer, March 2006.

[20] N. S. Correal, S. Kyperountas, Q. Shi, and M. Welborn. An uwb relative location system. In *Ultra Wideband Systems and Technologies, 2003 IEEE Conference on*, pages 394–397, 2003.

[21] Partha Dasgupta. *Trust as a Commodity*, chapter 4, pages 49–72. 1988.

[22] Y. Desmedt. Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them. In *Proceedings of SECURICOM'88, Sixth Worldwide Congress on Computer and Communications Security and Protection*, pages 147–159, 1988.

[23] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[24] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[25] R.C. Dorf. *The Mobile Communications Handbook*. CRC Press, Boca Raton, FL, USA, 1st edition, December 1995.

[26] John R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, 2002.

[27] P. Enge and P. Misra. Special issue on global positioning system. In *Proceedings of the IEEE*, pages 3–15. IEEE, January 1999.

[28] Kai Fan, Hui Li, and Yue Wang. Security analysis of the kerberos protocol using BAN logic. In *IAS '09: Proceedings of the 2009 Fifth International Conference on Information Assurance and Security*, volume 2, pages 467–470, 2009.

[29] William Feller. *An Introduction to Probability Theory and its Applications, vol. 1, 2nd Edition*, volume 1 of *Wiley Series in Probability and Mathematical Statistics*. Wiley-India, 1957.

[30] R. J. Fontana and S. J. Gunderson. Ultra-wideband precision asset location system. In *IEEE Conference on Ultra Wideband Systems and Technologies, 2002. Digest of Papers*, pages 147–150, 2002.

[31] R.J. Fontana, E. Richley, and J. Barney. Commercialization of an ultra wideband precision asset location system. In *2003 IEEE Conference on Ultra Wideband Systems and Technologies*, pages 369–373, November 2003.

[32] Julien Freudiger, Maxim Raya, Márk Félegyházi, Panos Papadimitratos, and Jean-Pierre Hubaux. Mix-zones for location privacy in vehicular networks. In *Workshop on Wireless Networking for Intelligent Transportation Systems (WiN-ITS) 2007*, 2007.

[33] E. Gabber and A. Wool. How to prove where you are: tracking the location of customer equipment. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 142–149. ACM, 1998.

[34] Diego Gambetta. *Can We Trust Trust?*, chapter 13, pages 213–237. 1988.

[35] Michelle Graham. A system for secure verification of location claims. *ACM SIGMOBILE Mobile Computing and Communications Review*, 12(2):47–49, April 2008.

[36] Michelle Graham and David Gray. Can you see me? the use of a binary visibility metric in distance bounding. In *Wireless Algorithms, Systems and Applications*, volume 5682, pages 378–387. Springer, August 2009.

[37] Michelle Graham and David Gray. Protecting privacy and securing the gathering of location proofs - the secure location verification proof gathering protocol. In *MobiSec*, volume 17, pages 160–171. Springer, June 2009.

[38] Michelle Graham and David Gray. Prover: A secure system for the provision of verified location information. In *Lecture Notes in Computer Science - Innovative Mobile User Interactivity (IMUI)*. Springer, October 2009.

[39] Marco Gruteser and Dirk Grunwald. Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, 10:315–325, 2005.

[40] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A reputation system for peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 144–152, June 2003.

[41] Vikram Gupta, Srikanth Krishnamurthy, and Michalis Faloutsos. Denial of service attacks at the mac layer in wireless ad hoc networks. In *Proceedings of 2002 MILCOM Conference*, volume 2, pages 1118–1123, October 2002.

[42] Neil M. Haller. The S/KEY one-time password system. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 151–157, 1994.

[43] Gerhard P. Hancke and Markus G. Kuhn. An rfid distance bounding protocol. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 67–73, Washington D.C.„ 2005. IEEE Computer Society.

[44] Gerhard P. Hancke and Markus G. Kuhn. Attacks on time-of-flight distance bounding channels. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 194–202, New York, NY, USA, 2008. ACM.

[45] Nevin Heintze, J. D. Tygar, Jeannette Wing, and H. Chi Wong. Model checking electronic commerce protocols. In *WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 147–164. USENIX Association, 1996.

[46] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice, Fourth Edition*. Springer-Verlag, 1997.

[47] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communication*, 24(2):370–380, February 2006.

[48] Leping Huang, K. Matsuura, H. Yamane, and K. Sezaki. Enhancing wireless location privacy using silent period. In *Wireless Communications and Networking Conference, 2005 IEEE*, pages 1187–1192, March 2005.

[49] Mei Lin Hui and Gavin Lowe. Safe simplifying transformations for security protocols or not just the needham schroeder pubic key protocol. *Computer Security Foundations Workshop, 1999. Proceedings of the 12th IEEE*, pages 32–43, 1999.

[50] A. Josang and R. Ismail. The beta reputation system. In *e-Reality: Constructing the Economy*, June 2002.

[51] Audun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43:618–644, 2007.

[52] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.

[53] Neal Koblitz. Elliptic curve implementation of zero-knowledge blobs. *Journal of Cryptology*, 4:207–213, January 1991.

[54] Loukas Lazos and Radha Poovendran. Serloc: secure range-independent localization for wireless sensor networks. In *WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security*, pages 21–30, 2004.

[55] Loukas Lazos and Radha Poovendran. Serloc: Robust localization for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 1(1):73–100, August 2005.

[56] Loukas Lazos, Radha Poovendran, and Srdjan Čapkun. Rope: robust position estimation in wireless sensor networks. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, pages 324–331. IEEE, 2005.

[57] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer-Verlag, January 1996.

[58] Gavin Lowe. Casper: A compiler for the analysis of security protocols. *Computer Security Foundations Workshop IEEE*, 0:18, 1997.

[59] Joseph P. Macker and M. Scott Corson. Mobile ad hoc networking and the ietf. *SIGMOBILE Mob. Comput. Commun. Rev.*, 2:9–14, 1998.

[60] Sergio Marti and Hector Garcia-molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50:472–484, March 2006.

[61] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, Nov/Dec 2001.

[62] R. N. McKean. *Economics of Trust, Altruism and Coporate Responsibility*, pages 29–43. Russell Sage Foundation, 1975.

[63] Pietro Michiardi and Refik Molva. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, volume 228, pages 107–121, 2002.

[64] Victor S. Miller. Use of elliptic curves in cryptography. In *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[65] John C. Mitchell, Arnab Roy, Paul Rowe, and Andre Scedrov. Analysis of eap-gpsk authentication protocol. In *ACNS'08: Proceedings of the 6th international conference on Applied cryptography and network security*, pages 309–327, 2008.

[66] John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern. Finite-state analysis of SSL 3.0. In *In Seventh USENIX Security Symposium*, pages 201–216, 1998.

[67] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21:993–999, December 1978.

[68] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: a P2P networking

infrastructure based on RDF. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 604–615, 2002.

[69] Wael Noureddine and Fouad Tobagi. The transmission control protocol. Technical report, http://citeseer.ist.psu.edu/noureddine02transmission.html, 2002.

[70] Rene Oosterlinck. *Tracking by Satellite: GALILEO*, chapter 5, pages 77–90. Organization for Economic Co-operation and Development (OECD), 2004.

[71] Andy Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[72] Rong Peng and Mihail L. Sichitiu. Angle of arrival localization for wireless sensor networks. In *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, 2006 (SECON '06)*, volume 1, pages 374–382, September 2006.

[73] J. Postel. User datagram protocol. *Internet Engineering Note IEN-88*, 1979.

[74] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43. ACM, 2000.

[75] Bharat Rao and Louis Minakakis. Evolution of mobile location-based services. *Communications of the ACM*, 46:61–65, 2003.

[76] T. S. Rappaport, J. H. Reed, and B. D. Woerner. Position location using wireless communications on highways of the future. *Communications Magazine, IEEE*, 34(10):33–41, 1996.

[77] Kasper Bonne Rasmussen and Srdjan Čapkun. Implications of radio fingerprinting on the security of sensor networks. In *3rd International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm 2007)*, pages 331–340, September 2007.

[78] Kasper Bonne Rasmussen and Srdjan Čapkun. Location privacy of distance bounding protocols. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 149 – 160, New York, NY, USA, 2008.

[79] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 11–21, New York, NY, USA, 2005.

[80] Jason Reid, Juan M. Gonzalez Nieto, Tee Tang, and Bouchra Senadji. Detecting relay attacks with timing-based protocols. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 204–213, New York, NY, USA, 2007.

[81] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: empirical analysis of ebay's reputation system. *The Economics of the Internet and E-Commerce, Advances in Applied Microeconomics*, 11, 2002.

[82] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[83] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. *Computer Security Foundations Workshop*, 0:98, 1995.

[84] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. Modelling and analysis of security protocols, 2001.

[85] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security*, pages 1–10, 2003.

[86] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179, 2001.

[87] Jochen Schiller and Agnès Voisard. *Location Based Services*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[88] Siraj A. Shaikh, Vicky J. Bush, and Steve A. Schneider. Specifying authentication using signal events in CSP. *Computers and Security*, 28:310–324, July 2009.

[89] Joo-Han Song, Vincent W. Wong, and Victor C. Leung. Wireless location privacy protection in vehicular ad-hoc networks. *Mobile Networks and Applications*, 15(1):160–171, February 2010.

[90] Avinash Srinivasan, Joshua Teitelbaum, and Jie Wu. Drbts: Distributed reputation-based beacon trust. In *Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*. IEEE, 2006.

[91] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.

[92] Penelope Sweetser. *Strategic decision-making with neural networks and influence maps*, pages 439–446. 2004.

[93] Joshua A. Tauber. Indoor location systems for pervasive computing. Technical report, Massachusetts Institute of Technology, August 2002.

[94] S. Čapkun and J. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications: Special Issue on Security in Wireless Ad Hoc Networks*, 24:221–232, Feb 2006.

[95] S. Čapkun and J. P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1917–1928, 2005.

[96] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Sector: secure tracking of node encounters in multi-hop wireless networks. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 21–32, New York, NY, USA, 2003.

[97] Srdjan Čapkun, Mario Cagalj, and Mani Srivastava. Secure localization with hidden and mobile base stations. In *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM 2006)*, pages 1–10, 2006.

[98] Srdjan Čapkun and Jean-Pierre Hubaux. Securing position and distance verification in wireless networks. Technical report, EFPL, 2004.

[99] Srdjan Čapkun, Kasper Rasmussen, Mario Čagalj, and Mani Srivastava. Secure location verification with hidden and mobile base stations. volume 7, pages 470–483. IEEE Educational Activities Department, April 2008.

[100] Yao Wang and Julita Vassileva. Trust and reputation model in peer-to-peer networks. In *Proceedings of the 3rd Annual International Conference on Peer-To-Peer Computing (P2P '03)*, pages 150–157, September 2003.

[101] R. Want, A. Hopper, V. Falcảo, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

[102] B. Waters and E. Felten. Secure, private proofs of location. Technical report, Princeton University, January 2003.

[103] Jeannette M. Wing. A symbiotic relationship between formal methods and security. In *Computer Security, Dependability, and Assurance: From Needs to Solutions*, pages 26–38, 1998.

[104] I. Ziskind and M. Wax. Maximum likelihood localization of multiple sources by alternating projection. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(10):1553–1560, October 1988.

Appendix A

Model Checking Casper Scripts

A.1 SLVPGP Casper scripts

A.1.1 Extension one casper script

```
#Free variables
C : Agent
V : Verifier
P, Pb :  Prover
SKAgent :  Agent -> SecretKey
PKAgent :  Agent -> PublicKey
SKProver :  Prover -> ProverSecretKey
PKProver :  Prover -> ProverPublicKey
SKVerifier :  Verifier -> VerifierSecretKey
PKVerifier :  Verifier -> VerifierPublicKey
nP, nPb, n2P, n2Pb :  Nonce
hP, hPb, h1P, h1Pb :  Hashed
xC : Location
tC, tP, tPb, tV : TimeStamp
proof, proofB : Proofs
dV : Verdict
InverseKeys = (PKAgent, SKAgent), (SKProver, PKProver),
   (PKVerifier, SKVerifier)


#Processes
```

```
Verifier(V, P, Pb, dV) knows PKAgent, PKProver, PKVerifier, SKVerifier(V)
Claimant(C, V, nP, nPb, h1P, h1Pb) knows PKAgent, PKProver,
   PKVerifier, SKAgent(C)
ProofProvider(P, V, hP, n2P, proof) knows PKProver, PKAgent,
   PKVerifier, SKProver(P)
ProofProviderB(Pb, V, hPb, n2Pb, proofB) knows PKProver, PKAgent,
   PKVerifier, SKProver(Pb)

#Protocol description
0.  -> C: xC
1.  C -> V: {C, xC}{SKAgent(C)}
2.  V -> C: {P, Pb}{SKVerifier(V)}
3.  C -> P: {P, C, {nP}{PKProver(P)}}{SKAgent(C)}
4.  C -> Pb:  {Pb, C, {nPb}{PKProver(Pb)}}{SKAgent(C)}
5.  P -> C: hP, n2P
6.  Pb -> C: hPb, n2Pb
7.  C -> P: h1P, {n2P}{SKAgent(C)}
8.  C -> Pb:  h1Pb, {n2Pb}{SKAgent(C)}
9.  P -> C: {P, tP, proof, C}{SKProver(P)} % mP
10.  Pb -> C: {Pb, tPb, proofB, C}{SKProver(Pb)} % mPb
11.  C -> V: {C, xC, tC}{SKAgent(C)}
11a.  C -> V: mP % {P, tP, proof, C}{SKProver(P)}
11b.  C -> V: mPb % {Pb, tPb, proofB, C}{SKProver(Pb)}
12.  V -> C: {dV, C, xC, tV}{SKVerifier(V)}

#Functions
symbolic PKAgent, SKAgent
symbolic PKVerifier, SKVerifier
symbolic SKProver, PKProver

#Specification
Agreement(P, V, [proof])
Agreement(Pb, V, [proofB]) -- V and PP must agree on the proof value
Agreement(V, C, [dV]) -- V & C must agree on the final value of V's verdict
```

```
#Actual variables
Claim, Eve :  Agent
Verif, Eve :  Verifier
ProofP, ProofPb, Eve :  Prover
NP, NPb, N2P, N2Pb:  Nonce
HP, HPb, H1P, H1Pb :  Hashed
XC : Location
DV : Verdict
Proof, ProofB : Proofs
TimeStamp = 0 ..  0
MaxRunTime = 0

#System
Verifier(Verif, ProofP, ProofPb, DV)
Claimant(Claim, Verif, NP, NPb, H1P, H1Pb)
ProofProvider(ProofP, Verif, HP, N2P, Proof)
ProofProviderB(ProofPb, Verif, HPb, N2Pb, ProofB)

#Intruder Information
Intruder = Eve
IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, PKAgent(Claim),
   PKProver(ProofP), PKProver(ProofPb), PKVerifier(Verif), SKAgent(Eve),
   SKProver(Eve), SKVerifier(Eve)}
```

A.1.2 Extension Two Casper Script

```
#Free variables
C : Agent
V : Verifier
P, Pb :  Prover
SKAgent :  Agent -> SecretKey
PKAgent :  Agent -> PublicKey
SKProver :  Prover -> ProverSecretKey
PKProver :  Prover -> ProverPublicKey
SKVerifier :  Verifier -> VerifierSecretKey
PKVerifier :  Verifier -> VerifierPublicKey
```

```
nP, nPb, n2P, n2Pb :  Nonce
hP, hPb, h1P, h1Pb :  Hashed
xC : Location
tC, tP, tPb, tV : TimeStamp
proof, proofB : Proofs
dV : Verdict
InverseKeys = (PKAgent, SKAgent), (SKProver, PKProver),
   (PKVerifier, SKVerifier)
```

#Processes

```
Verifier(V, P, Pb, dV) knows PKAgent, PKProver, PKVerifier, SKVerifier(V)
Claimant(C, V, nP, nPb, h1P, h1Pb) knows PKAgent, PKProver,
   PKVerifier, SKAgent(C)
ProofProvider(P, V, hP, n2P, proof) knows PKProver, PKAgent,
   PKVerifier, SKProver(P)
ProofProviderB(Pb, V, hPb, n2Pb, proofB) knows PKProver, PKAgent,
   PKVerifier, SKProver(Pb)
```

#Protocol description

```
0.  -> C: xC
1.  C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}
2.  V -> C: {{P, Pb}{SKVerifier(V)}}{PKAgent(C)}
3.  C -> P: {{P, C, nP}{SKAgent(C)}}{PKProver(P)}
4.  C -> Pb:  {{Pb, C, nPb}{SKAgent(C)}}{PKProver(Pb)}
5.  P -> C: hP, n2P
6.  Pb -> C: hPb, n2Pb
7.  C -> P: h1P, {n2P}{SKAgent(C)}
8.  C -> Pb:  h1Pb, {n2Pb}{SKAgent(C)}
9.  P -> C: {{P, tP, proof, C}{SKProver(P)} % mP}{PKAgent(C)}
10.  Pb -> C: {{Pb, tPb, proofB, C}{SKProver(Pb)} % mPb}{PKAgent(C)}
11.  C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}
11a.  C -> V: {{mP % {P, tP, proof, C}{SKProver(P)}}{SKAgent(C)}}
   {PKVerifier(V)}
11b.  C -> V: {{mPb % {Pb, tPb, proofB, C}{SKProver(Pb)}}{SKAgent(C)}}
   {PKVerifier(V)}
```

```
12.  V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}
```

#Functions

```
symbolic PKAgent, SKAgent
symbolic SKProver, PKProver
symbolic SKVerifier, PKVerifier
```

#Specification

```
Agreement(P, V, [proof])
Agreement(Pb, V, [proofB])
Agreement(V, C, [dV])
Secret(C, C, [V, P, Pb])
Secret(P, P, [V, C])
Secret(Pb, Pb, [V, C])
Secret(C, xC, [V])
Secret(P, proof, [V, C])
Secret(Pb, proofB, [V, C])
```

#Actual variables

```
Claim, Eve :  Agent
Verif, Eve :  Verifier
ProofP, ProofPb, Eve :  Prover
NP, NPb, N2P, N2Pb:  Nonce
HP, HPb, H1P, H1Pb :  Hashed
XC : Location
DV : Verdict
Proof, ProofB : Proofs
TimeStamp = 0 ..  0
MaxRunTime = 0
```

#System

```
Verifier(Verif, ProofP, ProofPb, DV)
Claimant(Claim, Verif, NP, NPb, H1P, H1Pb)
ProofProvider(ProofP, Verif, HP, N2P, Proof)
```

```
ProofProviderB(ProofPb, Verif, HPb, N2Pb, ProofB)

#Intruder Information
Intruder = Eve
IntruderKnowledge = {Verif, PKAgent, PKVerifier, PKProver, PKAgent(Claim),
   PKVerifier(Verif), PKProver(ProofP), PKProver(ProofPb), SKAgent(Eve),
   SKProver(Eve), SKVerifier(Eve)}
```

A.1.3 Extension Three Casper Script

```
#Free variables
O : OracleType
C : Agent
V : Verifier
P, Pb :  Prover
SKAgent :  Agent -> SecretKey
PKAgent :  Agent -> PublicKey
SKProver :  Prover -> ProverSecretKey
PKProver :  Prover -> ProverPublicKey
SKVerifier :  Verifier -> VerifierSecretKey
PKVerifier :  Verifier -> VerifierPublicKey
nP, nPb, n2P, n2Pb :  Nonce
hP, hPb, h1P, h1Pb :  Hashed
xC : Location
tC, tP, tPb, tV : TimeStamp
proof, proofB : Proofs
dV : Verdict
InverseKeys = (PKAgent, SKAgent), (SKProver, PKProver),
   (PKVerifier, SKVerifier)

#Processes
Verifier(V, P, Pb, kCP, kCPb, dV, nP, n1P, nPb, n1Pb) knows PKAgent,
   PKProver, PKVerifier, SKVerifier(V)
Claimant(C, V, O, h1P, h1Pb) knows PKAgent, PKProver, PKVerifier, SKAgent(C)
Oracle(O, C, V, P, Pb) knows PKAgent, PKProver, PKVerifier
ProofProvider(P, V, proof, hP, n2P) knows PKProver, PKAgent,
```

```
   PKVerifier, SKProver(P)
ProofProviderB(Pb, V, proofB, hPb, n2Pb) knows PKProver, PKAgent,
   PKVerifier, SKProver(Pb)

#Protocol description
0.  -> C: xC
1.  C -> V: {{C, xC}{SKAgent(C)}}{PKVerifier(V)}
2.  V -> C: {{nP, n1P, kCP, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKAgent(C)}
3.  V -> P: {{{P, nP, n1P, kCP}{SKVerifier(V)}}{PKProver(P)}}{SKVerifier(V)}
4.  V -> Pb:  {{{Pb, nPb, n1Pb, kCPb}{SKVerifier(V)}}{PKProver(Pb)}}
   {SKVerifier(V)}
5.  C -> O: {C}{PKVerifier(V)} % mCP
5a.  O -> P: {C}{PKVerifier(V)} % mCP
6.  C -> O: {C}{PKVerifier(V)} % mCPb
6a.  O -> Pb:  {C}{PKVerifier(V)} % mCPb
7.  P -> O: hP, n2P
7a.  O -> C: hP, n2P
8.  Pb -> O: hPb, n2Pb
8a.  O -> C: hPb, n2Pb
9.  C -> O: h1P, {n2P}{SKAgent(C)} % mCPN
9a.  O -> P: h1P, mCPN % ({n2P}{SKAgent(C)} % mCPN1)
10.  C -> O: h1Pb, {n2Pb}{SKAgent(C)} % mCPbN
10a.  O -> Pb:  h1Pb, mCPbN % ({n2Pb}{SKAgent(C)} % mCPbN1)
11.  P -> O: h(kCP) % mCPh, {nP, {{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1),
   tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}
   {PKVerifier(V)} % mPV}{kCP} % mCPk
11a.  O -> C: mCPh % h(kCP), mCPk % {nP, mPV % ({{P, mCPN % ({n2P}{SKAgent(C)}
   % mCPN1), tP, proof, mCP % ({C}{PKVerifier(V)} % mCP1)}{SKProver(P)}}
   {PKVerifier(V)} % mPV1)}{kCP}
12.  Pb -> O: h(kCPb) % mCPbh, {nPb, {{Pb, mCPbN % ({n2Pb}{SKAgent(C)} %
   mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)}{SKProver(Pb)}}
   {PKVerifier(V)} % mPbV}{kCPb} % mCPbk
12a.  O -> C: mCPbh % h(kCPb), mCPbk % {nPb, mPbV % ({{Pb, mCPbN % ({n2Pb}
   {SKAgent(C)} % mCPbN1), tPb, proofB, mCPb % ({C}{PKVerifier(V)} % mCPb1)}
   {SKProver(Pb)}}{PKVerifier(V)} % mPbV1)}{kCPb}
```

```
13.  C -> V: {{C, tC, xC}{SKAgent(C)}}{PKVerifier(V)}
13a.  C -> V: {mPV1 % ({{P, mCPN % ({n2P}{SKAgent(C)} % mCPN1), tP, proof,
   mCP1 % {C}{PKVerifier(V)}}{SKProver(P)}}{PKVerifier(V)})}{SKAgent(C)}
13b.  C -> V: {mPbV1 % ({{Pb, mCPbN % ({n2Pb}{SKAgent(C)} % mCPbN1), tPb,
   proofB, mCPb1 % {C}{PKVerifier(V)}}{SKProver(Pb)}}
   {PKVerifier(V)})}{SKAgent(C)}
14.  V -> C: {{dV, xC, C, tV}{SKVerifier(V)}}{PKAgent(C)}
```

#Functions

```
symbolic PKAgent, SKAgent
symbolic SKProver, PKProver
symbolic SKVerifier, PKVerifier
```

#Specification

```
Agreement(P, V, [proof])
Agreement(Pb, V, [proofB])
StrongSecret(C, C, [O, V])
StrongSecret(P, P, [O, V])
StrongSecret(Pb, Pb, [O, V])
StrongSecret(C, xC, [V])
StrongSecret(P, proof, [V])
StrongSecret(Pb, proofB, [V])
```

#Actual variables

```
TheOracle :  OracleType
Claim, Eve :  Agent
Verif, Eve :  Verifier
ProofP, ProofPb, Eve :  Prover
NP, NPb, N1P, N1Pb, N2P, N2Pb:  Nonce
HP, HPb, H1P, H1Pb :  Hashed
XC : Location
DV : Verdict
Proof, ProofB : Proofs
TimeStamp = 0 ..  0
```

```
MaxRunTime = 0

#System
Verifier(Verif, ProofP, ProofPb, KCP, KCPb, DV, NP, N1P, NPb, N1Pb)
Claimant(Claim, Verif, TheOracle, H1P, H1Pb)
Oracle(TheOracle, Claim, Verif, ProofP, ProofPb)
ProofProvider(ProofP, Verif, Proof, N2P, HP)
ProofProviderB(ProofPb, Verif, ProofB, N2Pb, HPb)

#Intruder Information
Intruder = Eve
IntruderKnowledge = {Verif, TheOracle, PKAgent, PKAgent(Claim), PKProver,
   PKProver(ProofP), PKProver(ProofPb), PKVerifier, PKVerifier(Verif),
   SKAgent(Eve), SKProver(Eve), SKVerifier(Eve)}
```

A.1.4 Alternative Intruder Information

A.1.4.1 Alternative intruder information for extensions one and two

Known C

```
IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, SKAgent(Eve),
SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP),
PKProver(ProofPb), SKAgent(Claim)}
```

Known C/P

```
IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, SKAgent(Eve),
SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP),
PKProver(ProofPb), SKAgent(Claim), SKProver(ProofP)}
```

Known C/Pb

```
IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, SKAgent(Eve),
SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP),
PKProver(ProofPb), SKAgent(Claim), SKProver(ProofPb)}
```

Known C/P/Pb

```
IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, SKAgent(Eve),
```

SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP),
PKProver(ProofPb), SKAgent(Claim), SKProver(ProofP), SKProver(ProofPb)}

Known P

IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, SKAgent(Eve),
SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP),
PKProver(ProofPb), SKProver(ProofP)}

Known Pb

IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, SKAgent(Eve),
SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP),
PKProver(ProofPb), SKProver(ProofPb)}

Known P/Pb

IntruderKnowledge = {Verif, PKAgent, PKProver, PKVerifier, SKAgent(Eve),
SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP),
PKProver(ProofPb), SKProver(ProofP), SKProver(ProofPb)}

A.1.4.2 Alternative intruder information for extension three

Known C

IntruderKnowledge = {Verif, TheOracle, PKAgent, PKProver, PKVerifier,
SKAgent(Eve), SKProver(Eve), SKVerifier(Eve), PKAgent(Claim),
PKProver(ProofP), PKProver(ProofPb), SKAgent(Claim)}

Known C/P

IntruderKnowledge = {Verif, TheOracle, PKAgent, PKProver, PKVerifier,
SKAgent(Eve), SKProver(Eve), SKVerifier(Eve), PKAgent(Claim),
PKProver(ProofP), PKProver(ProofPb), SKAgent(Claim), SKProver(ProofP)}

Known C/Pb

IntruderKnowledge = {Verif, TheOracle, PKAgent, PKProver, PKVerifier,
SKAgent(Eve), SKProver(Eve), SKVerifier(Eve), PKAgent(Claim),
PKProver(ProofP), PKProver(ProofPb), SKAgent(Claim), SKProver(ProofPb)}

Known C/P/Pb

IntruderKnowledge = {Verif, TheOracle, PKAgent, PKProver, PKVerifier,
SKAgent(Eve), SKProver(Eve), SKVerifier(Eve), PKAgent(Claim),

PKProver(ProofP), PKProver(ProofPb), SKAgent(Claim), SKProver(ProofP), SKProver(ProofPb)}

Known P

IntruderKnowledge = {Verif, TheOracle, PKAgent, PKProver, PKVerifier, SKAgent(Eve), SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP), PKProver(ProofPb), SKProver(ProofP)}

Known Pb

IntruderKnowledge = {Verif, TheOracle, PKAgent, PKProver, PKVerifier, SKAgent(Eve), SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP), PKProver(ProofPb), SKProver(ProofPb)}

Known P/Pb

IntruderKnowledge = {Verif, TheOracle, PKAgent, PKProver, PKVerifier, SKAgent(Eve), SKProver(Eve), SKVerifier(Eve), PKAgent(Claim), PKProver(ProofP), PKProver(ProofPb), SKProver(ProofP), SKProver(ProofPb)}

A.2 Broadcasting Casper scripts

A.2.1 Basic Casper Script

```
#Free variables
S : ServerType
A, B, C : Agent
O : OracleType
SK : Agent -> SecretKey
PK : Agent -> PublicKey
mB : MessageBob
mC : MessageCeline
InverseKeys = (PK, SK)

#Processes
ServerProc(S, A, B, C, O) knows PK
Initiator(A, O, S) knows PK, SK(A)
RecipientB(B, O, S) knows PK, SK(B)
RecipientC(C, O, S) knows PK, SK(C)
```

```
OracleProc(O, A, B, C, S) knows PK

#Protocol description (i)
0.  -> S: mB, mC
1.  S -> A: {mB}{PK(B)} % nMb, {mC}{PK(C)} % nMc
2.  A -> O: nMb % ({mB}{PK(B)} % nMb1)
#Protocol description (ii)
3.  O -> B: nMb1 % {mB}{PK(B)}
4.  A -> O: nMc % ({mC}{PK(C)} % nMc1)
5.  O -> C: nMc1 % {mC}{PK(C)}

#Functions
symbolic PK, SK

#Specification
Agreement(S, B, [mB])
Agreement(S, C, [mC])
Secret(A, A, [O])
Secret(B, B, [O])
Secret(C, C, [O])
Secret(S, mB, [B])
Secret(S, mC, [C])

#Actual variables
Alice, Bob, Celine, Eve :  Agent
Oracle :  OracleType
Server :  ServerType
MB : MessageBob
MC : MessageCeline

#System
ServerProc(Server, Alice, Bob, Celine, Oracle)
Initiator(Alice, Oracle, Server)
RecipientB(Bob, Oracle, Server)
RecipientC(Celine, Oracle, Server)
```

```
OracleProc(Oracle, Alice, Bob, Celine, Server)


#Intruder Information
Intruder = Eve
IntruderKnowledge = {Server, Oracle, PK, SK(Eve)}
```

A.2.2 Additional Intruder Information

Known A

```
IntruderKnowledge = {Server, Oracle, PK, SK(Eve), PK(Alice), SK(Alice)}
```

Known B

```
IntruderKnowledge = {Server, Oracle, PK, SK(Eve), PK(Bob), SK(Bob)}
```

Known C

```
IntruderKnowledge = {Server, Oracle, PK, SK(Eve), PK(Celine), SK(Celine)}
```

Known A/B

```
IntruderKnowledge = {Server, Oracle, PK, SK(Eve), PK(Bob), SK(Bob), PK(Alice),
SK(Alice)}
```

Known A/C

```
IntruderKnowledge = {Server, Oracle, PK, SK(Eve), PK(Celine), SK(Celine),
PK(Alice), SK(Alice)}
```

Appendix B

Verification Simulations

B.1 Proof Provider Selection Algorithms

The algorithms outlined in this section (algorithms 8 - 11) describe the different approaches taken by the Verifier when selecting Proof Providers based on a specific criterion. These approaches are employed within the verification simulations discussed in Section 6.7.

Algorithm 8 Selection Method 1 - All Available Volunteers

```
set current_volunteer to first in volunteer_pool
for all volunteers in volunteer_pool
   add current_volunteer to selected
   set current_volunteer to next in volunteer_pool
```

Algorithm 9 Selection Method 2 - Most Trustworthy Volunteers

```
for all volunteers in volunteer_pool
   get ProofProvider trustworthiness of current_volunteer
sort volunteers by trustworthiness
if number of volunteers <= maximum allowable Proof Providers then
   add all volunteers to selected_volunteers
else
   set most trusted in pool as current_volunteer
   while number_selected < maximum allowable Proof Providers
      add current_volunteer to selected_volunteers
      remove current_volunteer from pool
      set next most trusted volunteer in pool as current_volunteer
```

Algorithm 10 Selection Method 3 - Most Suitably Located Volunteers

```
if number of volunteers <= maximum allowable Proof Providers then
  add all volunteers to selected_volunteers
else
  while number in selected_volunteers < maximum allowable Proof Providers
    sort volunteers by location furthest west relative to claimed location
    set volunteer furthest west as current_volunteer
    if number in selected_volunteers < maximum allowable Proof Providers then
      while current_volunteer not added
        if current_volunteer not in selected_volunteers then
          add current_volunteer to selected_volunteers
        else
          set next furthest volunteer west as current_volunteer
    sort volunteers by location furthest east relative to claimed location
    set volunteer furthest east as current_volunteer
    if number in selected_volunteers < maximum allowable Proof Providers then
      while current_volunteer not added
        if current_volunteer not in selected_volunteers then
          add current_volunteer to selected_volunteers
        else
          set next furthest volunteer east as current_volunteer
    sort volunteers by location furthest north relative to claimed location
    set volunteer furthest north as current_volunteer
    if number in selected_volunteers < maximum allowable Proof Providers then
      while current_volunteer not added
        if current_volunteer not in selected_volunteers then
          add current_volunteer to selected_volunteers
        else
          set next furthest volunteer north as current_volunteer
    sort volunteers by location furthest south relative to claimed location
    set volunteer furthest south as current_volunteer
    if number in selected_volunteers < maximum allowable Proof Providers then
      while current_volunteer not added
        if current_volunteer not in selected_volunteers then
          add current_volunteer to selected_volunteers
        else
          set next furthest volunteer south as current_volunteer
```

Algorithm 11 Selection Method 4 - Random Subset of Volunteers

```
if number in volunteer pool <= maximum allowable Proof Providers then
  add all volunteers to selected_volunteers
else
  while number in selected_volunteers < maximum allowable Proof Providers then
    generate random number X
    if volunteer_X not in selected_volunteers then
      add volunteer_X to selected_volunteers
```

Algorithm 12 Counter-Algorithm for Selection Methods 1 & 4 - All Available & Randomly Selected Volunteers

for all locations in range of added_device
  increase location_value by 1
  if added_device is friend **then**
    increase location_value by 1
  else
    compute distance between x coordinates
    compute distance between y coordinates
    decrease location_value by $\left(\frac{1}{x\_distance*0.75} + \frac{1}{y\_distance*0.75}\right)$

B.2 Malicious Counter-Algorithms

The algorithms outlined in this section describe the different approaches taken by a malicious device to compute the optimum false location to be claimed within its environment. These approaches are performed by the influence map of a device at initialisation, or when another device updates its location. They are designed to counter the approaches employed by the Verifier when selecting Proof Providers in that manner, and are employed within the verification simulations discussed in Section 6.7.

When a device is initialised, it loads all information it possesses regarding other devices in the world into its influence map. The influence map attempts to compute the optimal false location to be claimed using this information. For each device added to the world, the influence map employs one of the counter-algorithms described here to gauge the impact of that device on its surrounding area. Each location within the map is assigned a value, and the values of those locations within range of the device being added are updated using a specific counter algorithm. The influence map can then compare each of the stored values to provide the optimal location for use in a false claim.

Algorithm 13 Counter-Algorithm for Selection Method 2 - Most Trustworthy Volunteers

for all locations in range of added_device
  increase location_value by 1
  if added_device is friend **then**
    increase location_value by added_device_trustworthiness
  else
    decrease location_value by added_device_trustworthiness

Algorithm 14 Counter-Algorithm for Selection Method 3 - Most Suitably Located Volunteers

for all locations in range of added_device
  increase location_value by 1
  if device is farthest known device from location **then**
    if added_device is friend **then**
      increase location_value by farthest_weight
    else
      decrease location_value by farthest_weight
  elseif added_device is friend **then**
    increase location_value by 1
  else
    compute distance between x coordinates
    compute distance between y coordinates
    if distance to location_value **is not equal to** device_range then
      decrease location_value by $\left(\frac{1}{(range-x\_distance)*0.5} + \frac{1}{(range-y\_distance)*0.5}\right)$
    else
      decrease location_value by 1