ROBUST DECENTRALIZED AUTHENTICATION FOR PUBLIC KEYS AND GEOGRAPHIC LOCATION

BY VIVEK PATHAK

A dissertation submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
Graduate Program in Computer Science

Written under the direction of
Liviu Iftode
and approved by

_____

_____

_____

_____

New Brunswick, New Jersey
January, 2009

ABSTRACT OF THE DISSERTATION

Robust Decentralized Authentication for Public Keys and Geographic Location

by Vivek Pathak
Dissertation Director: Liviu Iftode

Authentication has traditionally been done either in a decentralized manner with human assistance or automatically through a centralized security infrastructure. In the security infrastructure approach, a central trusted authority takes on the responsibility of authenticating participants within its domain of control. While the security infrastructure approach works well in traditional organizations, it does not address the needs of open membership systems.

We propose automatic decentralized authentication mechanisms for peer-to-peer systems, email systems, and ad-hoc networks. Our byzantine fault tolerant public-key authentication protocol (BPKA) provides decentralized authentication to peer-to-peer systems with honest majority. Authentication is done over an insecure asynchronous network without using trusted third parties or human input. We also authenticate public keys in the email environment through our social-group key authentication protocol (SGKA). The protocol provides end-to-end authentication at the email client without using infrastructure or centralized authorities. Finally, location authentication in ad-hoc networks is proposed through our geographical secure path routing protocol (GSPR). The protocol authenticates geographic locations of anonymous nodes in order to provide location authentication and anonymity simultaneously.

Acknowledgements

I would like to thank my adviser, Liviu Iftode for his support, advice, and encouragement. His ongoing help made it possible to overcome many unexpected twists and turns in my research work. I am also thankful to my co-adviser and committee member, Danfeng Yao, whose insight and advice was instrumental in overcoming a number of research problems. I am also thankful to my committee members, Vinod Ganapathy and Roberto Tamassia for their valuable suggestions. I wish to thank all the Rutgers professors for sharing their learning. I am also thankful to all the members of Discolab, and in particular to Florin Sultan, Lu Han, and Mohan Dhawan, for their suggestions and help. I am also grateful to Naftaly Minsky, Marios Dikaiakos, Tao Yang, Thu Nguyen, and Tomasz Imielinski for their advice and help. Finally, I am also very thankful to my wife and my parents for their continuous support and encouragement. Their sacrifices and support were vital for the completion of this work.

Table of Contents

Chapter 1

Introduction

Information security has been an area of great interest even before the use of digital computers. With the mainstream use of digital computers, a number of information security goals have been studied and applied in various application settings. This dissertation focuses on the security goals of authentication, fault tolerance, and privacy in modern computing environments.

1.1 What is authentication

Authentication is a term that is used in a number of related ways. It is generally used to provide assurance about the veracity of a claim. The claim and the assurance depend on the security goal, the application domain, the authentication procedure, and the underlying assumptions. Authentication serves the critical function of imparting trust in traditional documents like currency or contracts, where it is typically achieved through physical security features like watermarks or seals. Authentication also serves an important function in the contemporary networked computing environment. It is necessary to identify participants in order to avoid impersonation attacks in a controlled access scenario. Even in open access environments, where any one is free to join, participants have to be authenticated for the purpose of tracking reputation, managing friends, enforcing resource usage limits, and managing ongoing transactions. The rise of ubiquitous and mobile computing makes location a useful attribute to authenticate. Identity and location authentication are important in modern computing scenarios.

A number of attributes can be authenticated depending on the application domain and its information security needs. Identity authentication, or identification, is useful in an access control scenario where privilege and authorization depend on identity. Identity authentication is achieved either by verifying a password, which is expected to be private, or by checking for identity directly. Direct identity authentication is done either by using human judgment or through automatic biometric authentication. Open access networking environments like Email or the Internet do not restrict participation based on identity. However, authenticating the contents and senders of received messages are desirable security goals. Message authentication provides the assurance that the message contents have not been modified en-route. Sender authentication provides the assurance that the apparent sender is indeed the real sender of the message. Sender and content authentication are the basic network security goals.

A-priori relationships are needed to take advantage of sender authentication. If the communicating parties do not have an a-priori relationship, it is useful to authenticate the sender indirectly through a popular or reliable recommender. This is traditionally done through public key authentication. Authentication through public keys simplifies the authentication procedure by relying on the judgment of relatively few recommenders to vouch for the numerous identities. Authentication of the message source and content is typically achieved by verifying the digital signature on the message. The digital signature is created with the private key of the sender. Public key authentication is a fundamental problem of digital security because authenticated public keys are required for verifying the integrity of digital signatures [23, 80]. Just as authentication through reference helps in making access control choices by establishing "who knows you", location authentication helps by making these decisions based on "where you are". Location authentication is meaningful in practical situations because certain privileges are expected by virtue of being in particular locations. Location authentication permits richer access control policies by using location as an input to the trust deduction.

Authentication mechanisms differ in the properties of the authentication achieved. The level of robustness against defective or malicious participants differs among the authentication mechanisms. Another area of difference is having or avoiding single points of failure. Authentication mechanisms can also be distinguished in terms of their requirements. They may impose different computational and storage costs. Participants may also face risks like the loss of privacy. The authentication mechanisms may also differ in the assumptions and the threats they are able to withstand.

1.2 Why is trust needed

The assurance achieved through authentication depends on the trust assumptions underlying the authentication procedure. Centralized trust assumptions assume that one or more centralized players are competent and honest. Centralized trusted third parties are used for traditional organizational settings where the centralized trust model is meaningful. Authentication mechanisms may also explicitly depend on human evaluations of trustworthiness. This assumption affects whether the authentication mechanism is automatic or human-aided, thereby imposing different demands on user sophistication.

The advent of cloud computing and social networking on the Internet has led to a new class of authentication needs. The large-scale and open nature of these systems prohibits the usage of centralized trust assumptions and central trusted parties. This is because there may be no possible third party that can be trusted by all the participants. These upcoming systems are increasingly being used by technologically unsophisticated users. Thus, the authentication needs of the future must address decentralization and automation simultaneously.

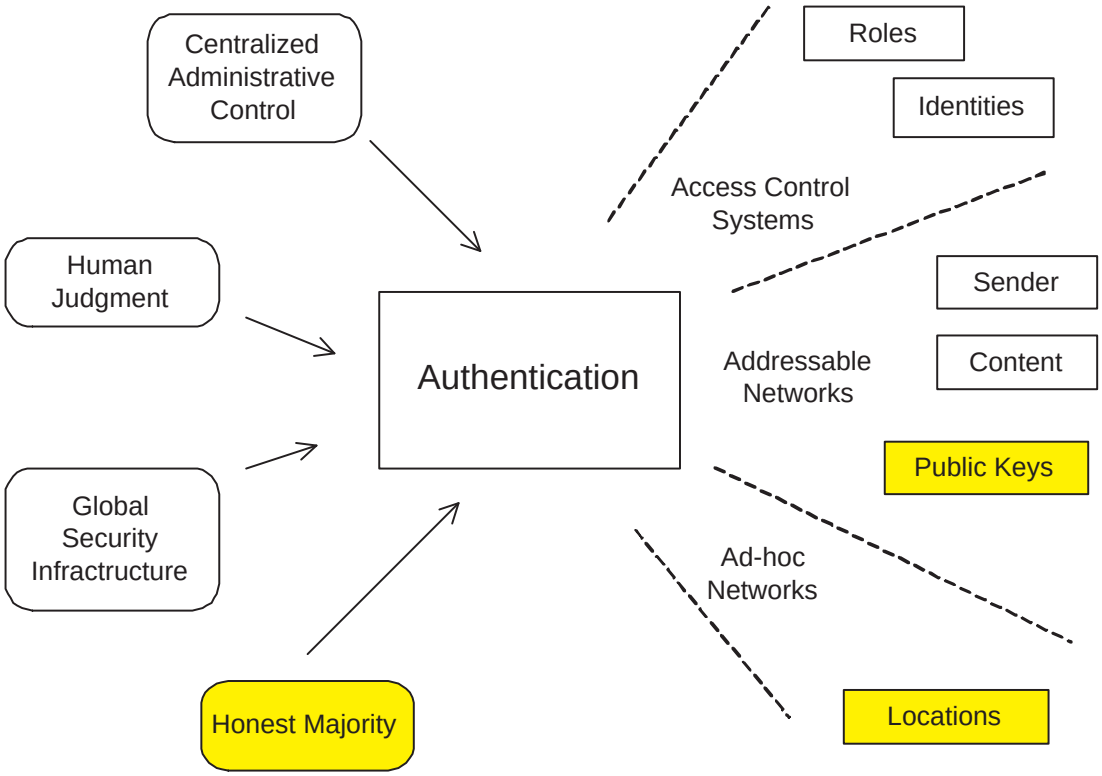

Figure 1.1: Overview of trust assumptions and authentication objectives in various application areas. The contribution area of this dissertation is indicated in highlighted boxes.

1.3 Thesis

Robust decentralized authentication of public keys and geographic locations can be achieved by assuming the presence of honest majority instead of relying on centralized security infrastructure or human trust judgment.

1.4 Authentication in addressable networks

Participants in traditional networks are identified through network addresses. Various authentication approaches have been proposed for these networks. The level of decentralization, need for human input, and tolerance of malicious behavior are the solution space dimensions outlined in this section.

1.4.1 Identity authentication

Identity authentication is used for access control typically in smaller networks where all the participants can be identified by a central trusted administrator. In this small network scenario,

the central administrator is also responsible for setting the access control policy and implementing the access control mechanism. Examples of identity based authentication include password based authentication in the UNIX operating system, and biometrics based identity authentication for high security installations [88]. The basic principle in these applications is that access can be completely determined purely based on identity, and that evaluation of the access function and its enforcement is done by a central trusted authority.

Identity based authentication has limitations. Using a central administrator to maintain policy and to enforce mechanism limits the number of participants in these networks because of the management overhead. Administering such a system may also require significant human input making it feasible either for high value security applications or for small networks with few participants. Identity based authentication is therefore found only in critical, specialized, or small applications. Since both policy, mechanism, and membership are controlled from a single logical administrator, this form of authentication is perhaps the simplest and most effective where applicable.

Research in identity authentication has focused on addressing the limitations of identity authentication. Improvement of identification based access control in order to address its limitations. The issue of naming and access control across administrative boundaries has been addressed by Howell by assuming certificate infrastructure [40, 41]. Another way of crossing administrative boundaries for a global file system has been proposed by Kaminsky et. al. [47]. This approach aggregates the administrative authorization decisions of multiple authentication servers into a single server that decides on access control. While these efforts expand the envelope of applicability for identity based access control, they do not address the basic limitations of identity based access control. These limitations include having a fundamental capability separation between the set of identities and the set of administrators. This makes identity authentication less suitable for applications with dynamic membership and open access.

1.4.2 Role based access

The abstraction of roles has been proposed to address the difficulty of identity based access control systems. Their popularity both in research and practice stems from the fact that roles are usually more stable in organizations than the individuals that occupy them. Sandhu et. al. formalized and analyzed the issues with role based access control [90]. Later research has attempted to generalize and formalize role based access control for widespread use [28, 89]. Roles are meaningful only within the context of given access control policies, which must be centrally controlled by a dedicated administrative staff. Ongoing administrative effort is also required for maintaining user membership within the roles in order to reflect organizational goals. These requirements limit the applicability of role based access control to traditional organizational settings.

Application of role based access control to distributed systems has been an area of research

interest. Bacon et. al. propose the OASIS architecture for implementing role based access control in distributed systems [10]. Their proposal increases the flexibility of access control specification by using an appointment mechanism and parametrized roles. However, their mechanism depends on the existence of pre-existing certificate infrastructure. Li et. al. propose a language oriented role based trust management mechanism for decentralized access control [63]. While their solution depends on pre-existing trust relationships between collaborating administrative players, it is highly scalable on the number of trusted beneficiaries. Dependency on pre-existing security infrastructure makes it hard to use these systems in modern open computing environments like the Internet.

1.4.3 Reputation based access

Using reputations of identities rather than identities themselves is an alternative way to grant privileges in distributed systems. This approach not only simplifies access policy, but also enhances participant privacy because reputations can be tracked for pseudonyms instead of real identities [5, 21, 48]. Although reputation is a useful simplifying abstraction, it depends on the existence of an underlying secure identification layer so that the reputation of one identity may not be used by another. Reputation based distributed trust has been investigated by a number of previous works. The Free Haven project uses a proactive mechanism based on recommendations to protect anonymity of the users [24]. The NICE platform allows the formation of co-operative groups in a peer-to-peer system through a reputation based mechanism [58]. Reputation oriented systems typically operate in the context of easily verifiable services that allow participants to easily compute robust reputations. This limits their application to easily verifiable services like peer-to-peer file sharing.

1.4.4 Cryptographic identifiers

Cryptographic identifiers is an alternative approach to identity authentication. It depends on ensuring that the the public key of a network end-point is related to its networking address, thereby eliminating the need for a certificate authority. The original proposal for solving the sender authentication problem through cryptographically verifiable identifiers came from Shamir [92]. In this scheme, the well known network address or identity was used as the public key while the private keys were generated by a secure key distribution center. The key distribution center limitation was addressed by allowing part of the network address to be used for cryptographic identifier as done by [69]. The basic idea behind the cryptographic identifiers is that network address depends on the public key of the network end-point.

Cryptographic identifiers are appropriate in applications where either the network addresses can be chosen as a functions of the self selected public keys or in applications where it is feasible to set up a secure key distribution center. In the former case, it requires support from

the networking infrastructure, and in the latter from a competent and reliable key distribution center. Thus, its applicability is restricted to advanced networks or to high value applications.

1.4.5 Certificate authorities

Certificate authorities are trusted third parties used for authenticating public keys. The trusted third party model requires that the certifying authority is trusted by all the participants. Certificate authorities issue digitally signed public key certificates that associate a public key to an identity. Verification of the digital public key certificate confirms that if the certificate authority is honest and capable, then the public key belongs to the identity stated in the certificate. This authentication architecture is hierarchically extended to create Public key infrastructures in the X.509 standard [19]. In this architecture, certificate authorities can issue digital certificates to other certificate authorities, which are placed lower in the the certification hierarchy. The special root certificate authority is trusted by everyone.

Trusted third parties are well suited for a client-server computing model where a few centralized service providers are granted public key certificates by the certificate authorities. However, they are not suitable to peer-to-peer systems for a number of reasons. It may be impossible to find sufficiently trusted parties in heterogeneous systems. Trusted third parties must support certificate revocation to prevent misuse of compromised private keys [70, 31, 6]. The off-line advantage of certificate authorities is reduced by the overhead of maintaining fresh revocation information. Considering the consistency and timely propagation issues imposed by the mechanism, and the administrative burden placed by the security policy, it may not be possible to scale up the centralized authentication mechanism for securing large systems like the Internet.

Public key infrastructures and certificate authorities have been the focus of intense research for more than two decades. Recent research directions in traditional network settings have included increasing the efficiency of public key infrastructures in large deployments. The NPKI system uses nested certificates for highly efficient certificate validation in hierarchically deep trust paths [61]. Increasing the efficiency of certificate revocation has also been a track of improvement [6, 62]. Another area of research has been the investigation of trust issues underlying practical public key infrastructure deployments [60]. Although public key infrastructures have had some success within organizational contexts, they do not fit the trust needs of large scale open distributed systems.

1.4.6 Threshold cryptography

Threshold cryptography is an approach designed to improve the robustness of cryptographic operations by distributing secret information among multiple parties. Usually a number of these parties have to co-operate in order to use the secret information. Threshold cryptography also ensures that an attacker can not recover the secret without compromising more than k of

the secret shares. Threshold cryptography has been used to create robust certificate authorities. Goldwasser et. al. present a method where it is impossible to create a digitally signed public key certificate without having a quorum of participants [35]. As a consequence, unless the number of malicious parties is as large as the quorum, false authentication is impossible. Threshold cryptography requires the existence of a trusted dealer that initializes the key shares. In this way, it depends on the honesty of the dealer. Threshold cryptography has been applied in COCA, a fault tolerant public key authentication service [107]. It is also the basis of a number of other secure services [15, 84, 106].

Threshold cryptography is also used to implement proactive recovery. To compromise such a system, the adversary is required to compromise the quorum within its vulnerability window or lose any previous progress due to a re-randomization of key shares [16, 39]. Although better than static secret shares, these scheme cannot recover from the compromise of a quorum because the same long term shared secret is recycled among the trusted parties. Although the signature procedure is distributed, the procedural aspects of granting a digital certificate (like verifying the identity is legitimate, has a real address, etc.) are still logically centralized. Therefore, while they improve robustness, threshold cryptography based certificate authorities do not address the boundary crossing trust needed in open distributed systems.

1.4.7 Domain keys

Domain keys is a domain based sender authentication approach for email sender authentication [7, 27, 95]. It originates from industry attempts at solving the email spam problem by authenticating sender addresses. Domain Keys Internet Mail (DKIM) [4] is an initial proposal in this area. An enhancement to the domain keys approach is proposed by Accredited Domain Keys Internet Mail, or ADKIM [36]. This enhancement improves the resistance of domain keys to DNS cache poisoning attacks. The domain keys approach works by associating domain public keys with the DNS records of domains. Corresponding private keys are used to digitally sign the headers of outgoing email messages. Receivers can confirm that the sender address is authentic because it is contained in the digitally signed mail header. This allows email receivers to authenticate sender email addresses.

The idea behind the domain keys approach is to make domain administrators accountable for all the email sent from their domain. In a typical organizational setting, the mail transport configuration is handled together with domain administration. Thus, the domain keys approach represents an appropriate level of granularity for spam control, especially in organizational settings. Domain based authentication methods represent an intermediate approach where domains are authenticated instead of end-points.

Sender policy framework (or SPF) is a proposed open standard for associating mail sending policy with domain records [100]. The motivation for this proposal comes from the observation

that sending email from an organizational email address has policy consequences for the organization. Unauthorized forwarding and transmission of emails from rogue servers is a threat. This threat is handled by the SPF proposal [100]. A similar approach for enforcing mail sending policy is the Sender Id proposal [1]. This proposal is similar to SPF except for what fields it validates in the sending policy. Validating email sending policy at the domain level is useful because they help in detecting forged sender addresses. For example, a domain `xyz.com` could nominate a particular server to send all the emails for senders in the domain. The receiving mail transfer agent would check if this policy is being respected, and refuse to accept emails coming from senders in another domain.

These proposed solutions are at the domain level, and are complementary to our end-to-end solutions. Our solution aims to achieve individual key authentication, which is at a finer granularity. Using the end-to-end argument [87], only the application that uses sender authentication is best equipped to correctly implement it. For example, users may want to distinguish senders on the same domain and be willing to receive email from `friend@abc.com` but not from `stranger@abc.com`. This kind of fine grained control is desirable in large open systems where it is necessary to distinguish multiple senders on the same domain.

1.4.8 Web of trust

The web of trust approach relies on natural human trust for public key authentication. It was introduced by the Pretty-good-privacy (PGP) email security system [32, 109]. PGP allows peers to authenticate public keys in an email environment by issuing digitally signed public key certificates to each other. The digital certificate represents an endorsement of the public key, thereby confirming the association between the key and its owner, and authenticating the public key. Authenticated sender public keys allow the receiver to validate sender identity and content integrity by verifying digital signatures on the incoming email messages. PGP can also provide confidentiality to email messages. The sender encrypts email with the receivers' public key. The encrypted message can be decrypted only by the intended receiver, thereby providing an assurance for confidentiality. The ability to provide public key authentication without additional infrastructure is a salient feature of PGP's web of trust. The need for human input for judging key authenticity is a limitation.

The web of trust works by making manual decisions on key ownership and trustworthiness. This extends the human concept of trust to digital identities. Researchers claim that PGP's limited usage can be attributed to its user sophistication needs [99]. Infrastructure free peer to peer public key authentication is achieved in PGP by relying on the implicit secure side channel of human trust decisions. Its dependence on human evaluation of key ownership and peer trustworthiness prevents its large scale usage in systems where human users are unable or unwilling to make these trust decisions.

1.4.9 Key continuity management

Key continuity management (KCM) is an approach for authenticating public keys based on human trust judgment [33, 38]. KCM takes the pragmatic approach of tolerating a window of vulnerability during the first interaction between two users. Making this compromise secures the ongoing interactions of unsophisticated users without depending on security infrastructure. Users generate and certify their own public keys, which are exchanged during the first interaction. This initial interaction is vulnerable to impersonation attacks. KCM addresses this vulnerability by assuming that human users are smart enough to tell if the other party is genuine or not, and that the chance of a man-in-the-middle attack is negligible during the initial interaction. The public key is considered to be authentic upon completion of the initial interaction. This public key is used to secure subsequent interactions.

The key continuity approach is highly usable because it does not need security infrastructure or user sophistication. It is used for public key authentication in typical SSH installations [104]. Human users are asked to approve newly encountered public keys, while the previously authenticated public keys are used transparently. The usability of KCM for email environments has been investigated in [33]. The study replicates a previous study on the usability of PGP [99]. It finds that key continuity is useful for securing emails even with an unsophisticated user base.

1.5 Authentication in ad-hoc networks

Ad-hoc networks are distinguished by their lack of global addressing infrastructure. They are desirable in transient networking applications like vehicular networking. Ad-hoc networks are also important for disaster recovery and military applications. The transient nature of connectivity between participants in these networks precludes the use of fixed network addresses. Transient ad-hoc addresses and the lack of a fixed routing infrastructure make it a challenge to authenticate participants in ad-hoc networks.

Participants in ad-hoc networks are usually referred to as nodes. Nodes detect their one-hop neighbors through a neighbor discovery mechanism. These one-hop neighbors are used by ad-hoc routing protocols for multi-hop routing through the ad-hoc network . The ad-hoc routing protocols can be categorized into a a few well established approaches. These approaches differ not only in the routing approach, but also in their concept of routing destination. This happens because ad-hoc networks lacks a global addressing infrastructure. Different concepts of destination lead to different models of authentication in ad-hoc networks. These different authentication approaches are outlined in this section.

1.5.1 Authentication for distance vector routing

Ad-hoc routing can be implemented by proactively setting up routing tables at the participating nodes. One approach for ad-hoc routing is to maintain routing tables of inter-nodal *distance vectors* at every node. This distance vector approach is used by the Destination sequenced distance vector routing (DSDV) protocol for ad-hoc routing [79]. DSDV operates by proactively propagating routing tables in the ad-hoc network either as complete routing table dumps or as incremental updates. The routing protocol is robust to connectivity changes and behaves well in mobile settings. Since the state required for storing and sharing routing tables is proportional to the number of nodes in the network, it is feasible only for small networks. This limitation has prevented DSDV from being deployed in real life commercial applications. Another characteristic of proactive routing is that some amount of network traffic is required for maintaining the routing tables. Avoiding the ongoing idle power consumption becomes important in low power networks like sensor networks. Although distance routing is not applicable to large networks, its robustness to network topology changes makes it a common choice for experiments and research in ad-hoc networks.

Implementing ad-hoc routing by maintaining routing tables implies that nodes have globally unique names or identifiers. Global names require a central naming authority that can assign unique names to nodes. This pre-existing centralization in the ad-hoc networking model makes it easy to justify the use of some centralized assumptions. The Secure Efficient Ad-hoc Distance vector routing (SEAD) protocol secures distance vector routing against impersonation and routing attacks as long as an initial hash value is authenticated per node [42]. The idea behind the protocol is to use one-way hash chains to ensure that misbehaving nodes cannot incorrectly modify the shared routing tables. Using cryptographic technique to enforce route consistency makes the protocol robust against a number of attacks. The underlying limitations of distance vector routing including limited network size are also inherited by SEAD.

Another class of attacks consists of malicious or selfish nodes reporting incorrect route weights, thereby making it hard or impossible to reach the correct nodes in through efficient and feasible routing paths. These path length based attacks have also been investigated. S-DSDV is a secure routing protocol based on DSDV that can detect fraudulent routing table updates caused by malicious nodes provided multiple nodes are not in collusion[98]. This non-collusion requirement is met by making the node layout assumption that malicious nodes are never one-hop neighbors. The S-DSDV approach uses a public key infrastructure for bootstrapping trust. It therefore has limited applicability because all the nodes must belong to the same logical trust domain. The node layout constraint also makes it unsuitable for general deployment.

1.5.2 Authentication for source routing

Dynamic source routing (DSR) is an ad-hoc routing technique where the source node provides an explicit routing path to the message [45, 46]. This routing approach requires source nodes to discover a route to the destination and then use the discovered route for source routing messages. Thus, the routing protocol has two stages: route discovery and source routing the message on discovered route. Route discovery is usually done by flooding the network for the desired node. Since the route discovery is done only when a message needs to be routed, the source routing approach is a reactive one. Unlike the proactive routing approaches, there is no ongoing network traffic for maintaining routing tables. This difference makes source routing attractive to power limited devices. It is also attractive to use source routing for sensor networks with infrequent inter nodal communication pattern. The weakness of source routing is that messages need to carry the entire routing path in their headers. This overhead increases linearly with route length, thereby limiting the route length that can be supported efficiently in source routing. DSR is a popular ad-hoc routing protocol, which has been used in a number of research and industry implementations [26, 86].

Authentication in dynamic source routing has been investigated by prior research. Hu and Perrig propose the Ariadne protocol for securing on-demand and source routing protocols in ad-hoc networks [43]. The protocol requires a secure cryptographic initialization phase consisting of either a pair wise secret key setup between all possible communicating pair of nodes, or setting up digital certificates signed by a certificate authority for all the nodes. This initialization step is not fully compatible with the ad-hoc usage scenario where nodes may need to be added or removed from the network at runtime. However, the protocol makes the important contribution of permitting nodes to authenticate their transmissions, and allowing the ad-hoc traffic to be routed on safe paths consisting of non malicious nodes. The use of efficient symmetric key cryptography makes Ariadne suitable for resource constrained ad-hoc networking environments.

An alternative approach for achieving authentication through self organized trust relationships is proposed by Hubaux et. al. in [44]. This self organized public key infrastructure is modeled after the web of trust of PGP. The trust management protocol allows mobile nodes to share public key certificates and thereby build a certification path to authenticate nodes. This trust generation procedure has been validated in the Secure dynamic source routing (SDSR) protocol [49]. While the use of self organized public key infrastructure for achieving authentication in ad-hoc networks appears to be promising, it depends on a critical simplification of the trust problem in ad-hoc networks. The assumption that the trust relationships present among PGP email users can be transferred to co-operating nodes in a mobile network appears questionable in settings where there is no one-to-one relationship between network nodes and humans. This authentication approach is therefore constrained to operate in environments where individual human owners own the nodes and take the time to issue public key certificates to other

nodes. The other issue with this approach is that long lived private keys are assumed to reside on the co-operating nodes. Considering the security of nodes in a mobile environment, this approach may not achieve the desired level of robustness in adversarial operating environments.

1.5.3 Authentication for geographic routing

Geographic routing is an established protocol for routing in ad-hoc networks [14, 30, 50]. It requires nodes to know their geographic locations. This requirement is increasingly simple to satisfy. A large number of personal mobile devices like PDA or cell phones have location information provided through geographic positioning system (GPS). Geographic routing relies on nodes knowing their geographic locations, and using their one-hop neighbors for routing packets to target geographic destinations. Location awareness simplifies ad-hoc routing making the local routing state at a node independent of network size or route length. However, using location as a routing method also also raises privacy concerns. Using geographic routing implies that node locations are well know. This allows adversarial nodes to track node locations and thereby violate location privacy.

The privacy threat posed by location aware devices has also been a topic of intense research. Defective or compromised devices could allow tracking of their users. A coalition of malicious nodes could co-operate to continuously track the geographic locations of correct devices (and their users). Existing research has taken two approaches for protecting user privacy: the first is to fudge the locations of identifiable nodes as in [37, 66]. Reducing the available accuracy of location is a promising approach for protecting node privacy. However, it still leaves the opportunity for collaborating malicious nodes to use the approximate location as a starting point for an exhaustive localization attack. The alternative approach is to use transient pseudonyms for temporary identification of nodes as proposed in [11, 18, 81, 101, 108]. Temporary pseudonyms help in protecting node privacy by preventing adversaries from linking a sequence of pseudonyms to a single physical node.

The type of authentication possible in networks using geographic routing depends on the node anonymity. Networks having permanent node names can authenticate messages by attributing them to source node names. An example of this is the voting based location authentication scheme is presented in [65]. This secure localization work assumes secure initialization and permanent node names. Thus, it assumes identity authentication and derives a voting based location authentication on top of it. The reliance on ling lived node names makes the localization protocol susceptible to location privacy violations.

Location authentication can also be implemented without depending on long lived node names. An out of the band method for location authentication has been investigated in [91]. Using speed of sound and speed of light as physical constants, the cryptographic protocol authenticates node location physically. Another approach to location authentication by improving

the robustness of location measurement is proposed in [64]. Here, the authors provide statistical tools and a mechanism for using measurements from multiple nodes to minimize the impact of intentional measurement errors on localization. The out of band location authentication is made robust by taking multiple readings. Using a secure out the band communication channel simplifies the problem at hand by having stronger assumptions. However, it may also introduce a new type of vulnerability, for example, if the adversary uses microphones in given locations.

1.6 Contributions

The main contributions of this dissertation are as follows:

- **Byzantine fault tolerant public key authentication in peer-to-peer systems**
  We design a byzantine fault tolerant public key authentication protocol (BPKA) for peer-to-peer systems. Authentication is done without trusted third parties and tolerates byzantine faults under an honest majority assumption. An anti-entropy version of the authentication protocol is developed in order to reduce the messaging cost. The cost implications of the authentication mechanism are studied by simulation. This work has been published in the *Journal of Computer Networks, Special issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security, 2006* [75].

- **Social-group key authentication for email**
  We design social-group key authentication (SGKA) for email in order to provide automatic public key authentication to unsophisticated email users. The solution is automatic, byzantine fault tolerant, eventually correct, incrementally deployable, backward compatible with the existing email infrastructure, and does not use trusted third parties. We investigate the performance of the proposed email authentication method through micro-benchmarks, simulation on an industrial and an academic email trace, and live experimentation on an instrumented mail authentication prototype. This work has been published in the *Proceedings of the Fifth Conference on Email and Anti-Spam, 2008* [74, 76].

- **Location authentication for ad-hoc networks**
  We design geographical secure path routing (GSPR), an infrastructure-free secure geographic routing protocol, which does not require out-of-band communication or shared secret initialization. The protocol authenticates the geographic locations of anonymous nodes in order to provide location authentication and anonymity simultaneously. The protocol does not require secure initialization and works correctly if there is a sufficient density of honest nodes. We also develop a novel method called geographic hash for encoding unforgeable geographic location data. This work has been published in the *Proceedings of the International Conference on Vehicular Electronics and Safety, 2008* [77, 78].

1.7 Outline of the dissertation

The outline of this dissertation is as follows. Chapter 2 addresses the problem of public key authentication in addressable networks. It presents a byzantine fault tolerant public key authentication protocol (BPKA), which is infrastructure free, and does not require human trust inputs for operation. The correctness and performance of the protocol are discussed, and an optimized version of the protocol is shown to achieve authentication with a low messaging overhead. Chapter 3 presents the social-group key authentication protocol (SGKA) for automatically authenticating public keys of email users. This chapter discusses the design and implementation of the protocol. Authentication performance is investigated through micro-benchmarks on an instrumented email authentication prototype, and through simulation on two real life email traces. Chapter 4 presents the geographical secure path routing protocol (GSPR) for authenticating geographic locations in ad-hoc networks. This chapter discusses the design and security of the protocol. A simulation based performance investigation of the protocol is also presented. Chapter 5 concludes the dissertation with a discussion of the contributions and a vision for the future.

Chapter 2

Byzantine Fault Tolerant Public Key Authentication

2.1 Problem statement

Public key authentication is a fundamental problem of digital security. Authenticated public keys are required for boot-strapping shared secret encryption methods and for verifying the integrity of digital signatures [23, 80]. Trusted third parties and webs of trust are the two established methods of public key authentication.

The trusted third party model uses a centralized offline public key certifying authority that is trusted by all the participants. This authentication architecture is hierarchically extended to create Public key infrastructure [19]. While trusted third parties are acceptable in a client-server computing model, they are not suitable to the users of peer-to-peer systems for a number of reasons. It may be impossible to find sufficiently trusted parties in heterogeneous systems. Trusted third parties must support certificate revocation to prevent misuse of compromised private keys [6, 31, 70]. The off-line advantage of certificate authorities is reduced by the overhead of maintaining fresh revocation information. Considering the consistency and timely propagation issues imposed by the mechanism, and the administrative burden placed by the security policy, it may not be possible to scale up the centralized authentication mechanism for securing large systems like the Internet.

PGP creates a web of trust, which allows peers to authenticate public keys by making manual decisions on key ownership and trustworthiness [32, 109]. Although PGP supports peer-to-peer key authentication, its dependence on human evaluation of key ownership and peer trustworthiness prevents its large scale usage in autonomous peer-to-peer systems. It also appears that its manual usage is limited to sophisticated users [99].

2.1.1 Our solution

The increasing usage of large autonomous peer-to-peer systems motivates the creation of our Byzantine fault tolerant public key authentication protocol (BPKA). The proposed mechanism involves a distributed system of mutually authenticating semi-trusted parties and tolerates byzantine faults. Authentication is eventually correct if no more than $\lfloor \frac{n-1}{3} \rfloor$ of the n parties are malicious or faulty. Authentication does not require predefined trusted third parties and

enables secure communication in heterogeneous groups. Since it allows the autonomous lightweight mutual authentication of strangers, BPKA is well suited for the authentication needs of peer-to-peer systems.

2.2 State of the art

Alternative approaches to provide fault tolerant authentication are known. There is a class of protocols relying on threshold cryptography that uses key shares with the following property: without a quorum of participants, it is impossible to create a digitally signed public key certificate [35]. As a consequence, unless the number of malicious parties is as large as the quorum, false authentication is impossible. Threshold cryptography requires the existence of a trusted dealer that initializes the key shares. In this way, it depends on the honesty of the dealer. Threshold cryptography has been used in COCA, a fault tolerant public key authentication service [107] and as the basis of a number of other secure services [15, 84, 106].

Threshold cryptography is also used to implement proactive recovery. To compromise such a system, the adversary is required to compromise the quorum within its vulnerability window or lose any previous progress due to a re-randomization of key shares [16]. Although better than static key shares, the scheme cannot recover from the compromise of a quorum because the same long term shared secret is recycled among the trusted parties. In contrast, our distributed BPKA authentication is proactively secure in the sense that it holds no long term secrets.

Role based access control in a distributed system has been studied earlier [10]. It uses roles instead of identities for granting access and therefore, avoids the issue of identity authentication [90]. Reputation based distributed trust has been investigated by a number of previous projects. The Free Haven project uses a proactive mechanism based on recommendations to protect anonymity of the users [24]. NICE allows the creation of trustworthy peer groups through a trust evaluation mechanism based on reputation [58]. Both systems aggressively eliminate (overtly) malicious parties to preserve their correctness. Our byzantine fault tolerant authentication builds upon this idea of peer reputation. It does not require external identity authentication and works on provable observations instead of recommendations.

PGP has applied decentralized trust to authentication in distributed systems [109, 32]. However, it requires human evaluation of trustworthiness, which limits its applicability for unsophisticated users and autonomous systems [99]. Trusted ambient communities [96] is an approach that incrementally builds trusted groups by observing the behavior of securely initialized peers. It is more permissive of malicious peers than our authentication mechanism, which needs verifiable challenge response proofs.

Cryptographic identifiers provide authenticated identities by selecting network identity related to the public key [69]. This approach is simple but constrained by the need to acquire specific network identifiers. Additionally, this authentication mechanism does not handle the

man-in-the-middle attack. A randomized approach to setting up weakly secure peer-to-peer networks is used in Smart Dust [8]. Although it does not focus on authentication, it shares distributed trust and a weakened adversary model with our byzantine fault tolerant authentication.

2.3 Model

Public key authentication by trusted third parties is traditionally done through certificate authorities that digitally sign a public key certificate. Verification of the certificate makes a statement of the following form: If the certificate authority is honest and capable, then the private key corresponding to the certified public key belongs to the given identity. Here the concept of identity is very general. It may span applications, individuals, and organizations. In contrast, our mechanism authenticates network end-points or peers. Authentication is a proof of possession of the private key under honest majority and other assumptions as described below.

2.3.1 Network

Consider a distributed system of mutually semi-trusting peers. They are interconnected by an asynchronous network and are identified by their network identifiers. The network does not guarantee message ordering or delivery. However, no part of the network becomes permanently disconnected. The network is also assumed to return delivery failure notifications. In particular, a notification is expected if a message is sent to a non-existent end point. Similarly, if an end-point does not implement the authentication mechanism, this fact can be detected by receiving a connection failure message.

We assume that disjoint message transmission paths exist to each peer from some of its peers. The parameter ϕ is defined to represent the proportion of non-disjoint paths among all the message transmission paths. Disjoint message transmission paths are not expected to be vulnerable to the man-in-the-middle attack. Further, if the man-in-the-middle attack can be mounted for more than a fraction ϕ of its peers, then the peer is considered to be faulty. Faulty end points, as shown in Figure 2.1, are not authenticated by our protocol. The authentication mechanism is designed to detect and ignore faulty peers.

2.3.2 Honest majority

Correctness of authentication depends on the existence of honest peers that faithfully execute the protocol. Informally, they tell the truth about their network identity and public key. While none of the peers inherently trusts any other peer, each believes that the honest peers are in a majority. We define honest peer and honest majority as follows:

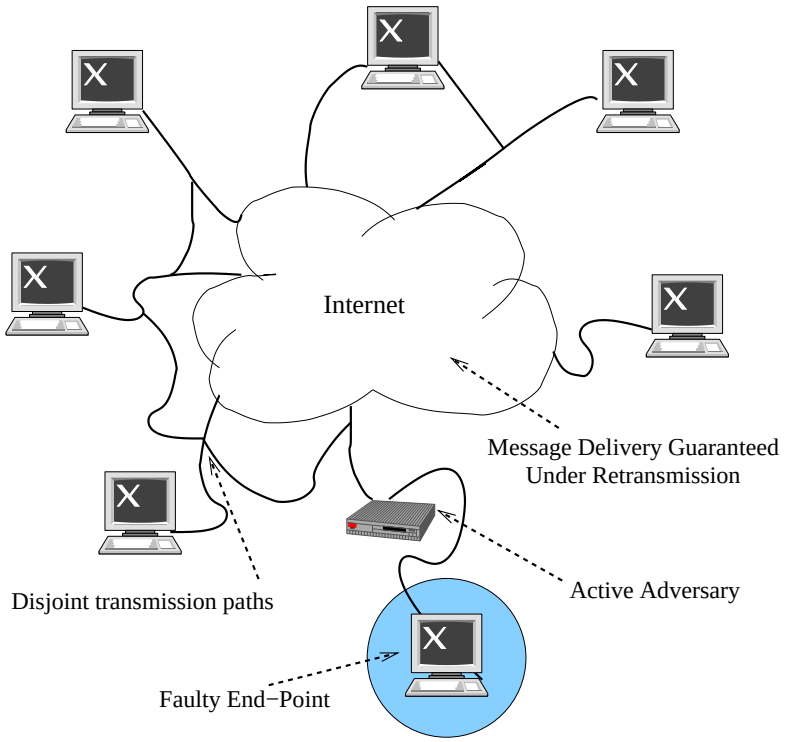

Figure 2.1: Adversaries and assumptions.

DEFINITION 1 *An* **honest peer** *protects the privacy of its private key and executes the authentication protocol correctly. A set of n peers has* **honest majority** *if the number of malicious or faulty peers* $t < \frac{1-6\phi}{3}n$.

Dishonest peers may behave in an arbitrary fashion, either because of being faulty or because of being malicious adversaries. It is assumed that the system of mutually authenticating peers has honest majority.

2.3.3 Adversaries

The computational power of adversaries is polynomially bounded. Hence, with a very high probability, adversaries cannot forge digital signatures or invert encryption transformations. We consider both active and passive adversaries. Passive adversaries have the power to eavesdrop on any message. While active adversaries have the power to inject arbitrary messages into the network[1], they cannot prevent message delivery for more that a small fraction ϕ of the honest parties. This adversary model is weaker than the classical one because the network adversary

[1]Since we do not address denial of service type of attacks, the spoofing power is not large enough to break the network or the parties processing the forged messages.

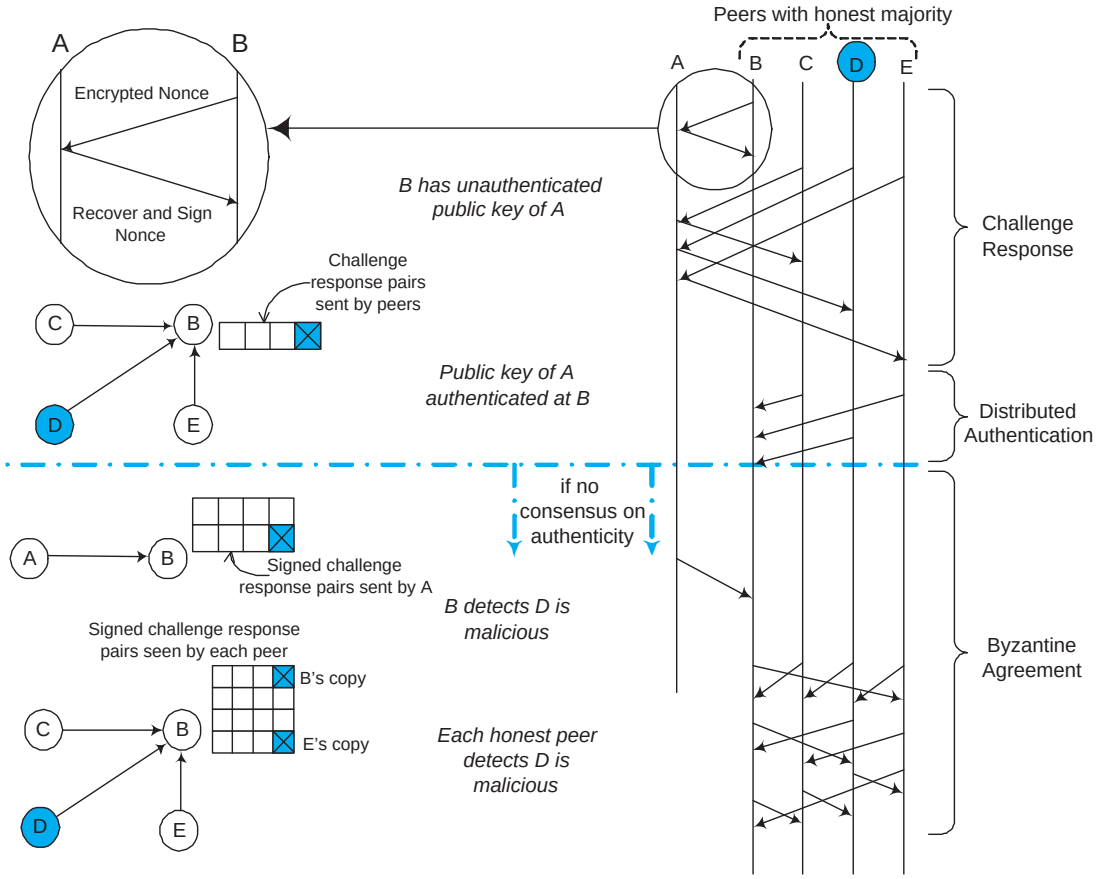

Figure 2.2: Authentication protocol example: A peer *A* is authenticated by *B* using its trusted peers. *D* is a malicious peer that tries to prevent authentication of *A*.

can not delete arbitrary messages.

The weakened active adversary is appropriate in wireless networks because of physical difficulties in silencing radio transmissions. Its use in Internet applications is justified by considering the difficulty of preventing message delivery to a large number of end-points. Practical experience with Internet based systems also suggests that message injection or spoofing is the preferred form of attack.

2.3.4 Authentication

We propose to authenticate public keys of peers through our authentication protocol. Public key authentication is defined as follows:

DEFINITION 2 Public key authentication *denotes the association of a public key to a peer. It also provides an assurance that the corresponding private key is known to the peer.*

Challenge response protocols can authenticate public keys in the absence of a man-in-the-middle attack. Given n message transmission paths to a peer, we allow for a limited number $n\phi$ of such attacks. Therefore, public keys of peers can be authenticated by multiple challenge response exchanges originating from different end-points.

The authentication protocol (Figure 2.2) consists of three main operations: CHALLENGE RESPONSE, DISTRIBUTED AUTHENTICATION, and BYZANTINE AGREEMENT. During challenge response, the peer to be authenticated is challenged with encrypted nonces by a set of peers. Since the nonce can be recovered only by the possessor of the private key, a correct response is a proof of possession.

In the distributed authentication phase, peers forward their proofs to other peers. A peer *B* can authenticate a peer *A* after it receives a number of valid proofs from different peers. If all the participants are honest, there will be consensus on validity. In this common operating case, the protocol terminates with *B* becoming convinced that the public key is authentic.

If there are conflicting claims on authenticity, *B* can deduce that either *A* or some of the peers are malicious or faulty. The protocol proceeds to Byzantine agreement where the sent and received messages of different parties are validated. As all the messages are digitally signed, malicious behavior can be discovered by this procedure.

The messaging cost of authentication motivates optimization of the common case when all trusted parties are indeed honest. The public key infection protocol implements *optimistic authentication*, which hides latency by proceeding before a public key is authenticated. Public keys and authentication proofs are propagated efficiently by an anti-entropy public key infection algorithm described in Section 2.6.

2.3.5 Trusted groups

Each peer has a probationary group, trusted group, and untrusted group of peers as shown in Figure 2.3. Peers gain knowledge of each other's public keys depending on their communication patterns. Newly discovered peers are added to the probationary group. Successful authentication moves a peer from the probationary group to the trusted group. Malicious peers are moved from the trusted group to the untrusted group. [2] Peers are also deleted from trusted groups for lack of liveness and for periodic pruning of trusted group. This is done in order to improve authentication performance.

[2] Continuous addition of malicious peers can cause the untrusted group to grow without limit. Therefore, peers may forget malicious behavior of the very distant past.

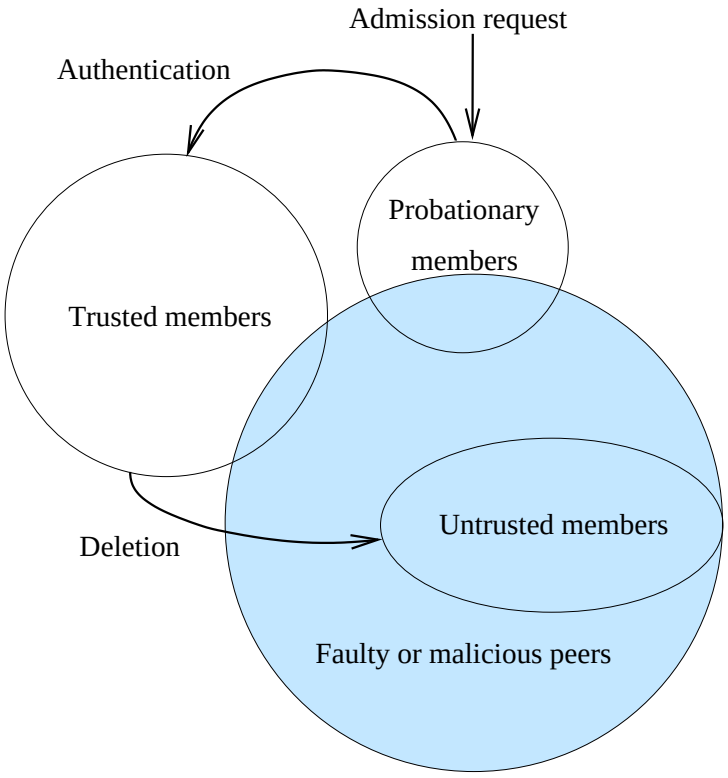

Figure 2.3: Trust groups for executing the authentication protocol at a peer.

2.4 Architecture

Byzantine fault tolerant public key authentication (BPKA) is implemented by executing the Authentication protocol and the Membership control protocol at each peer. The protocols are described as per the notation given in Table 2.1. A bootstrapping procedure is also provided for system initialization. All protocol messages have timestamps, source and destination identifiers, and digital signatures. Peers ignore messages with invalid signatures and maintain a most recent received time-stamp vector to guard against replay.

K_i	Public key of the principal i.
K_i^{-1}	Private key of the principal i.
$K_i(x)$	A string x encrypted with the public key of i.
r_i	A pseudo random number generated by peer i.
$\{x,y,z\}$	A message containing three strings x, y and z.
$\{x\}_i$	A message signed by i.
$\mathcal{T}(i)$	Trusted group of peer i.

Table 2.1: Notation for BPKA protocol

1. ADMISSION REQUEST
   A peer A makes a key possession claim by notifying the peer B. If A has an expired authenticated public key $K_A^\star$, it includes the proof of its possession $\mathcal{P} = \{A, K_A\}_{A^\star}$. B announces the claim to the group.

   $A \rightarrow B \quad : \quad \{A, B, \texttt{admission request}, \{A, K_A[, \mathcal{P}]\}_A\}_A$
   For each trusted peer P_i of B
   $B \rightarrow P_i \quad : \quad \mathcal{B}[i]$
   where
   $\mathcal{B}[i] = \{B, P_i, \texttt{authentication request}, \{A, K_A[, \mathcal{P}]\}_A\}_B$

2. CHALLENGE RESPONSE
   Each peer challenges A with an encrypted nonce, and A responds with the signed response. A also stores the challenge response pair $\{\mathcal{C}_{iA}, \mathcal{R}_{iA}\}$ from its interaction with peer P_i as $\mathcal{V}_A[i]$ for use in Byzantine agreement.

   At each trusted peer P_i of B
   $P_i \rightarrow A \quad : \quad \mathcal{C}_{iA} = \{P_i, A, \texttt{challenge}, K_A(r_i)\}_{P_i}$
   $A \rightarrow P_i \quad : \quad \mathcal{R}_{iA} = \{A, P_i, \texttt{response}, r_i\}_A$

3. DISTRIBUTED AUTHENTICATION
   Each peer returns the proof-of-possession $\{\mathcal{C}_{iA}, \mathcal{R}_{iA}\}$ to B. B saves the pair in a local variable $\mathcal{V}_B[i]$ and determines the public key to be authentic (or inauthentic) if there is consensus on validity (or invalidity) in the proofs received. If there is no consensus, B calls for Byzantine agreement.

   At each trusted peer P_i of B
   $P_i \rightarrow B \quad : \quad \{\mathcal{C}_{iA}, \mathcal{R}_{iA}\}_{P_i}$

4. BYZANTINE AGREEMENT
   B asks the peer A for the challenges it received, and its responses to them. It then compares the proofs received from the peers and those received from A. It also notifies the peers of the received proofs so that malicious parties are eliminated from the trusted group.

   $B \rightarrow A \quad : \quad \{B, A, \texttt{proof request}\}_B$
   $A \rightarrow B \quad : \quad \{A, B, \texttt{proof}, \mathcal{V}_A\}_A$

   If A is not proved malicious
   For each trusted peer P_i of B
   $B \rightarrow P_i \quad : \quad \{B, P_i, \texttt{byzantine fault}, \mathcal{B}, \mathcal{V}_B\}_B$

   For each trusted peer P_j of P_i
   $P_i \rightarrow P_j \quad : \quad \{P_i, P_j, \texttt{byzantine agreement}, \mathcal{B}, \mathcal{V}_j\}_{P_i}$

Figure 2.4: Authentication protocol.

2.4.1 Authentication protocol

The authentication protocol (Figure 2.4) consists of the following steps:

- ADMISSION REQUEST
  The protocol begins when B encounters an unauthenticated public key K_A. It announces the key to its trusted group and asks them to verify its authenticity.

- CHALLENGE RESPONSE
  Each peer P_i challenges A by sending a random nonce encrypted with A's supposed public key in the signed challenge message. A can recover the nonce only if it holds the private key K_A^{-1}. It returns the nonce in a signed response message. The challenge response message pair is a proof of possession for the public key. At end of the challenge response phase, each peer gets a proof of possession for K_A [3]. Each challenger waits for an application specific time-out. It deletes the proof if duplicate responses are received.

- DISTRIBUTED AUTHENTICATION
  The peers respond to B's authentication request by sending their proofs of possession to B. If all peers are honest, then there will be consensus on the validity of proofs. In this case, B gets the authentication result and the protocol terminates.

- BYZANTINE AGREEMENT
  If there are differing authentication votes, then either A or some of the peers are malicious or faulty. To detect if A is malicious, B sends the proof request message to A. The response consists of all challenge messages received, and the responses sent by A. If A is honest, it can prove that it received the messages because they were signed by the sending peer. It can also show a correct response. If A is not provably malicious, then some of the peers must be malicious or faulty. This leads B to announce a byzantine fault to the group. Now, each group member will send the byzantine agreement message to others. At end of this phase, the honest peers will be able to recognize malicious peers causing the split in authentication votes.

2.4.2 Bootstrapping

The bootstrapping procedure is provided to cold-start the system. This is in contrast with the situation when trusted groups already exist and a peer joins some of them. Bootstrapping initializes the authentication system by creating a trusted group consisting of the bootstrapped peers. The peers authenticate each other by requesting admission into this trusted group. It should have honest majority to function correctly.

[3] Since K_A is not yet authenticated, digital signature is not verified on the response message.

1. PUSH PROOFS
   A peer D periodically pushes the proof-of-possession $\{C_{DA}, \mathcal{R}_{DA}\}$ to peers that have not yet received its proof.

   For each trusted peer P_j that has not been sent the proof
   $D \rightarrow P_j \quad : \quad \{C_{DA}, \mathcal{R}_{DA}\}_D$

2. PULL PROOFS
   A peer B has some, but not all proofs of authenticity. It can ask any peer P_j for the proof to arrive at the authenticity and trusted group membership decision for the probationary peer A.

   For each trusted peer P_j that has not sent a proof
   $B \rightarrow P_j \quad : \quad \{B, P_j, \texttt{authentication request}, (A, K_A)\}_B$
   $P_j \rightarrow B \quad : \quad \{C_{jA}, \mathcal{R}_{jA}\}_{P_j}$

Figure 2.5: Membership control protocol.

2.4.3 Membership control protocol

Membership control (Figure 2.5) serves three purposes. It preserves honest majority of trusted groups, maintains consistency of trusted group definition among sets of frequently communicating peers, and prevents excessive growth of trusted group size to limit the cost of authentication. The group operations of the protocol are described below:

Addition to trusted groups

Each peer maintains a list of to-be-sent authentication proofs for each probationary peer. It lazily pushes these proofs to its trusted peers. Thus, the probationary peer becomes trusted at each trusted peer. A peer may pull proofs because lazy push may delay a required authentication. Peers pull the proofs by sending authentication request messages.

Deletion from trusted groups

Peers are deleted from trusted groups for malicious activity or lack of liveness. A malicious message causes execution of the Byzantine agreement phase, which captures malicious behavior except for the lack of liveness. Deletion occurs as a result of byzantine agreement [57] on the maliciousness of the proof. If the group has honest majority, all the honest trusted peers delete the malicious peer from their trusted groups and add it to the untrusted group[4].

[4]It is possible that honest peers may be deleted from dishonest groups. This causes them to join other groups, most of which have honest majority.

A peer that fails to respond to messages in a timely manner is considered failed due to lack of liveness and is deleted from the trusted group. Performance of authentication is maintained by preserving a suitable group size. Thus, honest peers voluntarily delete themselves from trusted groups by randomly selecting a trusted peer and ceasing to respond to its messages. The probability of deletion is chosen as a function of trusted group size in order to create suitably sized trusted groups.

Group migration

Authentication depends on honest majority of trusted group. However, trusted group membership is granted on key authentication and does not guarantee that the authenticated peer will not act maliciously in future. In particular, it is possible that a number of covertly malicious peers join a trusted group so that they are in majority. Trusted groups are periodically flushed to provide proactive security against unobservable loss of honest majority. This process also guards against the Sybil attack [25] that relies on a malicious peer creating multiple fake identities. The progress made by multiple fake identities is lost by group migration.

2.4.4 Complexity

Authentication of a public key requires the execution of the BPKA protocol. Let n be the number of peers participating in the protocol, and let k be the public key length. The number of messages, the message size, and the computational cost of the BPKA protocol are summarized in Table 2.2. The table is constructed by considering the cost on a peer B for authenticating

Protocol operation	*Number of messages*	*Message size*	*Computation*
CHALLENGE RESPONSE	$O(1)$	$O(k)$	$O(\text{poly}(k))$
DISTRIBUTED AUTHENTICATION	$O(n)$	$O(k)$	$O(\text{poly}(k))$
BYZANTINE AGREEMENT	$O(n)$	$O(n)$	$O(n^2\text{poly}(k))$
Worst Case	$O(n)$	$O(n)$	$O(n^2\text{poly}(k))$
Best Case	$O(n)$	$O(k)$	$O(\text{poly}(k))$

Table 2.2: Complexity of public key authentication through BPKA.

the public key of peer A. The messages sent and received by B are counted from the protocol definition (given in Figure 2.4). The computational cost is calculated by using $O(\text{poly}(k))$ as the complexity of challenge response and digital signature operations. The table also summarizes the best and worst case results. Since the presence of malicious or faulty peers triggers the expensive BYZANTINE AGREEMENT operation, the best case complexity is achieved if the protocol operates in a benign environment.

Table 2.3: Effect of attacks during challenge response. A is authenticated by B and its peers P_i.

Sender under attack	**A**	**B**	**P_i**
Spoofing	delay	K-invalid	K-invalid
Man in the middle	faulty	faulty	faulty
Incorrect response	P-invalid, A-invalid or K-invalid	K-invalid or delay	A-invalid or delay
No response	delay	delay	delay

2.5 Analysis

This section analyzes the correctness of authentication in honest majority groups. It also shows that the group dynamics resulting from membership control creates honest majority groups with high probability. Previous studies [94] have focused on proving boolean correctness assertions on two and three party authentication protocols. The authentication protocol described here is open ended in number of peers and incremental in its approach. Therefore, a direct case by case analysis of the protocol is developed below.

2.5.1 Challenge response

Consider a peer B with a trusted group of peers $\{P_1,\ldots,P_i,\ldots,P_n\}$. Let A request admission into the trusted group of B. Each peer P_i sends a proof of possession $\{\mathcal{C}_{iA},\mathcal{R}_{iA}\}_{P_i}$ to B, where

$$\mathcal{C}_{iA} = \{P_i, A, \texttt{challenge}, c_i\}_{P_i}$$

$$\mathcal{R}_{iA} = \{A, P_i, \texttt{response}, r_i\}_{A}$$

Let the proof of possession be valid if $c_i = K_A(r_i)$ and both $\mathcal{C}_{iA}$ and $\mathcal{R}_{iA}$ are properly signed.

CLAIM 1 *If P_i and A are honest, the proof of possession valid, and the communication path P_iA does not lose messages, then K_A is authentic with very high probability.*

PROOF: By contradiction, let K_A be inauthentic. Since P_i is honest, it transmits a correct challenge containing $c_i = K_A(r_i)$ to A, and does not disclose its nonce r_i.

Since the network path does not lose messages, the challenge will be delivered to A and the response delivered to P_i. Thus, if a single response is received, A must be the responder[5]. Since it computes $r_i = K_A^{-1}(c_i)$, it knows the private key, a contradiction. □

[5]If multiple responses are received, they are marked invalid by the protocol.

Attacks on challenge response

The challenge response protocol can be attacked in a number of ways. Messages may be spoofed and originating from sources other than their apparent origin X. Man in the middle attacks may cause a peer X' to impersonate X and protocol attacks could be launched by a peer X not following the prescribed protocol.

Let a proof of possession be P-invalid if the challenge is not properly signed, A-invalid if the response is not property signed, K-invalid if $c_i \neq K_A(r_i)$, and faulty if it is valid but K_A is not owned by A. Messages exchanged between the trusted peers are safe from spoofing and man in the middle attacks since they are signed by authenticated public keys. Considering the various possibilities of attacks on the protocol, the effect on correctness of challenge response is analyzed below. A summary is provided in Table 2.3.

We consider Spoofing, Impersonation and Protocol attacks on the authentication architecture. Spoofing is defined as the attack where an adversary A' assumes the identity of a peer A. This attack is detected by the challenge response mechanism. Impersonation is a man in the middle type of attack where an adversary M impersonates A while communicating with B, and B while communicating with A. In accordance with the mechanics of the attack, A and B cannot communicate directly without passing through M. We define protocol attacks as the set of attacks that are mounted by providing incorrect responses (or lack or responses) to various protocol messages. A number of other protocol attacks like replay, type flaws and encapsulation are rendered ineffective by the use of timestamps, message identifiers, and digital signatures respectively. In general, source and destination identifiers are part of message definition when the identity of communicating parties matters.

The adversary mounts a successful attack if at least one of its following goals are satisfied:

G1 Violate authentication
The adversary convinces an honest peer that the public key of A is $K_{A'}$ when it is not.

G2 Violate honest majority
The adversary creates an adverse selection of group members that lack honest majority

Consider the case of malicious peers that are not trusted by honest parties. They can attack challenge response in one of the following ways:

- **Spoofing**
  A malicious peer A' may try to impersonate an honest peer A by sending the `admission request` message. If A is already part of the trusted group, then each trusted peer has its correct authenticated public key K_A. Since A' cannot produce the required proof $\mathcal{P} = \{A', K_{A'}\}_A$ without the knowledge of K_A^{-1}, each honest peer will ignore the invalid request.

Each peer will challenge A if it does not belong to the trusted group. The peer A responds to the challenges because it is honest. As it does not have $K_{A'}^{-1}$, the response will be invalid. If the adversary also sends a spoofed response, then the honest peer H will receive two distinct responses, one valid and one invalid. H considers the incorrect message and A' is not authenticated. Therefore, none of the goals are satisfied.

- **Impersonation**
  By the limited power of active adversary, some of the challenges must reach A. Thus, some peers will see valid proofs for $K_{A'}$ while the others will get invalid proofs. The authentication will fail because the majority of peers contact A. Thus, **G1** is not satisfied.

  Because each peer can prove that it issued the proper challenge, and received the corresponding response, none of them is proved malicious. Thus, **G2** is not satisfied.

- **Protocol attack**
  The malicious peer A' can only choose the responses for the messages it sends. This limits it to the `admission request` message and the response message. Because a receiving peer B shall verify the validity of the digital signature, the message format and the correct recipient, the only choice for the peer is to create a message of the following form:

  $$\{X, B, \texttt{admission request}, \{X, K_X[, \mathcal{P}]\}_X\}_X$$

  This is equivalent to spoofing if $X \neq A'$. If $X = A'$, then the malicious peer will receive challenge messages from the peers. If it responds with incorrect source or destination identifier, incorrect signature or incorrect format, then the message will be discarded as ill-formed. Thus, it can at best send a message of the following form to the peer P_i:

  $$\mathcal{R}_{iA} = \{A', P_i, \texttt{response}, r_i'\}_{A'}$$

  Since $r_i \neq r_i'$ causes invalidity, the peer P_i sends the received response to its peer B that does not find a consensus. Thus, **G1** is not satisfied.

  The protocol now goes into the Byzantine agreement phase. B asks A' to send its copies of the challenge response proofs. A cannot produce a valid challenge since it does not know K_{P_i} and cannot create a new challenge. Its only option is either to send nothing or to send the available copy. Since it cannot prove that it received a correct challenge and recovered its response, the trusted peer is not proved to be malicious. Thus, **G2** is not satisfied.

If the malicious peer already belongs to a trusted group, then the following attacks are possible:

- **Spoofing**

Spoofing by trusted peers is restricted to parties outside the trusted group. This is because trusted peers have an authenticated public key of the peer being spoofed, and can detect an incorrect signature on the spoofed message. A malicious trusted peer may send spoofed challenge messages to a probationary peer. These messages will not count in distributed authentication because of invalid signature on the challenge. Further, if the probationary peer sends this invalid proof to any trusted peer, the insider is provably malicious and will be deleted in byzantine agreement phase. Neither **G1** nor **G2** is satisfied.

- **Impersonation**
  Consider a trusted peer P_i being impersonated by P_i' due to a man in the middle attack. P_i belongs to the trusted group because of successful authentication. The adversary P_i' cannot convince the trusted group that it is P_i because it does not know $K_{P_i}^{-1}$. Since badly signed messages are ignored, P_i appears to be non-live to some of the peers and **G1** fails. Also P_i is not proved to be malicious and **G2** fails.

- **Protocol attack**
  A malicious insider can delay the entry of a peer A into the trusted group by not sending a correct `authentication request` message. This does not satisfy either of the goals because A can find other honest peers.

  The challenge response phase can be affected by the malicious insider in the following way: It can fail to send the challenge or send a number of challenges. However, A cannot be proved malicious by any such strategy because it can remember the challenges and produce them to other trusted peers. These challenges will be sent in the Byzantine fault phase when an honest peer observes lack of consensus on authenticity of K_A.

  A malicious insider can delay sending the proof of possession. However, it cannot construct a bad response signed by A. As the malicious peer may only delay the authentication of A, neither **G1** and **G2** are not satisfied.

Therefore, the challenge response protocol provides correct authentication and preserves honest majority of trusted groups.

2.5.2 Distributed authentication

Distributed authentication ends with a consensus of valid if every participant is honest and there are no attacks. Given the presence of attacks and malicious peers, distributed authentication of A by B is correct as follows.

CLAIM 2 *A peer A is not mis-authenticated if it is honest.*

PROOF: Mis-authentication requires consensus on faulty proofs. Because A is not malicious or faulty, consider two possibilities: B is malicious or A' spoofs messages.

If B is malicious, it can either inform its peers of an incorrect $K_{A'}$ or fail to send correct messages to its peers. If B sends incorrect key(s), the honest peers will receive `authentication request` messages and send challenges to A. Because A responds by decrypting according to its correct private key, the proofs will be K-invalid. If B does not send the correct messages, honest peers will send no challenges and some proofs will be missing.

If A' spoofs responses for A, then the honest peers will delete their proofs because A will respond too. Since there must be missing or K-invalid proofs, there can not be a consensus on faulty proofs. □

CLAIM 3 *Honest majority is preserved at every honest peer.*

PROOF: If B is honest, lack of consensus leads to byzantine agreement. It compares proofs sent by peers with the proofs sent by A. If A is provably malicious because of sending a K-invalid proof to some peer and valid to another, or because of sending P-invalid proofs on request of B, the protocol ends with A being marked malicious. No honest peer is deleted.

If A is not provably malicious, the protocol moves into the second phase. This implies either some trusted peers are malicious or there is a man in the middle attack.

Each peer sends the proofs it has (for authenticity of K_A) to its trusted peers. Malicious peers could send conflicting proofs of possession to their peers. These actions are detected by byzantine agreement as follows: Consider an honest peer P_j receiving the `byzantine agreement` message from other peers. Let t of the peers be malicious and may offer arbitrary proofs. Secondly, ϕn of the peers may not be able to reach A and may have faulty proofs to offer. Finally, proofs from another ϕn peers may be faulty because the peer P_j has a compromised path to them. In the worst case, these three sets of peers are disjoint, and $2\phi n + t$ of the proofs can be missing. Thus, every peer eventually gets at least $n - 2\phi n - t$ proofs. However, the malicious peers could respond eagerly, causing $2\phi n + t$ of the $n - 2\phi n - t$ received proofs to be faulty. Therefore, a majority of the proofs are identical and correct at every honest peer if

$$n - 4\phi n - 2t > 2\phi n + t$$

i.e.

$$t < \frac{1 - 6\phi}{3} n$$

Therefore, using a majority vote after Byzantine agreement allows the peers to form trusted groups that contain only the honest peers that are not in the path of a man in the middle attack. This preserves the honest majority.

If B is malicious and sends conflicting requests to the peers, its signed `authentication request` messages will cause it to be detected by Byzantine agreement on the requests received. Again, by deletion of the malicious peer B, honest majority is preserved. □

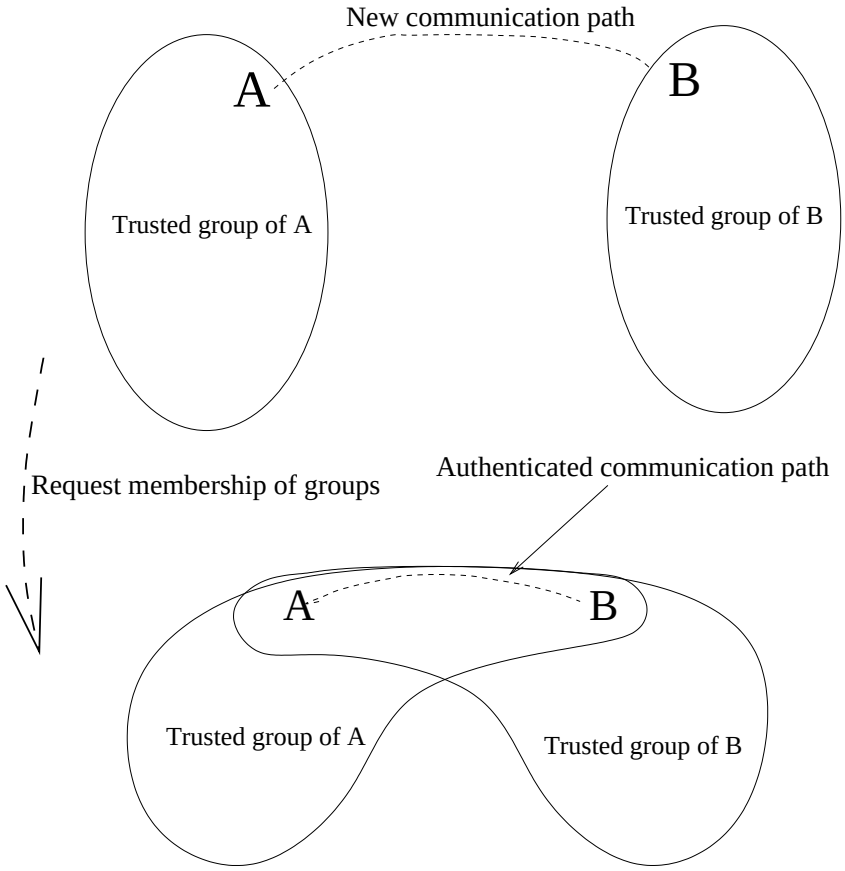

Figure 2.6: Dynamics of authenticated communication.

2.5.3 Group evolution

Admission requests are caused by the need for secure communication. If the peers A and B intend to communicate securely, they will check if $A \in \mathcal{T}(B)$ and $B \in \mathcal{T}(A)$. In this case, the problem is trivially solved.

Otherwise, A will request admission to $\mathcal{T}(B)$ and B will request admission to $\mathcal{T}(A)$. If both A and B are honest, the admission requests will succeed in the common operating case when their groups are also honest as shown in Figure 2.6. If one of the requests fails, then either Byzantine agreement will correct the groups as described earlier, or periodic pruning of trusted groups will ensure honest majority as described in the following section. In either case, the honest peers can eventually authenticate each other.

2.5.4 Formation of honest majority groups

Since honest members form trusted groups by following the membership control protocol, any provably malicious peers are deleted from trusted groups. On the other hand, if malicious peers

can successfully masquerade as honest peers, then the continuous group migrations cause the distribution of covertly malicious parties to be same as a random selection. Therefore, honest majority groups are formed with a probability greater than that of random selection.

A trusted group with $\frac{3t}{1-6\phi}+1$ peers has honest majority if t peers are malicious or faulty. Because the value of ϕ does not change the behavior of random selection, consider the honest majority group with $3t+1$ peers. Let the fraction of dishonest parties be $\frac{1}{3}-\varepsilon$, where $0<\varepsilon\leq\frac{1}{3}$. Under the assumption of independent random selection of members, the probability $P(i)$ of choosing $\mathcal{T}$ with i dishonest members is given by the following binomial probability:

$$P(i)=\binom{3t+1}{i}\left(\frac{1}{3}-\varepsilon\right)^{i}\left(\frac{2}{3}+\varepsilon\right)^{3t+1-i}$$

The probability $P_h=\sum_{i=0}^{t}P(i)$ of selecting an honest majority group is computed numerically and shows a rapid convergence to 1 as ε approaches $\frac{1}{3}$.

In practice, the untrusted sets preserve information about past malicious activities and impede the free assimilation of dishonest parties into trusted sets. Thus, two honest parties A and B can have an expectation $(1-P_h)^k<\frac{1}{2^k}$ of being incorrectly authenticated if they continue communication through k group migrations.

2.6 Public key infection

The protocols implementing distributed authentication are expensive in messaging cost. Trusted groups execute protocols that include broadcast messages leading to an $\mathbf{O}(n^2)$ cost for mutual authentication in trusted groups. In order to reduce the messaging cost, we use an epidemic algorithm called *Public key infection* for lazy propagation of protocol messages. As shown in Figure 2.7, the public key infection protocol stores the authentication protocol messages in a message cache, and forwards them to other peers through anti-entropy sessions. Because each protocol message is protected by an unforgeable digital signature, it is possible to store and forward messages through intermediate peers. This section discusses the protocol design, correctness, and the performance analysis in terms of messaging and space requirements.

Consider a lazy messaging layer underlying the authentication protocol discussed earlier. This layer maintains a cache of the undelivered messages and provides eventual consistency in the following sense: The outcome of the lazy protocol will approximate the outcome of the eager protocols described earlier. Thus, before the two protocols achieve the same assignment of public keys to peers, we shall be in the state of *optimistically trusting* the authenticity of public keys. If the optimistic trust is broken, we mark the offending peer untrusted as required by the authentication protocol.

Public key infection does not require knowledge of physical time but operates by maintaining a number of logical timestamps. A summary of the data structures required for public key

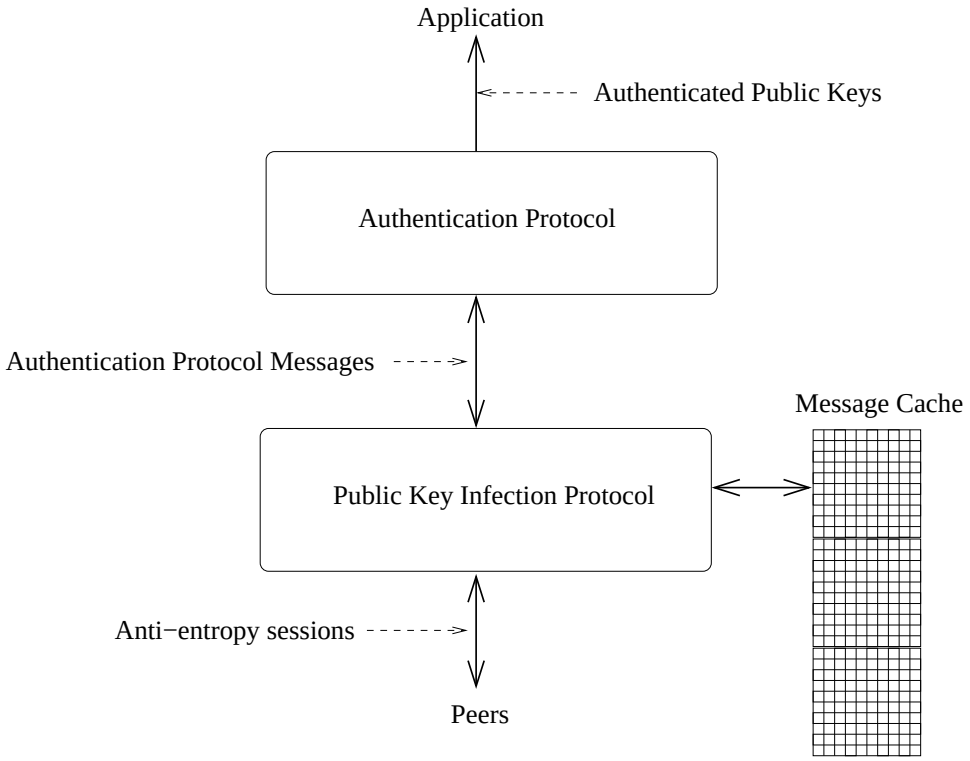

Figure 2.7: Overview of Public Key Infection.

infection is given in Table 2.4. Each peer maintains a numeric logical timestamp, `lts`, called the Lamport timestamp [56]. This timestamp is maintained by updating it to the maximum of the incoming message timestamp and the local timestamp. This timestamp provides a partial order on events in the distributed system. Each peer also maintains a causal timestamp, `cts`, which is simply a local event counter incremented on send and receive events. This timestamp is sent with each outgoing message, allowing peers to maintain a timestamp vector, `ctv`, of causal timestamps.

Key infection works by deferring the transmission of messages sent by the authentication protocol. The following steps are taken when a protocol message m with source s and destination d is pushed to the key infection layer: Firstly, the causal timestamp, $\texttt{ctv}[s]$, and the Lamport timestamp, `lts`, are incremented to record the change of state in the distributed system (Table 2.4). Secondly, the cache record $\{\texttt{l}, \texttt{ctv}[s], s, d, m\}$ is inserted into the message cache. The message cache data structure, as shown in Table 2.5, holds the outgoing messages and timestamps in order to achieve eventual delivery of authentication protocol messages.

`lts`	The Lamport timestamp.
`ctv[i]`	The last causal timestamp known for peer *i*.
`ltv[i]`	The Lamport time of the most recent message originating from peer *i*.
`srv[i]`	The stable read timestamp. Its value is the largest (known) Lamport time such that all messages with a smaller timestamp have been received at peer *i*.

Table 2.4: Data Structures required for Public Key Infection.

`lt`	The Lamport timestamp
`ct`	The causal timestamp at source
`src`	Source
`dest`	Destination
`mesg`	The cached authentication protocol message

Table 2.5: The cache record data structure for storing authentication protocol messages delivered through anti-entropy sessions.

Anti-entropy sessions

Anti-entropy sessions between peers transfer the protocol messages in an epidemic manner. The timing of anti-entropy sessions can either be application dependent in order to take advantage of piggybacking on application messages, or could depend on a timeout to prevent too much divergence from the eager protocol execution. The following steps define an anti-entropy session between the peers *s* and *d*.

- **Exchange time stamps**
  Lamport timestamps are exchanged and used to update the Lamport timestamp vector `ltv`. The local Lamport timestamp, `lts`, is also updated by the standard algorithm [56].

- **Send cached messages**
  The causal timestamp vectors are compared in order to decide which messages should be sent to the other peer. For any locally cached record *r*, if the following holds:

$$r.\mathtt{ct} > \mathtt{ctv}_d[r.\mathtt{src}]$$

It can be inferred that d has not seen the message stored in r. This is because the causal timestamp of the message, $r.\mathtt{ct}$, is later than the last causal timestamp, $\mathtt{ctv}_d[r.\mathtt{src}]$, that is known to d. Thus, all the records satisfying this *message exchange condition* have to be transmitted to the peer d in order to enable eventual delivery at the destination.

- **Receive cached messages**
  The peer d computes a set of records to be transmitted by the same logic. On receiving a record r, the receiver inserts r in the message cache, and updates the component $\mathtt{ltv}[r.\mathtt{src}]$ if $\mathtt{ltv}[r.\mathtt{src}] < r.\mathtt{lt}$. As a consequence, the Lamport timestamp vector contains the latest known Lamport time for each peer.

- **Exchange stable read time stamp**
  The stable read time stamp is computed as $\min_i(\mathtt{ltv}[i])$. The peer s assigns this value to $\mathtt{srv}[s]$. The updated stable read timestamp vector, $\mathtt{srv}$, is transmitted to d.

- **Delete received messages**
  The exchange of stable read timestamps allows the receiver to advance its components in the usual manner. The component $\mathtt{srv}[i]$ is advanced to the received value, $\mathtt{srv}_s[i]$, if $\mathtt{srv}[i] < \mathtt{srv}_s[i]$. Updates to the stable read timestamp also allow delivered messages to be deleted from the message cache. Consider a cached message r such that:

  $$r.\mathtt{lt} < \mathtt{srv}[r.\mathtt{dest}]$$

  Clearly, since the value $\mathtt{srv}[r.\mathtt{dest}]$ is computed at $r.\mathtt{dest}$ as minimum of the Lamport timestamps received from the peers, a message with greater or equal Lamport time must have been received from the message source as well. By assumption of ordered message delivery, the message r has already been received, and can be deleted. In this manner, the peer can delete all the records that satisfy this *deletion condition*.

Encrypted timestamps

The epidemic algorithm operates in a semi-trusted environment. Thus, it is necessary to ensure the integrity of timestamps. We create *encrypted timestamps* by requiring the peer i to generate the pair $\{t, K_i^{-1}(t)\}$ instead of the timestamp component t. Thus, the protocol processing outlined above would always pass timestamp values as pairs. Receivers would verify the correctness before acting on a timestamp value. By the assumption of non-invertibility, the secure timestamp can be generated only by the peer i. Thus, for an honest peer i, it is impossible to forge the timestamp component representing the state at i. Since the authentication protocol requires correct operation only from the honest peers, secure timestamps are sufficient to preserve the correctness of authentication protocol.

2.6.1 Complexity and coverage

The messaging cost and the rate of progress of public key infection are determined by anti-entropy sessions. Let the peers do an anti-entropy session with a randomly chosen peer every unit time. Consider a message transmitted at the first round of exchanges. If $|\mathcal{T}| = n$, then the fraction f of initially uninfected peers is $\frac{n-1}{n}$. Let f_i be the fraction of uninfected peers at round i. Since an uninfected peer can remain uninfected only by contacting another uninfected peer, f_i^2 peers remain uninfected with the update at round $i+1$. Thus, on an average:

$$\begin{aligned} f_{i+1} &= f_i^2 \\ &= f^{2^i} \end{aligned}$$

The number of uninfected peers drops doubly exponentially with time. Since the number of exchanges initiated by a peer is one per unit time, the number of messages sent and received by a peer is in $\mathbf{O}(1+\frac{1}{n})$.

2.6.2 Size of the message cache

The rates of message creation and deletion determine the size of the message cache. Let μ be the rate of message creation at the authentication protocol layer. Thus, the message cache receives μ new messages per unit time from the authentication protocol. Consider the situation at round i with respect to the messages created during the first round. Since the fraction of uninfected peers is same as the probability P_d of the destination being uninfected, we have:

$$P_d = f^{2^i}$$

Suppose the destination gets infected at round $i^\star$. Again, by the anti-entropy propagation of its stable read timestamp, we have the probability P that a peer is infected with the message but not with the stable read timestamp of the destination:

$$P = (1 - f^{2^i}) f^{2^{i-i^\star}}$$

Infection with the message but not with the stable read timestamp ensures that the message is not deleted. Hence, the expected fraction of cached messages is $(1 - f^{2^i}) f^{2^{i-i^\star}}$. However, a cached message could be created at any previous exchange round. Therefore, we have the following summation for the message cache size N:

$$N = \sum_{i=1}^{\infty} \sum_{i^\star=1}^{i} n\mu(1 - f^{2^i}) f^{2^{i-i^\star}}$$

Relating the series to the integral $\int e^{e^x} dx$, which is evaluated using integration by parts, we have the following relation on the message cache size:

$$N < \mu \frac{e \log e}{2} n \log(\frac{n+1}{n})$$

We know that $\log(\frac{n+1}{n})$ is in $\mathbf{O}(\frac{\log n}{n})$. Also, the rate of message insertion μ is in $\mathbf{O}(n)$ because messages are sent to all members in the trusted group. Hence, the number of cached messages is in $\mathbf{O}(n \log n)$.

2.7 Simulation

We devised a simulation system to investigate authentication cost in various scenarios. The simulation system is a stripped down version of the authentication module implemented in the library. All the message transfers are replaced by function calls that update counters to simulate system activity. The simulation is designed to replicate the authentication protocol in a universe of peers. The trust relationships between the peers can be preset during bootstrapping. Application level message exchange can be simulated to trigger authentication of unknown peers. The simulation consists of about 1500 lines of C++ code and uses the libc pseudo random number generator to make application level choices.

The scenarios of interest are bootstrapping and the authentication of new peers on an ongoing basis. The flow of time is uniformly measured in epochs with each epoch consisting of the time needed for one cryptographic operation and one message transfer. This way of measuring time allows extrapolation of the simulation results to systems having various tradeoffs of processing power and network latency.[6] It also allows us to state all simulation results in terms of number of messages exchanged by a peer.

2.7.1 Bootstrapping cost

The bootstrapping procedure requires the trusted peers to authenticate each other as if they were mutually probationary. This process was simulated with respect to the bootstrapping set size, the topology of the trust graph connecting the bootstrapping peers, and the proportion of malicious peers in the universe. We simulated the bootstrapping process on a universe of 1000 peers with trusted group sizes from 6 to 56.

We chose different types of trusted groups, as shown in Figure 2.8. Bidirectional trust was used to create clusters of mutually trusting peers. This is realistic in geographically local or

[6]The extrapolation should assign zero message transfer time and the measured cryptographic operation time to calculate the cost of Public key infection.

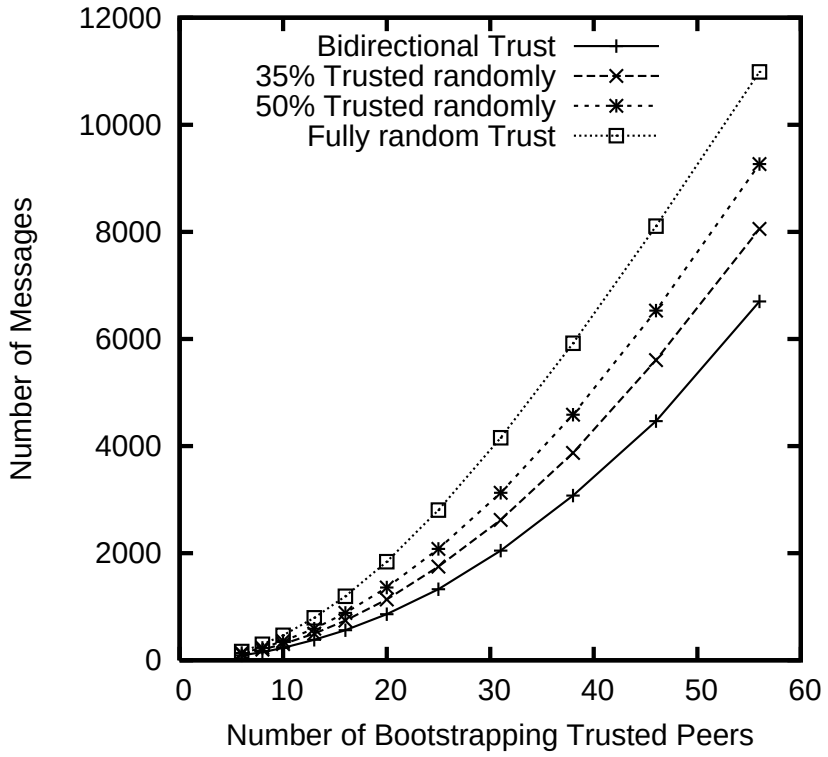

Figure 2.8: Average Bootstrapping cost per Peer.

in small world type of trusted environments. These clusters of bidirectional trust were perturbed by randomly selecting some of the members from the universe. This makes the trust relationship unidirectional, and imposes additional cost on the peers. The cost of bootstrapping increases because it becomes less likely that a trusted peer can return cached proofs of possession. We also note that the bootstrapping cost is quadratic in group size, as shown in Figure 2.8. This is expected because the bootstrapping peers authenticate each other.

The effect of malicious peers was studied for various group sizes. The malicious peers were randomly distributed in the universe. Their actions were not provably malicious on challenge response, thereby leading to the execution of Byzantine agreement. The trusted groups were created with 15% random selection. The results shown in Figure 2.9 show that the incurred bootstrapping cost increases rapidly with increasing group size and with increasing proportion of malicious peers. This is expected because each malicious peer forces an overhead of $O(|\mathcal{T}|^2)$ message transfers on each of its peers.

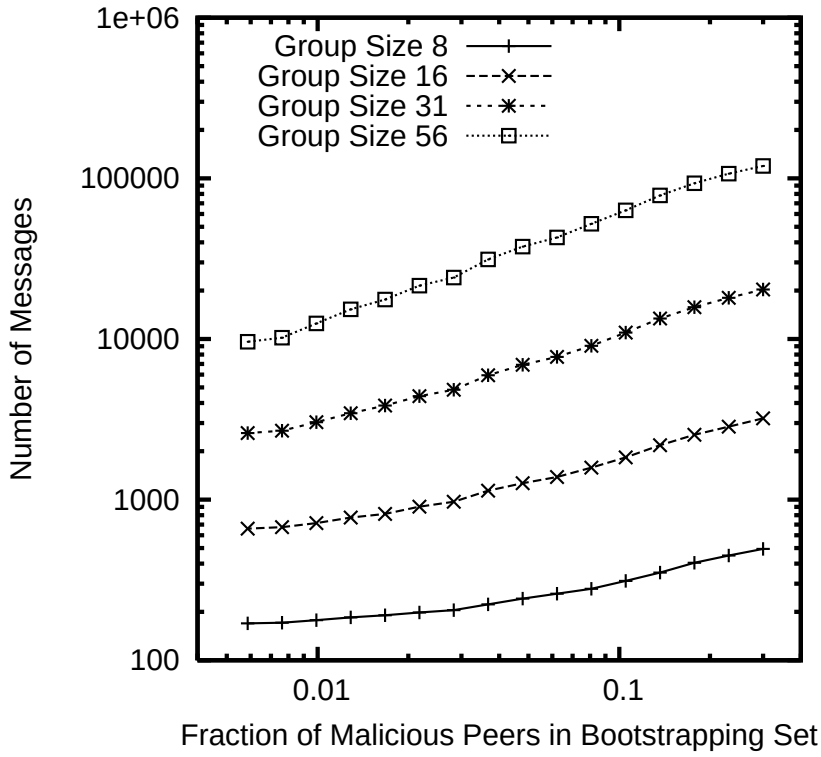

Figure 2.9: Effect of Malicious Peers on the Average Bootstrapping Cost.

2.7.2 Authentication cost

We investigated the messaging cost incurred by group members and the probationary peer for various group sizes. This simulation was conducted on a universe of 2000 peers with statistics collected in 1000 runs of 1000 application messages each. The application messages were sent to an unauthenticated peer with a probability of 0.1. As shown in Figure 2.10, the cost of authenticating a new peer is independent of group size for the trusted peers. The cost increases linearly with group size for probationary members. It is slightly cheaper to authenticate peers in groups with randomized selection because some of the challenge response steps are avoided. This happens if the probationary peer is already trusted by some group members. The role of malicious peers is investigated in Figure 2.11. Authentication cost shows a rapid increase with increasing proportion of malicious peers. The effect of malicious peers is greater on larger groups as expected. This happens because malicious peers cause the execution of the expensive byzantine agreement phase with a quadratic messaging cost.

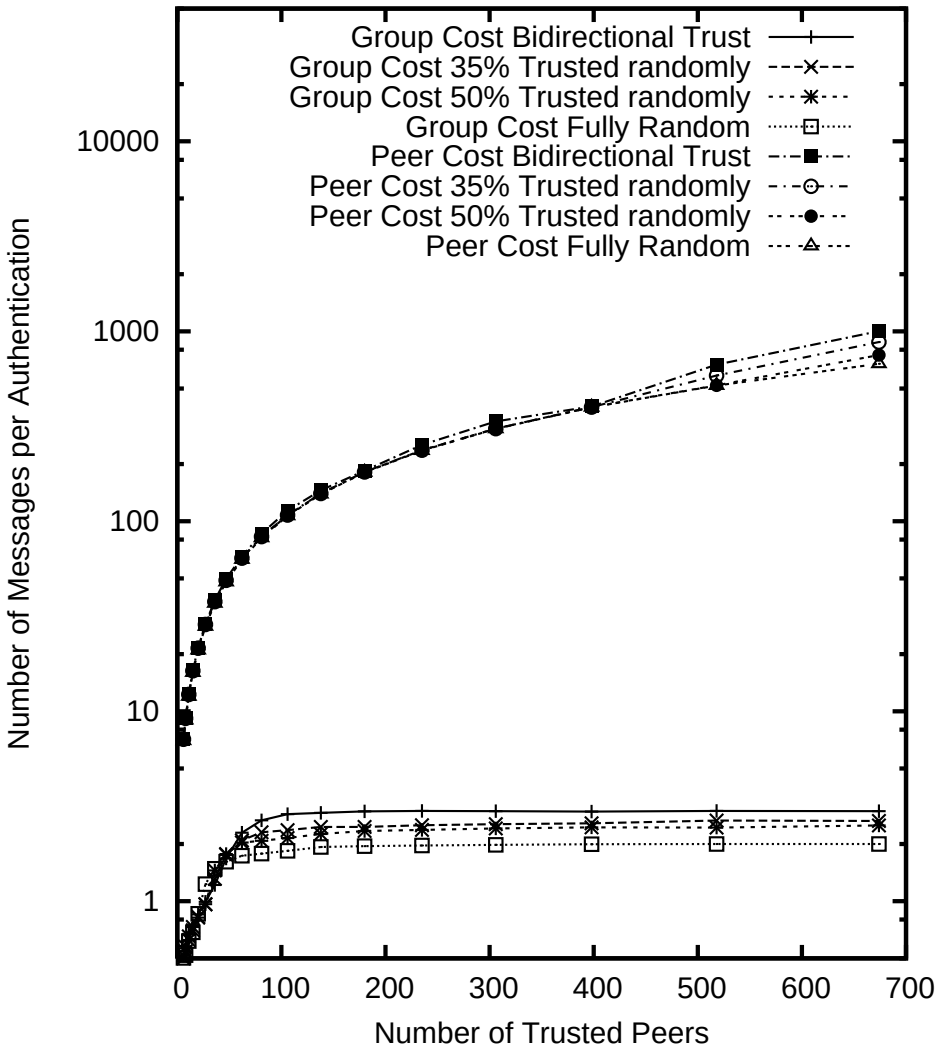

Figure 2.10: Cost of Authentication.

2.8 Discussion

Our model supports an *incremental growth of trust*. Optimistic authentication allows the trust to increase by the successful authentication of a public key through many group migrations. Since most of the peers are in honest groups, it becomes increasingly unlikely that a long sequence of dishonest groups is selected. Thus, our protocols provide *soft authentication*, which contrasts them from the traditional notion of authentication.

The traditional model with its all-or-none approach provides stronger authentication with weaker fault tolerance. We trade off the authentication strength and use stronger network assumptions to provide an autonomous and fault tolerant authentication mechanism. It can be

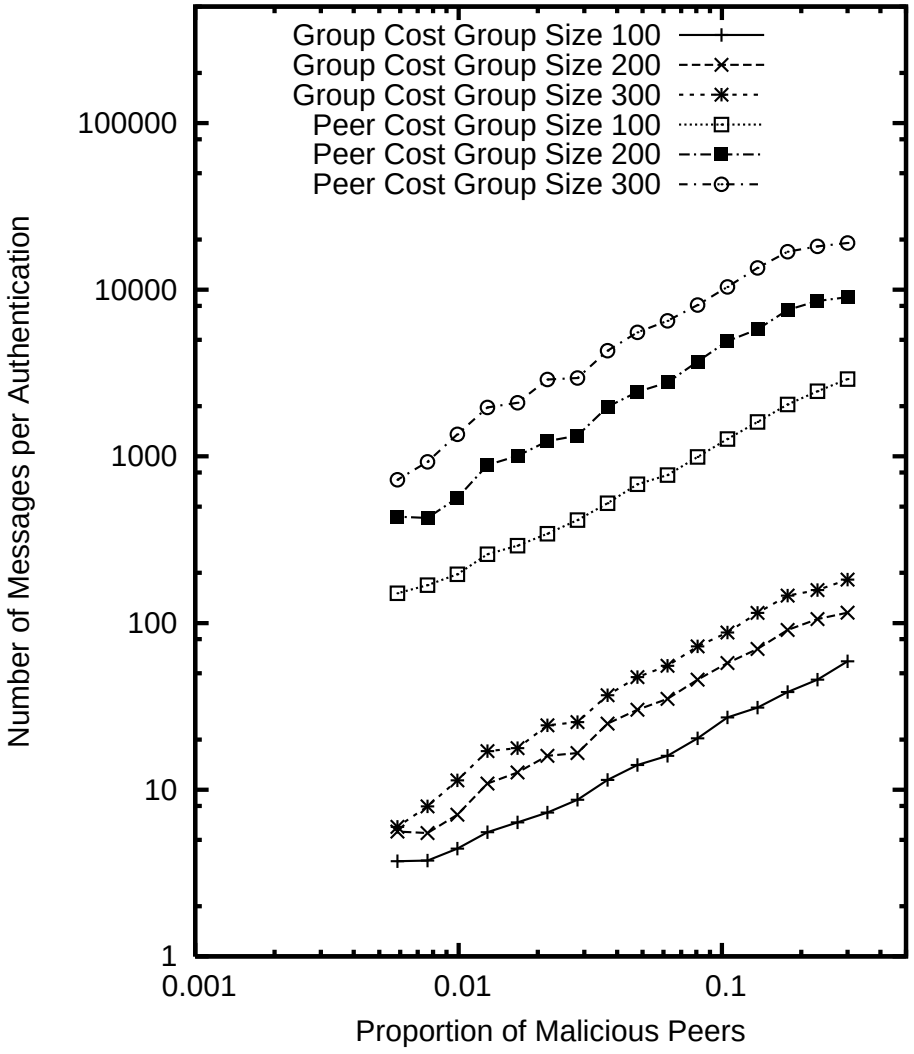

Figure 2.11: Authentication with Malicious Peers.

argued that the security infrastructure approach forces system designers to take a boolean approach to security. This has allowed most of the Internet traffic to remain insecure even though the computational power and software engineering needed to secure it are available. Byzantine fault tolerant authentication is useful because it allows the co-operative formation of self authenticating systems.

2.9 Summary

Byzantine fault tolerant public key authentication provides a new approach to tackle the authentication problems of peer-to-peer systems. The salient features are lack of total trust and single

points of failure. Our approach allows a natural growth of trust without requiring trustworthy hierarchies of delegating and recommending parties as is done in other trust management systems. Weakening the network adversary model makes our approach feasible. Although weaker than the traditional model in terms of adversary power, our authentication solution is stronger in terms of fault tolerance.

The rise of peer-to-peer systems on the Internet is the motivation for developing the byzantine fault tolerant public key authentication protocol. In the following chapter, we extend this approach to social groups of email users.

Chapter 3

Improving Email Trustworthiness through Social-Group Key Authentication

3.1 Problem statement

Electronic mail is one of the most popular applications on the Internet. Unlike traditional mail that can be signed by hand, electronic mail does not have a built-in authentication mechanism. In particular, the absence of sender authentication makes it possible to spoof sender identity. It is also possible to modify message contents en-route because messages do not carry digital signatures, which could provide message authentication. The lack of sender authentication and message authentication limits the effectiveness and trustworthiness of email. It is non-trivial to determine the true identity of the sender because messages could be spoofed, i.e. appear to be from a different sender than the real sender. The low cost of sending electronic mail coupled with ease of spoofing has led to a flood of spam on the Internet. Having sender authentication would not only contain spoofing, but also enable tackling the spam problem by using authenticated sender identities to classify messages as trusted or otherwise. Similarly, message authentication would increase trustworthiness of electronic mail making it more effective for personal and business use. These motivations make sender authentication and message authentication important enhancements to email.

While the original email specification [20, 73] does not address authentication, the S/MIME enhancements [82, 83] have added support for message authentication. Message authentication in S/MIME depends on sender authentication, which is provided by an external public key infrastructure (PKI). This works well in an organizational setting, where a central trusted party can certify public keys associated with all the email addresses. However, the centralized trust model becomes unsuitable for communications across organizational boundaries or for private communication through free email systems. Since the email user base is decentralized with peers belonging to different logical trust domains, the authentication infrastructure should be decentralized too. This requirement is not addressed by the S/MIME standard.

A popular security add-on for electronic mail is Pretty Good Privacy, commonly known as PGP [109]. It allows users to authenticate public keys of other users in a peer-to-peer manner. Human judgment of trustworthiness guides the authentication decisions. Authenticated public

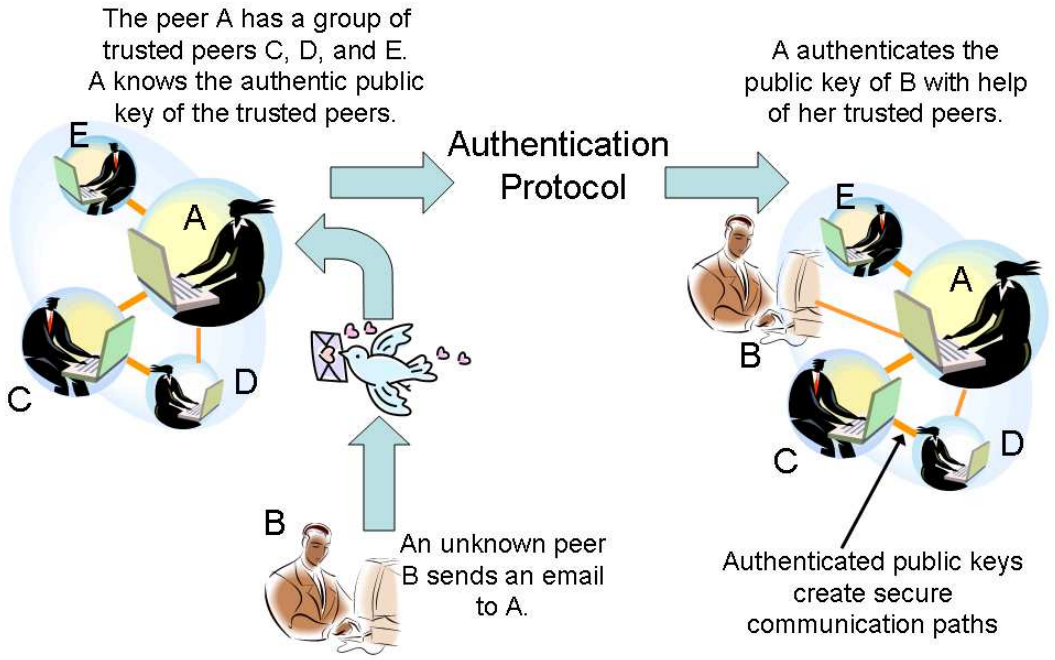

Figure 3.1: The big picture: public key authentication. *A* authenticates the public key of *B*.

keys can provide both sender authentication and message authentication through digital signatures [80]. Although PGP has been freely available for over a decade, it is yet to be adopted by a majority of mail users. A number of factors contribute to its limited use. Firstly, it requires a significant level of sophistication to evaluate the trustworthiness of peers. It can be argued that common users are either unwilling or unable to make these decisions [99]. Secondly, even if the users are willing to put a one time effort to manage their trusted peer groups, known as key rings in PGP terminology, they may be unwilling to invest this effort on a regular basis. The vast user base of electronic mail views it as a plug-and-play service with free email accounts available from a number of providers. Therefore, a suitable authentication method must be autonomous, i.e., able to run with minimal or no human input. This requirement is not addressed by PGP.

Therefore, we believe that a widely acceptable electronic mail authentication solution must support the following requirements:

1. Operate without depending on centralized third parties for authentication decisions.

2. Provide autonomous operation with minimal human intervention.

Fortunately, the authentication requirements for electronic mail are less strict than those for authorization and access control. Let authentication be called *eventual authentication* if it is achieved with a finite delay, and if the extent of the delay depends on the communication patterns of the participants. We note that eventual authentication is acceptable in the email environment because of its asynchronous communication pattern. Eventual authentication is also useful because it can overcome the false positives of spam filters and provide confidence in the received message, albeit with a delay. On the other hand, while email communication crosses organizational and administrative boundaries, it typically occurs within social groups of collaborating individuals. Therefore, it is reasonable to assume that the social groups of collaborating individuals have honest majority.

3.1.1 Our solution

Our social-group key authentication proposal for email (SGKA) is described and evaluated in this chapter. The proposed solution is an instantiation of our byzantine fault tolerant public key authentication protocol (BPKA) described in Chapter 2, which supports soft authentication of public keys without centralized infrastructure. The social-group key authentication protocol authenticates public keys of email users. It runs as an overlay on the email protocol [54], thereby supporting incremental deployment and backward compatibility. Digital signatures [71] generated from the authenticated public keys provide sender and message authenticity to email.

Our public key authentication protocol provides eventual authentication. This means that users may receive digitally signed messages from peers whose public keys are yet to be authenticated. While eventual authentication of sender public key would authenticate both sender identity and message content, there exists a period before eventual authentication when the public key authenticity is undecided. This is different from a false negative where the public key of an honest peer cannot be authenticated. We note that while authentication can be delayed, there are no false negatives in public key authentication as shown in Section 2.5. Similarly, there are no false positives under the honest majority assumption. Since the underlying public key authentication protocol is autonomous and decentralized, social-group key authentication inherits these characteristics. Authentication is supported in an end-to-end manner without additional infrastructure or human input. Our solution is therefore compatible with the usability requirements described above. It also allows incremental deployment and preserves backward compatibility with existing email infrastructure. In summary, this chapter discusses the following contributions:

- We implement social-group key authentication for email. Our solution is automatic, byzantine fault tolerant, eventually correct, incrementally deployable, backward compatible with the existing email infrastructure, and does not use trusted third parties.

- Performance of the proposed solution is investigated through micro-benchmarks, simulation on an industrial and an academic email trace, and live experimentation on an instrumented mail authentication prototype.

3.2 Social-group key authentication protocol

The secure association of public keys to email addresses is referred to as *public key authentication* in this chapter. This section explains how the BPKA protocol proposed in the previous chapter is applied to the email environment.

3.2.1 Email setup and security model

The BPKA protocol assumes that the participating peers are identified by their network addresses, which are email addresses in the context of this chapter. Based on this premise, we do not distinguish the email address A from the user who uses that address. We assume that every user U has a public key (K_U) and a private key (K_U^{-1}). Every email message contains the public key of the sender and is signed by the sender using his or her private key.

The BPKA protocol requires that the asynchronous network connecting the peers provide delivery failure notifications for non-existent destinations. The network should support eventual delivery on retransmissions, and not become permanently partitioned. Assuming that temporary failures in the email network are eventually repaired, the email network satisfies these requirements [54].

Public keys are authenticated with help of a group of peers called the trusted group:

DEFINITION 3 (Trusted Group) *The* trusted group *is used for authenticating public keys of new peers. On authentication of its public key, the new peer becomes part of the trusted group. The public key of every peer belonging to the trusted group is known and trusted.*

The trusted group is initialized from the address book of the user. We note that belonging to another peer's trusted group does not affect the authentication protocol. Because the authentication protocol requires message transfer between trusted peers, additional extension fields are added to email headers.

DEFINITION 4 (Email Header Extension) *The following email header extension fields are used by social-group key authentication protocol for public key authentication* [1]*:*

- *X-Bft-Auth-PublicKey : public key of the sender.*

[1] Bft in the email headers stands for Byzantine fault tolerance.

- *X-Bft-Auth-Data : unauthenticated public key of other peers, nonces, cipher text, or trust decisions.*

- *X-Bft-Auth-MesgInfo : the protocol operation that sends out the message and the specific stage within that operation (for operations that have multiple stages). Protocol operations can be one of the following:* EMAIL_PEER *,* EMAIL_RESPONSE *, or* INFER_TRUST.

- *X-Bft-Auth-Signature : digital signature signed with the private key of the sender.*

Using SMTP extension header fields for carrying social-group key authentication data provides backward compatibility. The email messages sent by the authentication-enabled mail clients would contain social-group key authentication protocol messages, which are processed by the email clients supporting the protocol. The additional protocol messages are ignored by other email clients because email systems should ignore unknown extension headers [73].

3.2.2 Adversarial model

We assume the following strong adversarial model. Adversaries mounting passive attacks are allowed to overhear all the communication between peers. The active attacks are restricted compared to the classical "network is the adversary" model as follows: The active adversaries have unlimited spoofing power, i.e., they can inject arbitrary messages into the network. However, they have limited power to prevent message delivery. In particular, for the BPKA protocol to operate at a peer P, it should be impossible to prevent (eventual) message delivery for more than a fraction ϕ of P's peers. We note that since email servers are widely distributed, a practical value of ϕ is zero for general email communication over the Internet.

Peers in the trusted group can be honest, malicious, or faulty. The protocol does not distinguish between the latter two cases, but provides public key authentication service to the honest peers. The protocol correctly authenticates the public keys of honest peers if the trusted group has honest majority.

DEFINITION 5 (Honest Majority) *A trusted group has* honest majority *if fewer than t of the n trusted peers are malicious or faulty, where* $t = \frac{1}{3}n$*. A peer is malicious if it does not follow the protocol correctly, and faulty if its authentication vote is incorrect.*

For example, a faulty peer may suffer man-in-the-middle attacks causing it to vote incorrectly while a malicious peer may intentionally give wrong authentication votes.

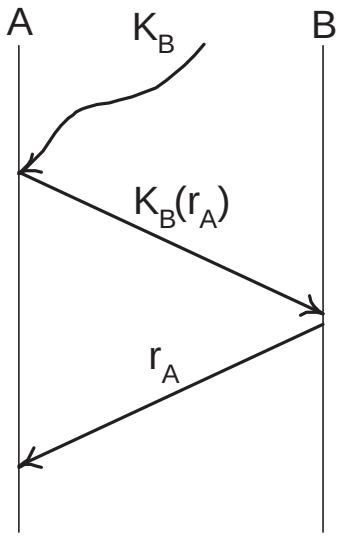

Figure 3.2: Challenge response in BPKA protocol. A uses the nonce r_A to authenticate the public key K_B of B in absence of man in the middle attack.

3.2.3 Our protocol

The purpose of our protocol is to let Alice authenticate Bob's public key with help from the peers in her trusted group. Our social-group key authentication protocol has the following operations: EMAIL_INIT, EMAIL_PEER, EMAIL_RESPONSE, and INFER_TRUST. The protocol operations are described below along with the exchanged messages. For brevity, only the contents of X-Bft-Auth-Data and X-Bft-Auth-MesgInfo email extension headers are described. The remaining extension headers are populated as follows: Public key of the sender is stored in X-Bft-Auth-PublicKey extension header, and the X-Bft-Auth-Signature extension header stores the digital signature created with the sender's private key.

- EMAIL_INIT: Alice receives an email message from Bob whose public key K_{Bob} is not authenticated.

$$Bob \quad \rightarrow \quad Alice$$

- EMAIL_PEER: This operation is run by Alice. Alice emails the peers in her trusted group $A_1, \ldots, A_n$ for authenticating K_{Bob}, the unauthenticated public key of Bob. The email message has type EMAIL_PEER in the X-Bft-Auth-MesgInfo header, and key K_{Bob} in the X-Bft-Auth-Data header. For all $i \in [1,n]$, we use below formula to represent the email message sent by Alice to peer A_i in her trusted group.

$$Alice \quad \rightarrow \quad A_i \qquad K_{Bob}$$

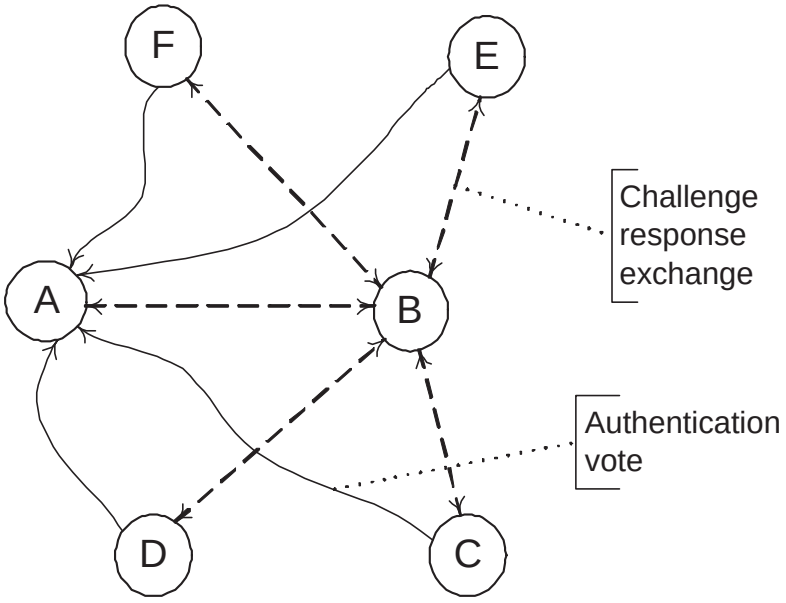

Figure 3.3: Distributed authentication in BPKA protocol. A authenticates the public key of B by gathering authentication votes from its trusted peers C, D, E, and F.

- EMAIL_RESPONSE: This operation is run by each A_i with the participation of Bob. As shown in Figure 3.2, the peer A_i runs the CHALLENGE RESPONSE operation of BPKA protocol and decides if the public key K_{Bob} of Bob is authentic or not. The challenge consists of a random number r_{A_i} chosen by A_i and encrypted with K_{Bob}, the supposed public key of Bob. In response, Bob is expected to recover the random number r_{A_i} chosen by A_i, and demonstrate its ownership of the public key K_{Bob}. The detailed steps of this operation are given below. Each message has the type EMAIL_RESPONSE stored in the X-Bft-Auth-MesgInfo header.

$$\begin{array}{ccccc} A_i & \rightarrow & Bob & & K_{Bob}(r_{A_i}) \\ Bob & \rightarrow & A_i & & r_{A_i} \\ A_i & \rightarrow & Alice & & T_{A_i}(Bob) \end{array}$$

  Each peer A_i emails back its *trust vote* $T_{A_i}(Bob)$ to Alice. The trust vote consists of the signed challenge message sent by A_i, the signed response sent by Bob, and a true or false vote on authenticity.

- INFER_TRUST: This operation is mainly run by Alice and may also require the participation of Alice's peers and Bob. Alice's inputs are the trust votes received from her peers A_i. If the trust votes are in agreement on the authenticity of K_{Bob}, then Alice decides according to the unanimous decision of her peers. This inference is based on the DISTRIBUTED AUTHENTICATION operation of BPKA protocol shown in Figure 3.3.

If Alice receives disagreeing trust votes from her peers, she initiates the BYZANTINE AGREEMENT operation of BPKA protocol, which allows Alice to determine who among Bob or her peers is malicious or faulty. We note that Bob needs to participate in the BYZANTINE AGREEMENT step because either Bob or any of Alice's peers may be malicious or faulty. Alice sends the vector of received trust votes to Bob and her peers A_i. On receipt of this message, Alice's peers and Bob exchange the trust vote vectors among themselves. Using the symbol "|" to denote multiple sources or destinations, the messages exchanged in this protocol operation are shown below. Each of the messages contains INFER_TRUST in the X-Bft-Auth-MesgInfo header.

$$\begin{array}{ccccc} Alice & \rightarrow & A_i|Bob & & T_{A_i}(Bob) \\ A_i|Bob & \rightarrow & A_j & & T_{A_k}(Bob) \end{array}$$

Alice decides whether or not to trust Bob's key by majority on the trust votes. This part of the authentication protocol also permits Alice and her peers to identify and exclude malicious or faulty peers from trusted groups.

3.2.4 Security of our protocol

Our social-group key authentication protocol is secure against the adversarial model defined in Section 3.2.2, assuming the trusted group has honest majority (see Definition 3). Its security is based on the security of the BPKA protocol (discussed in Section 2.5) because its operations are aggregations of BPKA operations. The embedded BPKA operations are also executed in the same sequence. Since the email environment satisfies the requirements of the BPKA protocol (as discussed in Section 3.2.1), the security of our social-group key authentication protocol follows from the security of the BPKA protocol.

3.2.5 Complexity of the protocol

The complexity of authentication with the social-group key authentication protocol is same as that of BPKA described in Section 2.4.4. Given n peers with public keys of length k, the number of messages to authenticate a peer is $O(n)$, the message size is $O(k)$, and the computational cost is $O(\text{poly}(k))$. This equivalence follows from the same argument given in Section 3.2.4.

3.3 Implementation of email authentication

We implemented social-group key authentication as a plugin for Thunderbird email client from the Mozilla application suite. This section outlines the design issues, application choices, and practical considerations encountered during its design and implementation. An overview of the plugin architecture is also provided.

The Mozilla suite of applications [3] allows developers to extend application functionality by developing plugins. XPCOM objects are the basic unit of plugin development. These objects allow run time linking and expose their interface through a compiled interface definition file. A compiled XPCOM object can be accessed as a first class Javascript object from the user interface controlling scripts. The user interface itself is defined through the XUL user interface language with Javascript making XPCOM calls on receiving user interface events. The entire package of compiled XPCOM objects, user interface elements, and controlling scripts is referred to as a plugin. We followed the standard procedure [2] to embed BPKA library in Thunderbird in order to provide social-group key authentication for email.

The email authentication plugin architecture is shown in Figure 3.4. *Authentication Adapter* is an XPCOM object that exposes the authentication interface. It is statically linked to the Byzantine fault tolerant public key authentication (BPKA) library. The authentication adapter interface provides authentication protocol messages to be attached to outgoing emails, and consumes the protocol messages from incoming emails. The interface also contains calls to query and authenticate the public keys associated with email addresses. The authentication adapter is used for implementing social-group key authentication. Its functionality is integrated into the Thunderbird email client through the *Scripted Extension Access* module. In Figure 3.4, the gray box indicates the standard Thunderbird email client, while the remaining boxes show the major components of our plugin implementation.

3.4 Overlay considerations

We use SMTP extension headers to create an overlay for the social-group key authentication protocol. This maintains compatibility with existing email infrastructure because legacy email clients should ignore unknown extension headers [73]. Running the protocol as an overlay on top of email introduces performance limitations and design constraints. This section investigates these issues in order to choose implementation parameters that are practical in the email environment.

3.4.1 Trusted group size limits

We evaluated the overhead of BPKA protocol through a simulation study in Section 2.7. The cost of public key authentication depends on various controllable parameters like bootstrapping group size, trusted group size, probationary group size, and the rate of authentication of new peers. These parameters must be selected in order to match the computational and messaging power available in typical email systems with the requirements of the protocol.

The authentication protocol can operate as an overlay above the mail transport by using the extension fields defined in SMTP. This is in line with many anti-spam implementations. However, SMTP mail transfer agents impose a limit on the maximum header size. This is done in

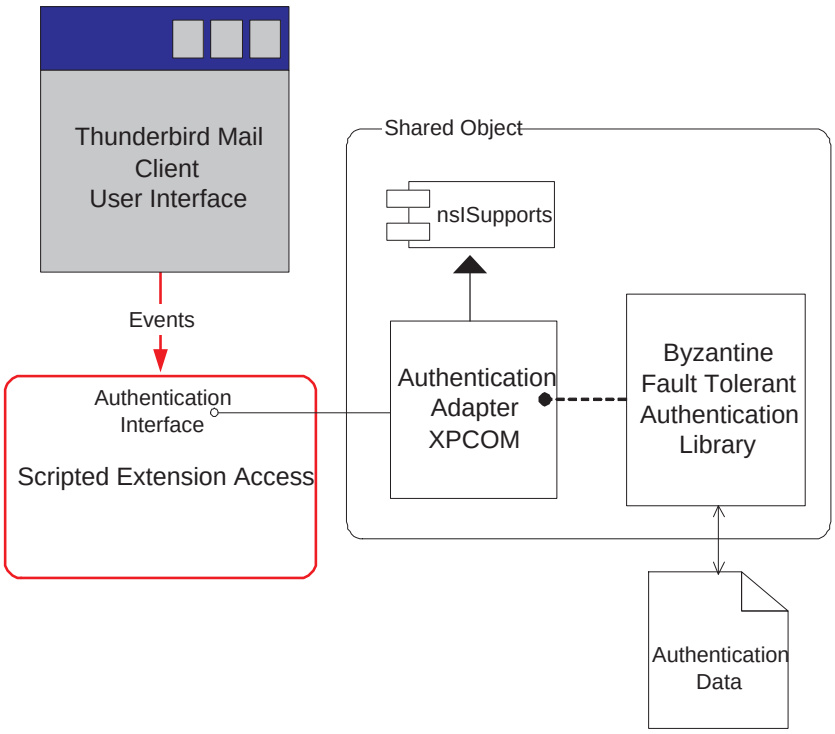

Figure 3.4: Authentication plugin architecture for the Thunderbird email client.

order to avoid denial of service attacks. For example, `sendmail`, a popular UNIX mail transfer agent, supports the maximum header size of 32 KB. This limits the maximum authentication payload that can be attached to a single message. Since the authentication protocol requires increasing amounts of messaging overhead with increasing trust group size, the maximum group size that can be supported in the overlay is limited. Using a public key size of 1Kb, and ZLIB library for compression, we tested the final header load for different authentication message payloads. A high compression ratio can be achieved on the authentication messages because they are are serialized in XML format. Figure 3.5 shows the size overhead of authentication messages resulting from the university mail trace described in Section 3.5. We choose a payload of at most 300 compressed authentication messages in order to impose less than 10KB overhead on the mail header.

Messaging cost of authentication depends on the trusted group size and the rate of discovery of new peers. The budget of 300 authentication messages per email affects the maximum size of trusted group that can be maintained. Getting hold of mailbox statistics is challenging because of privacy issues. Therefore, we gathered statistics of unique mail addresses and number of messages from the mailboxes of a few colleagues. The results indicate that about 20% of the messages are sent to, or received from new peers, and need to be authenticated. Applying this ratio to the limit of 300 authentication messages per email, we can afford 1500 authentication

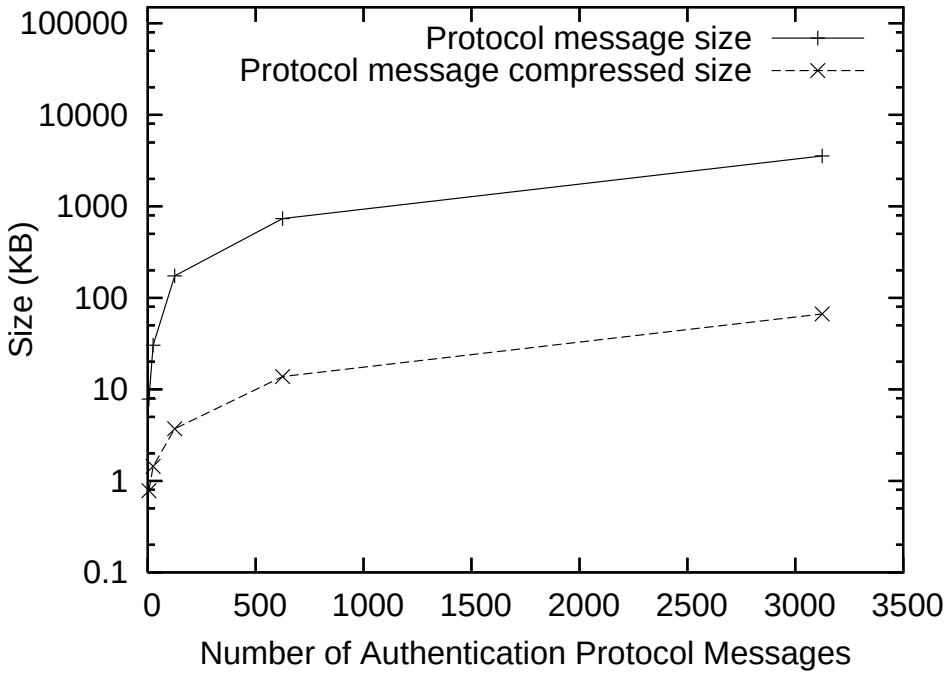

Figure 3.5: Authentication message size with and without compression. ZLIB compression is used on raw authentication messages created for 1Kb sized public keys.

messages per un-authenticated peer. Our previous simulation results in Section 2.7 indicate a maximum trusted group size limit of 75 peers for this messaging cost. This upper limit on trusted group size is designed into the system.

Designing this limit into the system ensures that the new peers can be successfully bootstrap the system at any given point of time. We also note that while the cost on trusted group is independent of group size, the peers to be authenticated do incur an increasing messaging cost with group size. Since the messaging cost on the probationary peer is same as the bootstrapping cost, the proposed group size limit is also appropriate for continuous operation.

3.4.2 Bootstrapping groups

To determine a meaningful heuristic for generating bootstrapping groups (see Section 2.7), we analyzed the email communication patterns available from the anonymized University email trace (described in detail in Section 3.5). Figure 3.6 shows the cumulative distribution of number of user accounts with respect to email messages sent or received over a 92 day period. We find that a large number of user accounts are idle with minimal sending and receiving activity. Using the distribution, we cut off accounts that do not have at least 3 outgoing messages and at least 9 incoming messages over the period of the study. This reduces the number of

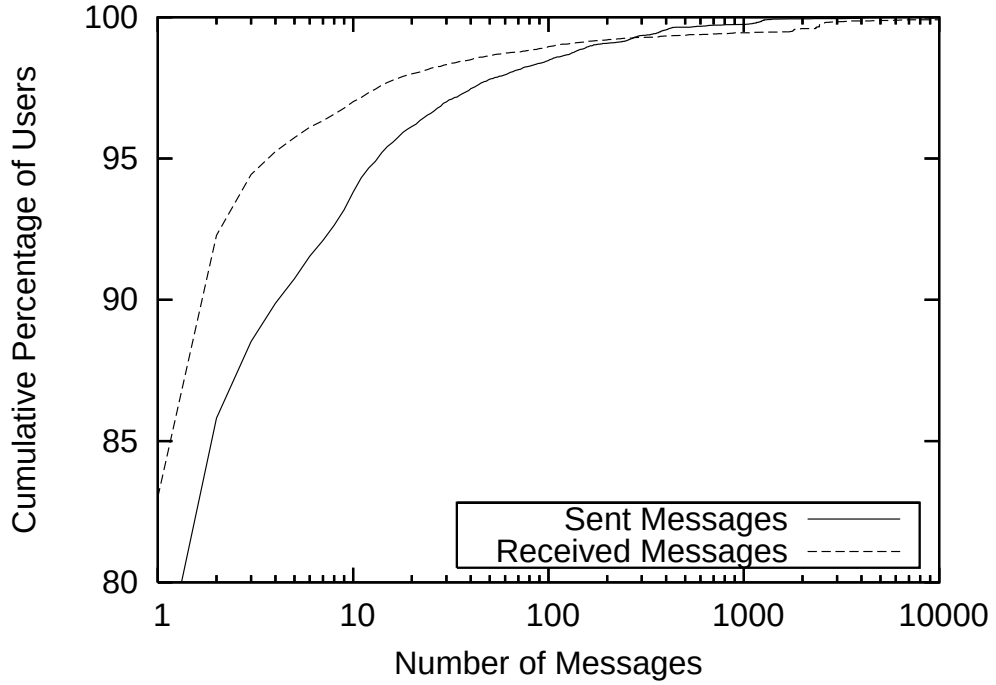

Figure 3.6: Activity profile of user accounts over the 92 day email trace period. The trace has 27,623 user accounts and 1.19 million emails.

user accounts in the study from 27,623 to 715. This active subset of user accounts is analyzed against two possible heuristics for generating bootstrapping trust groups: The *Outgoing heuristic* selects bootstrapping peers from destination addresses of outgoing emails. The *Two-way heuristic* selects bootstrapping peers from both the destination addresses of outgoing emails and the source addresses of incoming emails.

The selection heuristics are applied to the mail trace by considering the first 10, 30, and 90 days of the trace. Using the initial subset of the trace is desirable because future communication patterns will not be available in real life. The size of the bootstrapping group for each mailbox is calculated using the given heuristic and time window from the mail trace. The cumulative number of mailboxes having more than a given number of bootstrapping peers is plotted in Figure 3.7. It can be observed that one way communication is quite common in email as shown by the gap between the two heuristics. In order to have a frequently communicating subset of users, we apply the 30-day two-way heuristic on the 92 day mail trace. This results in a set of 53 peers that have at least 4 peers in their bootstrapping group. This subset of active users is chosen as the experimental base.

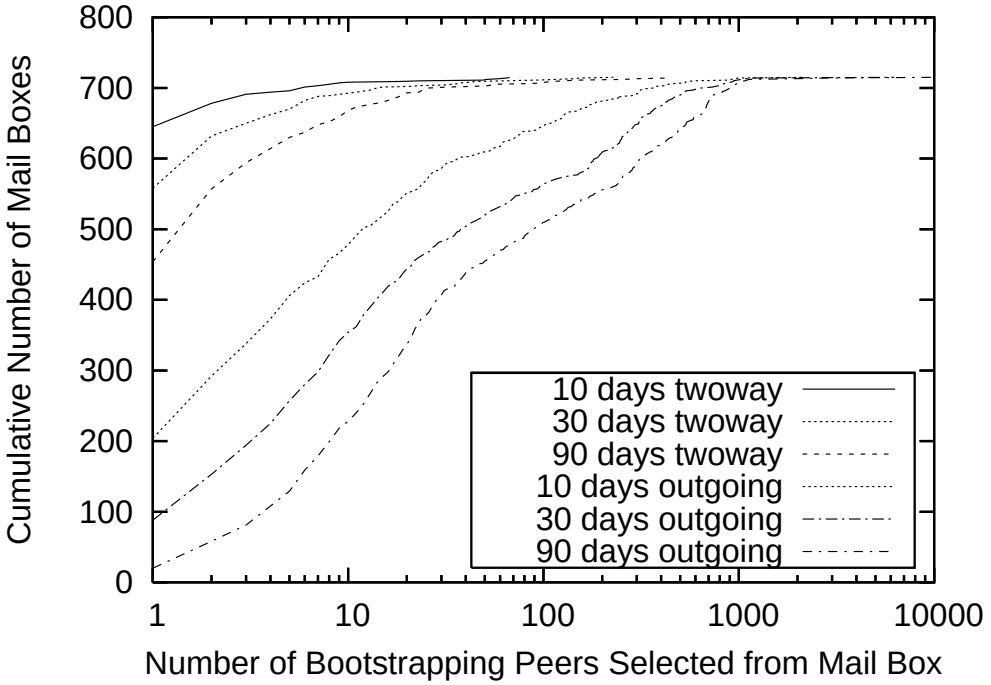

Figure 3.7: Selection of bootstrapping groups for 715 active user accounts in the email trace.

3.4.3 Handling the email communication model

A characteristic feature of electronic mail is that the sender and receiver do not need to be connected at the same time. This offline nature of email requires changes to the epidemic public key infection algorithm that works with anti-entropy sessions between peers. Anti-entropy sessions determine the set of messages to be sent by checking the time-stamp vectors at the receiver [22, 29, 56]. We modify this approach to compute the set of messages based on stale timestamps from a previous receive event. This approach effectively splits the anti-entropy exchange into two one sided sends. It increases the cache usage because stale timestamps require sending a larger number of messages. Since the receiving peers would silently ignore duplicate messages, the modified connectionless epidemic algorithm maintains the functionality of public key infection.

Email supports one to many communication. This communication pattern does not match the peer-to-peer anti-entropy sessions. This mismatch is handled by computing the set of outgoing messages based on all the receivers. Having multiple destinations forces a pessimistic choice of messages by selecting messages based on the most out-of-date peer. This modification affects the performance of the protocol by carrying extra messages. However, correctness is not affected since each potentially undelivered message is transmitted.

Trace	Number of messages	Time duration
University	1197043	92 days
Industry	2549767	56 days

Table 3.1: Email traces used for evaluation.

3.4.4 Eager and lazy modes

The authentication mechanism can be run in lazy or eager modes. In lazy mode, the authentication plugin does not proactively send out any email messages specifically for the key authentication purpose. The protocol messages are therefore transmitted entirely through organically exchanged emails in a piggybacking fashion. In the eager mode, additional plugin generated email messages may be sent out to peers. These messages would be automatically handled and absorbed at the receiving end plugin, and therefore would not change the user experience. We note the downside of eager mode that the added protocol messages may be consumed by spam filters. This problem can possibly be addressed by human means, by asking the mail administrators to disable particular spam filters. However, losing eager mode authentication messages only causes delay. It does not affect correctness because the lazy mode protocol will eventually achieve authentication by piggybacking on user emails.

3.5 Experimental evaluation

The objective of experimentation is to characterize client costs, and to establish the suitability of social-group key authentication in a real life scenario. The experimentation is done in two stages. The first evaluation is a micro-benchmark consisting of sending and receiving messages from an instrumented authentication plugin. The second evaluation consists of localized execution of two anonymized email traces, one from a university and another from the industry. The details of the traces are given in Table 3.1. The university trace is collected from a `sendmail` log behind the spam filter, while the industry trace is collected from the Internet mail gateway ahead of the spam filter. Statistics are collected for data overhead imposed on email messages, cache size at the peers, and the performance of authentication. Experiments are also done for comparing the performance of eager and lazy mode authentication.

3.5.1 Micro benchmarks

A set of micro benchmarks was conducted on a 2.4GHz Intel Pentium 4 desktop running LINUX Fedora Core 5. The objective of micro benchmarks is to determine the latency introduced by the addition of authentication plugin in the email processing path. The added latency of sending and receiving emails was measured for different public key sizes, as shown

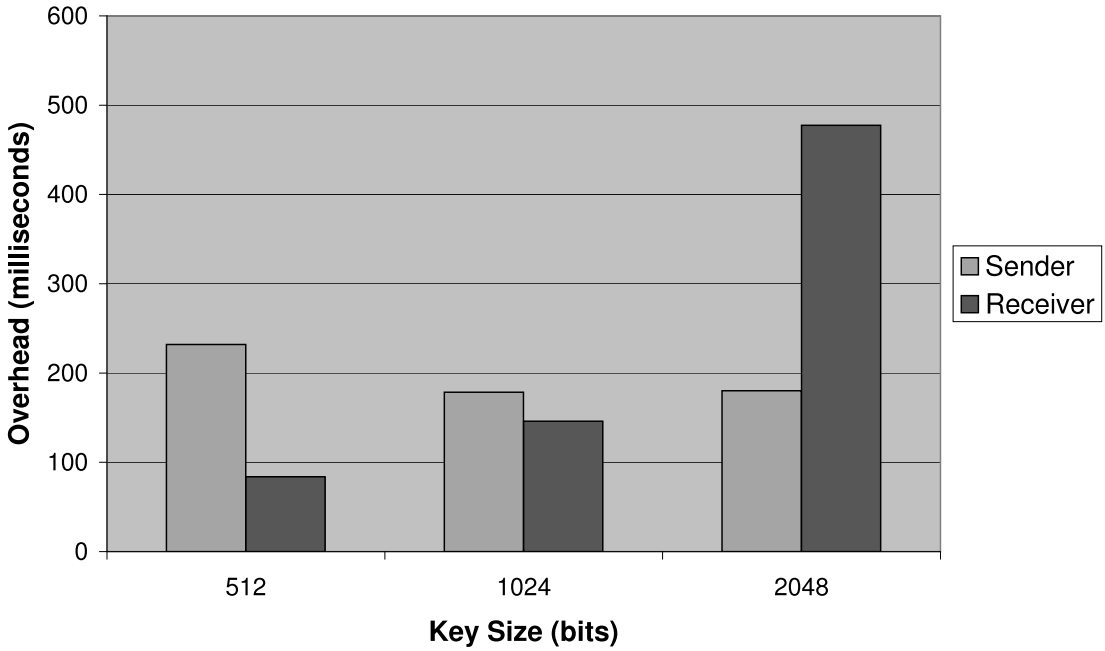

Figure 3.8: Additional email processing latency for authenticating public keys of different sizes.

in Figure 3.8. The sender cost was about 200ms for all the different public key sizes. Sender latency does not depend on public key size because it is dominated by message serialization costs. On the other hand, the receiver costs are dominated by the cryptographic operations of digital signature verification and responding to challenges. As shown in Figure 3.8, the receiver costs increase from 85ms for 512 bit keys to about 500ms for 2Kb keys. While both of the costs are within usability limits, one can observe that receiver processing can be done asynchronously in a separate thread. Therefore, one can expect a net addition of about 200ms latency to email operations due to the authentication plugin.

The effect of trusted group size on authentication plugin overhead was also measured, as shown in Figure 3.9. The overhead on the sender increases with increasing trusted group size because of the increasing overhead of serializing a larger number of messages for trusted peers. The overhead increased from 160ms for a trusted group size of 8 to 220ms for a trusted group of 18 peers. The receiver overhead does not depend strongly on trusted group size and takes about 65ms. The overhead of compression and making function calls across the authentication interface were measured and found to be less than 10ms in all the cases. These overheads are not sensitive to authentication protocol operational parameters, as expected. Sending overhead depends on trusted group size, while the receiving overhead depends on key size. Since the overhead introduced by the plugin is less than 500ms in all the cases, it is extremely reasonable from a usability standpoint.

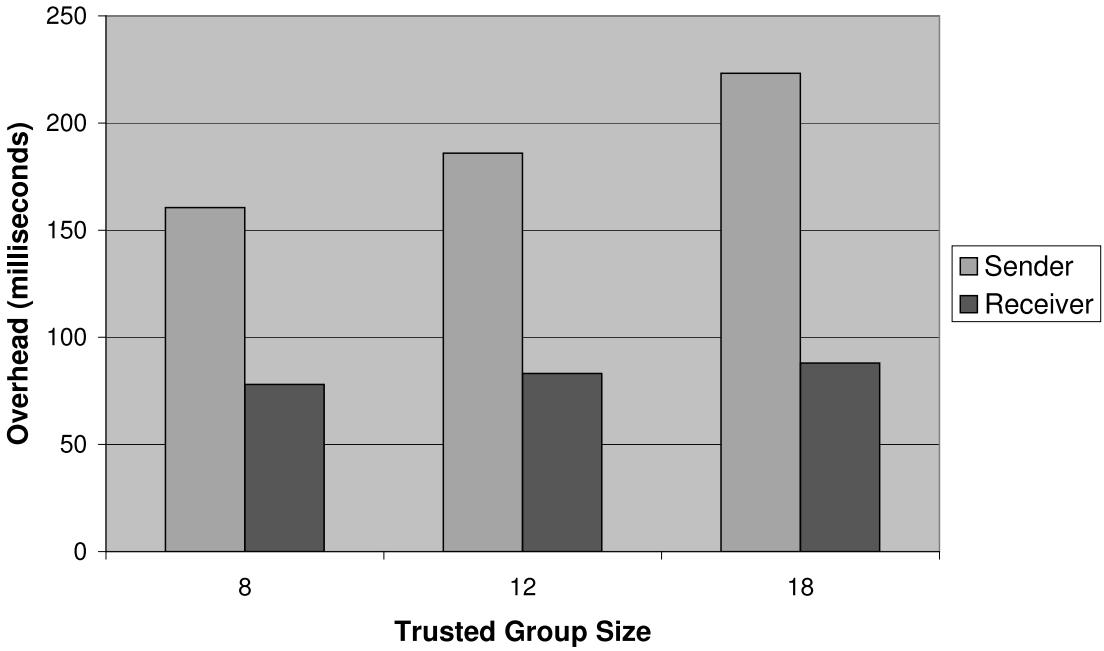

Figure 3.9: Additional email processing latency for authentication with different trusted group sizes.

3.5.2 University workload

As described in Section 3.4.2, the email trace is trimmed to contain email interactions of 53 peers that have bootstrapping groups of size 4 or more. A maximum size of 10 is chosen for the bootstrapping group in order to limit the processing time of the trace. The trimmed trace has 873,752 email messages as compared to the original 1.19 million. This mail trace is used to drive the authentication system on a single computer. The resulting message overhead, cache size, and authentication progress are collected from the logs. We experiment with different values of the following controllable parameters of the authentication system: trusted group size, expiration time for detecting non-liveness of peers, and the maximum number of protocol messages that can be attached to an email message.

Bootstrapping group selection

The authentication protocol performance is sensitive to bootstrapping group selection. In order to ensure progress, the initial candidates were filtered through a two way communication rule as discussed earlier in Section 3.4.2. Experiments were conducted for understanding the suitable bootstrapping group size in the email environment. Bootstrapping groups of sizes 8, 16, 32, and 64 were selected as shown in Figure 3.10. A number of selection methods were developed. The serial and random methods shown in Figure 3.10 select bootstrapping peers by first seen, and by uniform random selection on the candidates respectively. The product criterion prefers peers with a higher product of sent and receives messages. The balance criterion

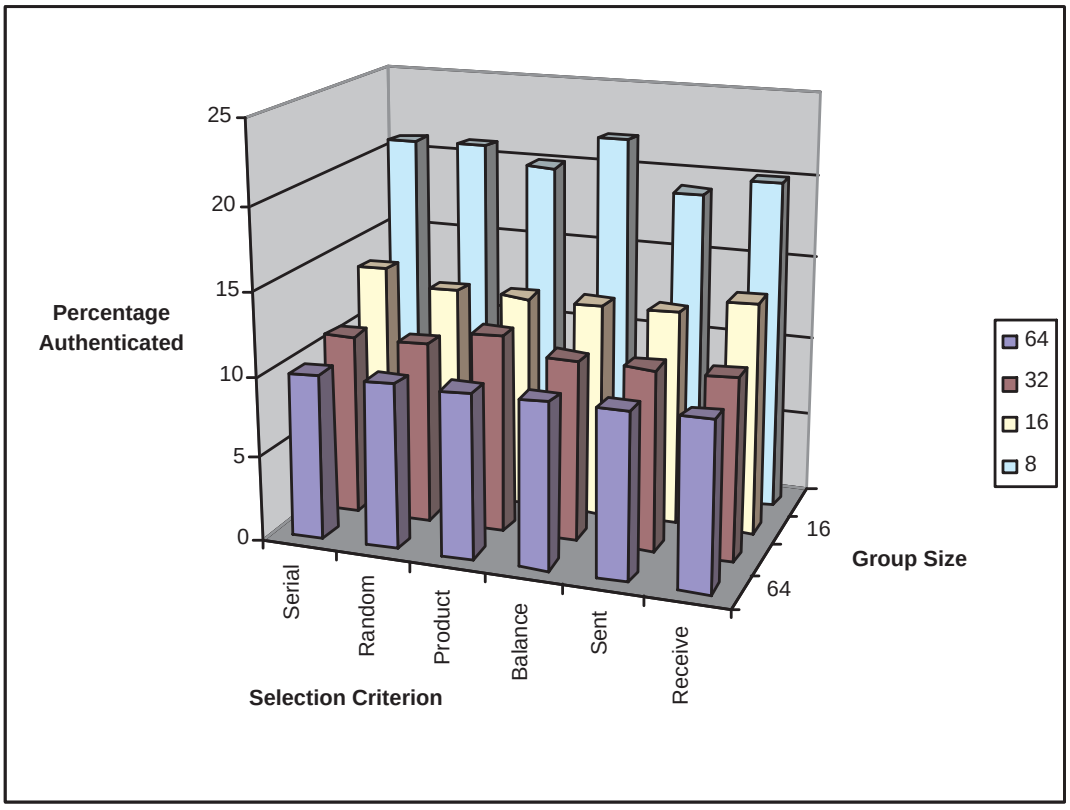

Figure 3.10: Variation of authentication performance with the size and selection criterion of the bootstrapping group.

prefers bootstrapping group candidates that have balanced bidirectional communication, i.e. the absolute difference of sent and received messages is minimized. Sent and Receive criteria use the number of sent and received messages respectively.

The performance of authentication is measured by the number of peers that can be authenticated, and then, averaging over all the mailboxes. We find that the balanced selection rule has the best completion performance. This is because the underlying protocol requires bidirectional communication for progress. The performance also increases with smaller group sizes because fewer peers can delay authentication. Based on these observations, we select the trusted group size to be 10 peers, and use balanced selection criterion for populating bootstrapping groups.

Impact of message expiry

The authentication protocol operates as an overlay on the email infrastructure. As a lazy protocol it is susceptible to excessive log growth at the peers. We use an explicit message expiry time and carry it with all the protocol messages. This ensures that each protocol interaction has a finite life time. This limits the log size at each peer. We added this feature on observing that

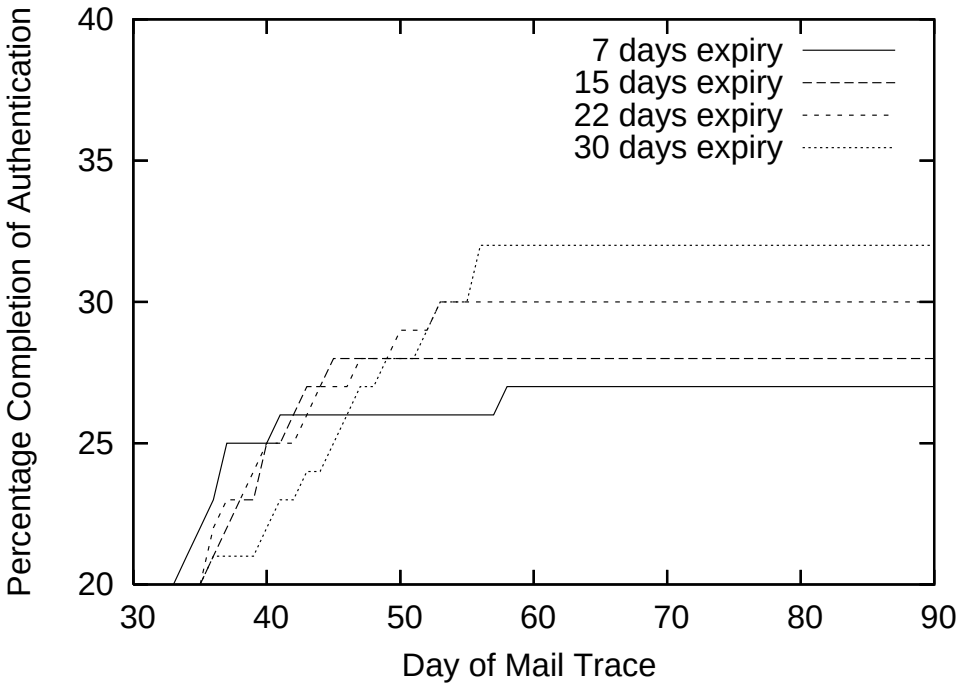

Figure 3.11: Variation of authentication performance with the expiry time of authentication protocol messages.

executing the trace becomes difficult without having a reasonable message expiry time. The accumulation of stale messages would severely impact the performance. We experimented with a number of practical values for message expiry as shown in Figure 3.11. The effect of message expiry on authentication performance was found to be marginal. Therefore, a moderate message expiry interval of 15 days was used in the experiments.

Message overhead

Authentication protocol messages are piggybacked on email through SMTP extension fields. Because SMTP implementations limit the mail header size, the number of protocol messages that can be attached to a single email message is limited. In order to understand the overhead introduced by the authentication overlay, we experiment with message payloads of 50 and 100 authentication protocol messages per email message.

The overhead on email messages due to piggybacking of compressed protocol messages is shown in Figure 3.12. Recall the payload constraint of 300 messages and the header size constraint of 10KB applied in Section 3.4.1. The observed overhead respects the constraint, as shown by the flat maximum message overhead observed for payloads of 50 and 100 messages. The median overhead and minimum overhead are shown for the payload value of 100. The overhead is positively biased because of a few idle peers. We observe that the experimental message payload values of 50 and 100 messages are reasonable for use with the 32KB header

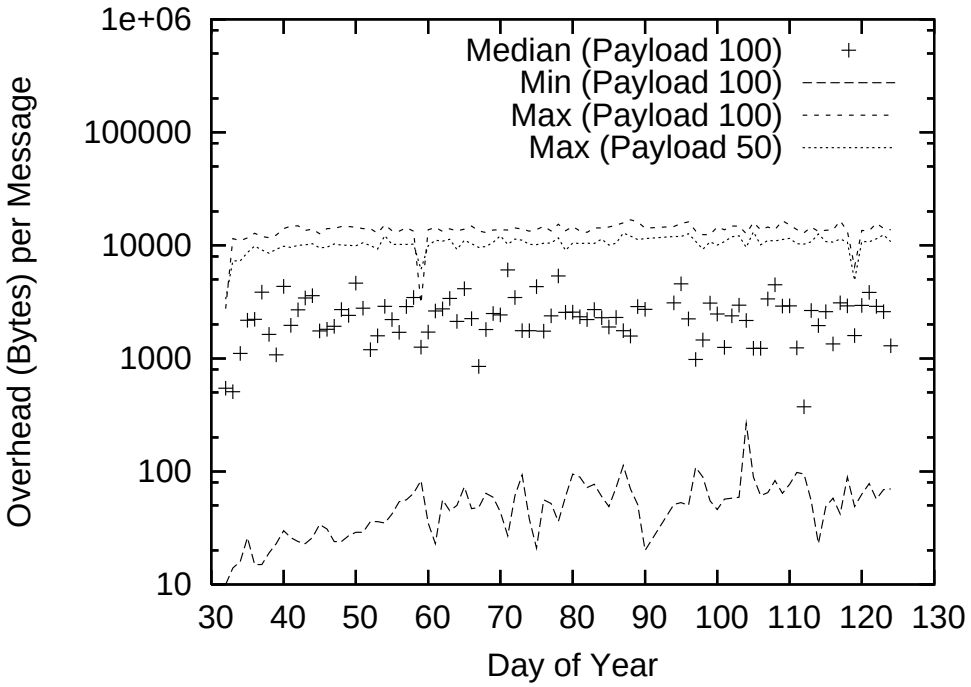

Figure 3.12: Overhead on email messages carrying the authentication protocol message payload.

size limit of SMTP.

Cache usage

Public key infection protocol relies on the lazy propagation of protocol messages. The messages that are not yet delivered need to be cached at participating peers. Using the message payloads of 50 and 100 protocol messages per email, we study the number of cached protocol messages as the authentication protocol progresses. The results are shown in Figure 3.13.

The number of cached protocol messages shows an increase as the protocol progresses. The distribution does not show a significant positive or negative bias as shown by the median being placed in the middle of minimum and maximum values. The initial trend shows an increase in number of cached messages as the protocol authenticates the bootstrapping peers. The median number of cached messages stabilizes as the rate of production and expiry of messages balances out. As shown in the figure, this happens approximately on the 50^{th} day of the trace.

We also note that the maximum permitted payload affects the number of cached messages. As shown in Figure 3.13, the maximum number of cached messages decreases marginally with decreasing payload. The number of messages is also closely related to the actual size of the cache as shown in Figure 3.14.

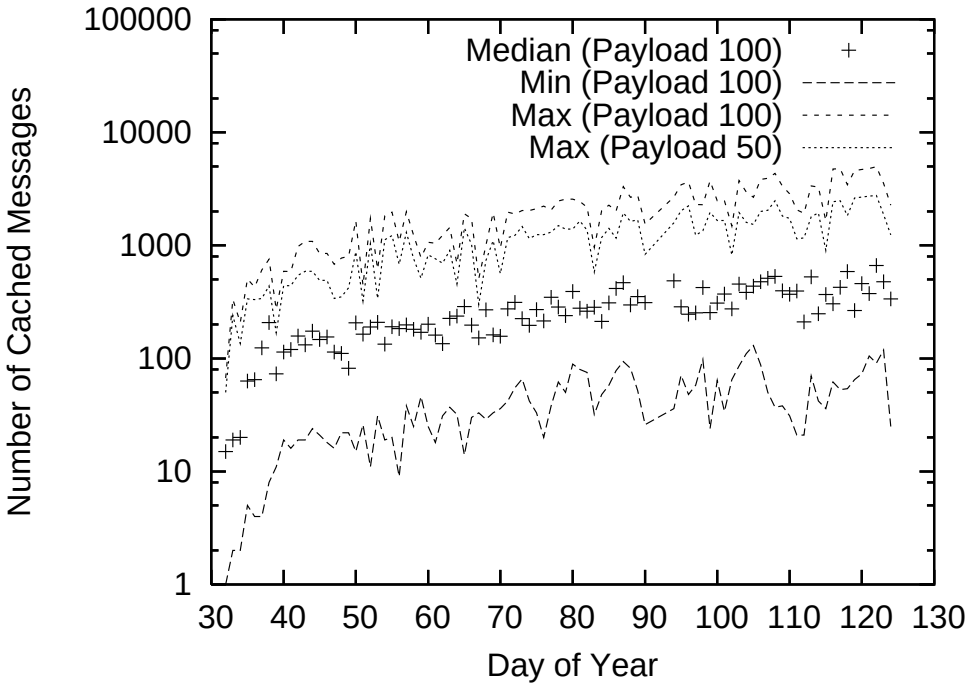

Figure 3.13: Number of authentication protocol messages cached at email clients.

Mail Checking Interval	Extra Email Ratio	Time for 80% authentication
Weekly	0.502	14 days
Daily	0.702	2 days
Hourly	7.038	2 hours

Table 3.2: Overhead and authentication latency for eager mode authentication.

Lazy mode authentication performance

The authentication protocol results in the authentication of public keys of peers. The progress of authentication is shown in Figure 3.15. It can be noted that there is a wide disparity between the progress of authentication between the best peer and the average performance of authentication. This gap can be attributed to the fact that most of the email users do not send a lot of messages. The implementation of authentication as an overlay on SMTP limits the rate of progress of authentication. Using a trusted group size limit of 10 peers, payload capacity of 100 messages, and a 15 day message expiry interval, the average peer can authenticate about 35% of its peers of interest in the 92 day run.

It is noteworthy that increasing payloads allow faster completion of the protocol. This is clear from the slower rate of authentication obtained with a payload of 50 messages as compared to 100 messages. This behavior is expected since the progress of the protocol is constrained by the payload limit, which restricts the immediate delivery of all possible messages.

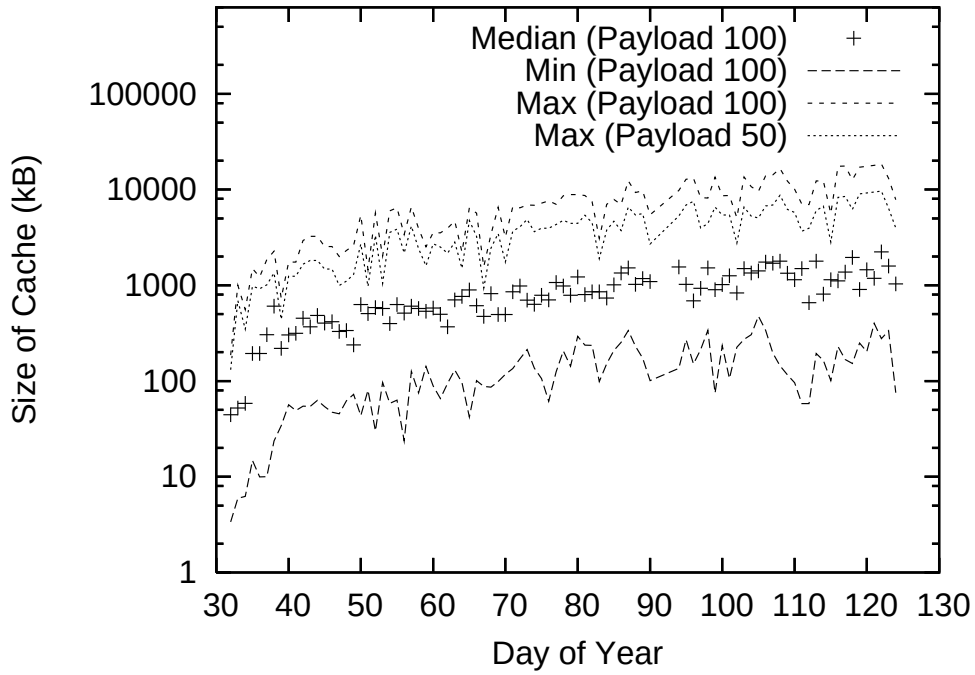

Figure 3.14: Storage overhead caused by the cached authentication protocol messages.

The authentication protocol requires challenge response results from all the trusted peers. However, even one challenge response result from a trusted peer provides some confidence in the authenticity of the public key being authenticated. This "optimistic authentication" is also studied as shown in Figure 3.15. The average completion of optimistic authentication is at 55% at the end of 92 days, i.e. averaging over all the peers, more than half of the peers have been authenticated. This progress is satisfactory considering that the protocol is backward compatible with the mail infrastructure, has lazy operation, and is fully autonomous.

Eager mode authentication performance

Eager mode authentication performance is evaluated for various eager sending intervals. This assumes that human users typically power up the email client to check for new received emails even if they do not send out any email. This periodic powering up of the email client is used for sending out the social-group key authentication messages to peers. This speeds up the authentication performance because users who only read emails can also be used for authentication.

We experimented with various periodic intervals for activating eager mode. As shown in Figure 3.16, the rate of authentication seen by all the peers increases as the periodic interval between eager exchanges is reduced. The eager protocol results in a substantial speedup in authentication performance as compared to lazy baseline authentication. The eager protocol can authenticate more than 90% of the peers within a week if users just check their emails once a day. This is a huge speedup over the slow rate of authentication seen in the lazy case. The

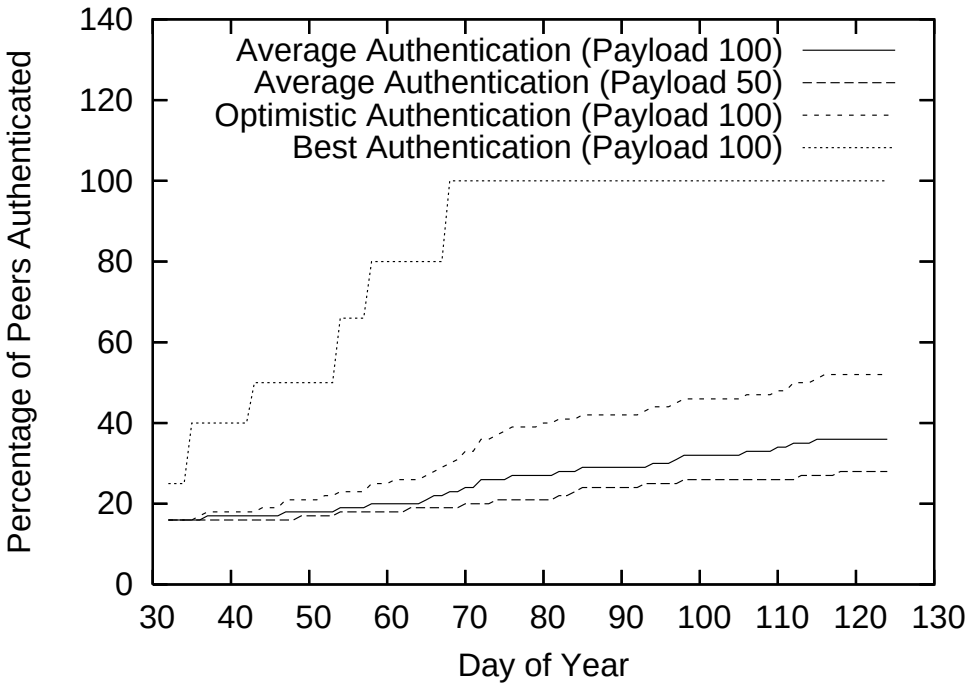

Figure 3.15: Authentication progress for different authentication message payloads.

overhead introduced by eager mode is measured in terms of ratio of additional email messages sent as compared to the organic email messages captured in the mail trace. The median of ratio overhead introduced by the eager mode is very marginal as shown in Figure 3.17. The time to reach 80% completion is about twice the eager send interval as also shown in Table 3.2. Thus, the eager mode latency can be used to trade off authentication delay for increased messaging overhead.

3.5.3 Industry workload

The second real trace is collected from the Internet mail gateway of a US corporation. This trace is collected ahead of the spam filter and poses a unique challenge for the authentication mechanism. As shown in Figure 3.18, while 70% of the addresses received more than 100 messages, less than 50% sent out more than 2 replies over the 56 day period. This is consistent with the large amount of incoming spam and can be contrasted with Figure 3.6, which shows the distribution on the spam filtered university trace.

The authentication protocol authenticates less than 2% of peers in the industry scenario. This can be contrasted with Figure 3.15, where the authentication protocol can authenticate a much larger percentage of peers. Analysis of the industry workload shows that senders outnumber the receivers by about 68 to 1. Therefore, most of the senders are not receivers. Since the authentication protocol is required to authenticate the public key of a sender, the few receivers can authenticate only some of the many senders. In order to interpret this result, we considered

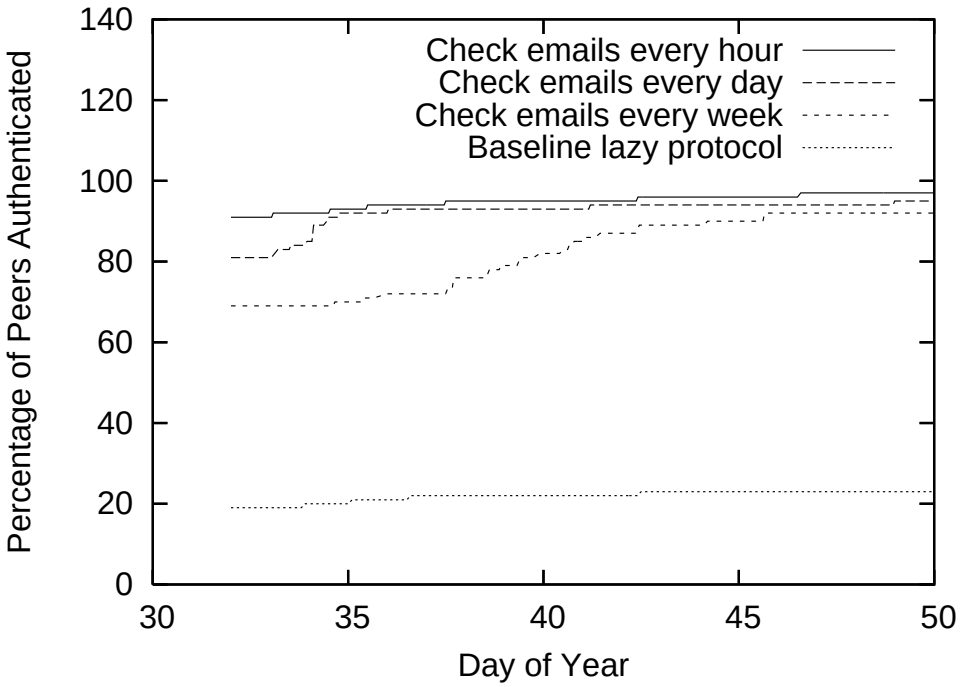

Figure 3.16: Progress of eager mode authentication with different email checking frequencies.

the instances where a receiver responds to the sender. The industry mail trace had 5 such instances. In two instances, the sender is authenticated by the receiver. We define *effectiveness of authentication* as the fraction of times a receiver can successfully authenticate the sender. We find that the effectiveness of authentication on the industry trace is 40%. In comparison, the university workload has 2301 such instances, and the effectiveness of authentication is 36%. Therefore, the performance of authentication on the industry trace is comparable to that on the university trace.

3.6 Comparison with alternative solutions

PGP is a popular public key authentication system. It works with the help of human decisions of trustworthiness that are expressed as public key certificates. PGP uses key rings to store trust relationships that are expressed as public key certifications. It can be noted that the unidirectional trust relationships in PGP can be mapped to a directed trust graph. The vertices of this trust graph are PGP keys and key certificates are the directed edges. Therefore, authentication of a node reduces to finding a path to the node in the trust graph. The observed behavior of PGP trust graph has been analyzed in [17, 85, 93]. The trust graph tends to have one very large strongly connected component. For example, biggest key-ring mentioned by `dtype.org` was 1.9GB in April 2002. We can compare the storage requirements by comparing this size with the cache size in our protocol. The storage costs are relatively small, of the order of 10MB, as shown in the experimental evaluation. Therefore, peer-to-peer public key authentication is

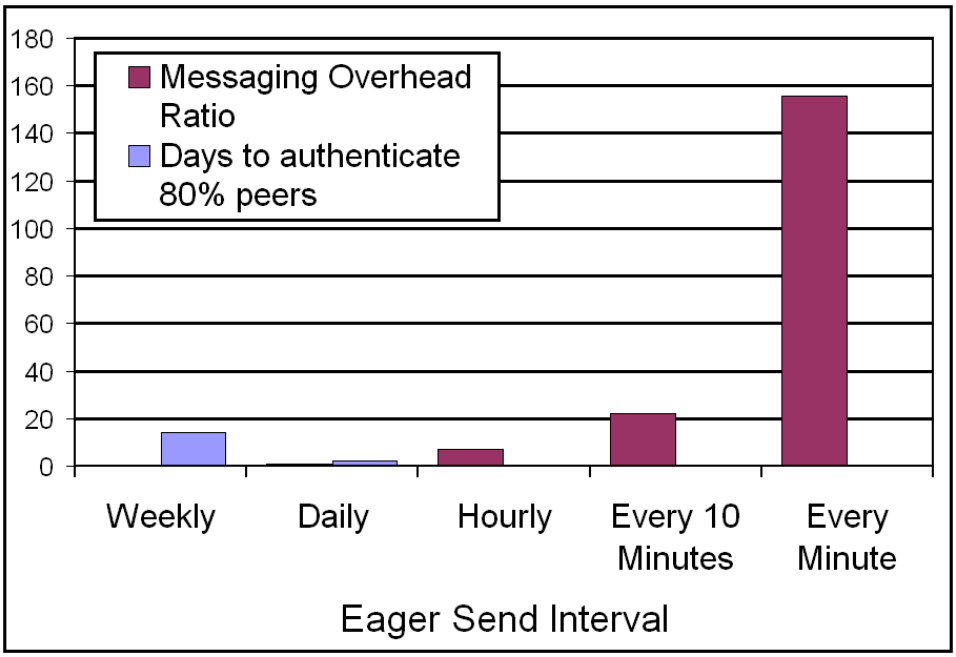

Figure 3.17: Messaging overhead and authentication latency for eager mode authentication.

autonomous and has lighter storage requirements as compared to PGP.

A number of sender domain authentication proposals have been put forward to tackle the spam problem. These include Sender Policy Framework [100], Sender ID [1], Domain Keys Internet Mail (DKIM) [4], and accredited DKIM (ADKIM) [36]. All of these proposals associate cryptographic material and mail sending policy with the DNS records of domains. This information is used by receivers to detect forged sender addresses. For example, a domain `xyz.com` could nominate a particular server to send all the emails for senders in the domain. The receiving mail transfer agent would check if this policy is being respected, and refuse to accept emails coming from senders in another domain, say `somebody@abc.com`. These proposed solutions are at the domain level, and are complementary to our user level solution. Our solution aims to achieve individual key authentication, which is at a finer granularity. Using the end-to-end argument [87], only the application that uses sender authentication is best equipped to correctly implement it. For example, users may want to distinguish senders on the same domain and be willing to receive email from `friend@abc.com` but not from `stranger@abc.com`. This kind of fine grained control may be complementary to domain level authentication. An additional benefit of our approach is that the computational cost of cryptographic processing is moved away from the mail gateway to the large number of user desktops.

The widespread use of spam control solutions with false positive errors has reduced the reliability of electronic mail. Garriss et. al. propose the use of social information inherent in the communication patterns to eliminate the false positives of spam filters [34]. However, this work makes stronger assumptions by prohibiting man-in-the-middle attacks and placing complete

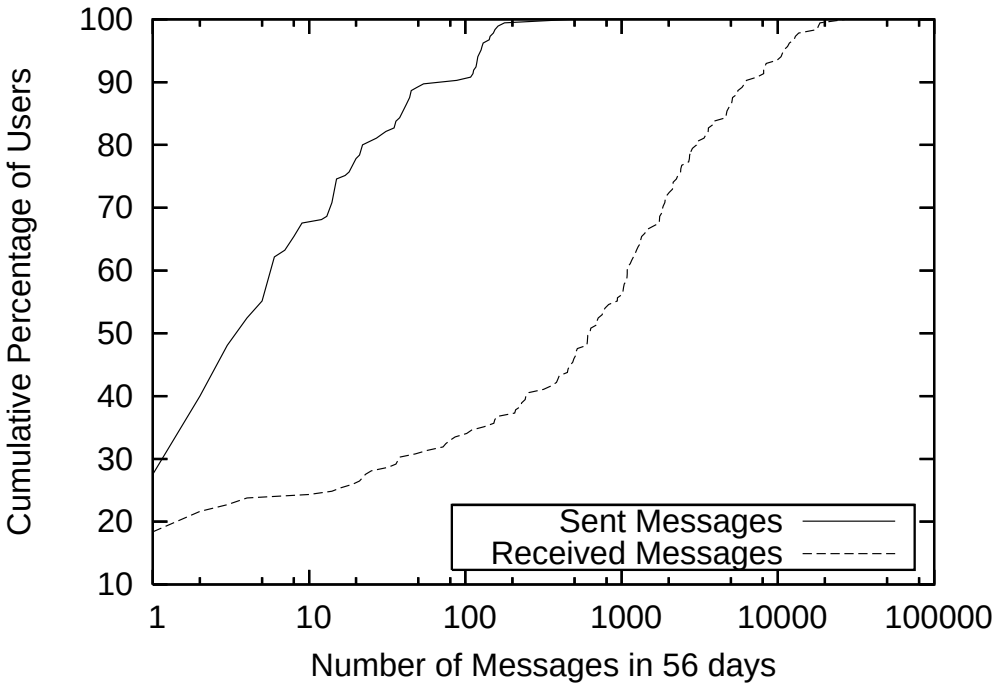

Figure 3.18: Activity profile of industry user accounts over the 56 day email trace period. The trace has 1.44 million email addresses and 2.5 million emails.

trust in the attestation servers that manage attestations of social relationship. Walfish et. al. propose another approach to solve the spam problem without introducing false positives [97]. This approach enforces a sending quota in a lightweight fashion but depends on global trust for quota allocators. Unlike these approaches, our sender authentication solution does not require global trust for any entity, resists man-in-the-middle attacks, and provides a useful sender authentication substrate that can be used to prevent false positives of spam filters. There are a number of commercial anti-spam solutions that use a challenge response mechanism to authenticate sender addresses. It can be noted that these solutions affect the usability of email by delaying the delivery of important messages. Our work differs from these solutions in two ways. First, while authentication could be delayed, message delivery is not affected. Second, while the challenge response step of these solutions is vulnerable to the man-in-the-middle attack, our solution resists such attacks.

3.7 Summary

We implement and evaluate social-group key authentication, an automatic, byzantine fault tolerant authentication system for email. We have implemented the authentication mechanism on the Thunderbird email client. Our authentication mechanism has been evaluated through micro-benchmarks, and with two real life email traces. Evaluation results show that the overheads are acceptable, and that the authentication mechanism is effective in real life scenarios.

The social-group key authentication protocol operates on social groups of collaborating users. Authentication in peer-to-peer systems has already been addressed through the byzantine fault tolerant public key authentication protocol developed in the previous chapter. In the following chapter, we will generalize to the ad-hoc networking model, and develop a protocol for authenticating public keys and geographic locations.

Chapter 4

Securing Geographic Routing in Mobile Ad-hoc Networks

4.1 Problem statement

Ad-hoc networks are becoming increasingly important with greater availability and popularity of networking enabled devices. These infrastructure-free networks, which are assembled on the fly, are vulnerable to malicious nodes and other adversaries. Both the routing mechanism and the routed data flow are potential targets. Geographic routing is an established protocol for routing in ad-hoc networks [14, 30, 50]. It relies on nodes knowing their geographic locations, and using their one hop neighbors for routing packets to target geographic destinations. Location awareness simplifies ad-hoc routing but also raises privacy concerns. Adversarial nodes may collaborate to track node locations, thereby violating location privacy. Securing geographic routing while simultaneously protecting location privacy is the focus of this chapter.

Securing ad-hoc routing is challenging because of the lack of pre-existing routing and security infrastructures. Nodes must create the routing infrastructure without using global knowledge. Lack of secure node identification is an additional challenge. Malicious or compromised nodes may pose as new nodes, or as known good nodes. Existing secure ad-hoc routing proposals have assumed various degrees of infrastructural support for addressing these challenges. In contrast, our approach uses anonymous nodes and a cryptographic protocol for securing ad-hoc geographic routing in an infrastructure free manner. Our solution trades off security infrastructure for computational and messaging overhead. Ongoing improvements in the capabilities of networked devices make this choice reasonable.

In this chapter, we present an infrastructure-free secure geographic routing protocol. Our protocol protects location privacy and requires that nodes be able to determine their own geographic location. Geographic routing protocols are known to be particularly susceptible to location errors and attacks [55, 59]. Our protocol authenticates geographic locations, thereby making it robust against malicious nodes. It also prevents malicious nodes from being used for routing. The routing paths taken by messages can be authenticated, which allows the development of new security policies, such as trusting message content if the message is routed through designated safe areas. In contrast to existing location authentication research, our approach does not require out-of-band communication or shared secret initialization.

4.1.1 Our solution

We propose geographical secure path routing (GSPR) for securing ad-hoc routing against malicious nodes and passive adversaries. The routing protocol operates on location aware anonymous nodes in order to provide privacy preserving secure geographic routing for ad-hoc networks. The protocol has the following goals:

- Route messages to desired geographic locations in the presence of malicious nodes. Detect and avoid bad geographic regions containing malicious or faulty nodes.
- Authenticate the self-generated public keys and geographic locations of nodes on the routing path.

This chapter describes the GSPR protocol in detail. Attacks from malicious nodes and passive adversaries are analyzed in order to demonstrate the correctness and attack resistance of the protocol. The overhead introduced by the protocol is modeled analytically and investigated in various operational and attack scenarios using the NS2 network simulator [72].

4.2 State of the art

Secure routing in ad-hoc networks has been investigated by a number of prior works. Hu and Perrig propose the Ariadne protocol for securing on-demand and source routing protocols in ad-hoc networks [43]. Similarly, the SEAD protocol [42] secures distance vector routing in ad-hoc networks. Both the protocols requires a secure cryptographic initialization phase, but use highly efficient symmetric key cryptography. While our approach uses expensive modular arithmetic, it does not require secure initialization. A public key infrastructure based approach to securing distance vector routing is presented in [98, 105]. Public key infrastructure is also used in [9] to secure on-demand ad-hoc routing protocols. Threshold cryptography based secure ad-hoc routing is presented in [106]. All of these approaches are infrastructure oriented. Our secure routing protocol, in contrast, is infrastructure free.

The privacy threat posed by location aware devices has also been a topic of intense research. Defective or compromised devices could allow tracking of their users. A coalition of malicious nodes could co-operate to continuously track the geographic locations of correct devices (and their users). Existing research has taken two approaches for protecting user privacy: the first is to fudge the locations of identifiable nodes as in [66, 37]. The second is to use pseudonyms for temporary identification of nodes as proposed in [11, 101, 81, 108, 18]. We use the latter approach because secure geographic routing requires authenticated locations. Using temporary pseudonyms for location aware nodes allows us to provide location privacy and location authentication simultaneously.

A multi-hop anonymous challenge mechanism has been used by Mahajan et. al. [67] for

detection of free riders in ad-hoc wireless network. Their mechanism requires two-hop transmission of challenge messages. Our recursive challenge response is related to their mechanism in the sense that remote nodes are used for security processing. Unlike in our work, the authors do not study the effects of mobility.

A secure cryptographic initialization based scheme for providing robust location estimation in a hostile environment is presented by Liu et. al. [65]. Our approach is distinguished in the sense that we do not require any secure initialization. However, our approach is more expensive, requiring asymmetric cryptography.

A statistical approach for robust localization using triangulation is described by Li et. al. [64]. Our approach is similar in the sense we allow the existence of malicious nodes. The difference in the approach is that we use cryptographic primitives at a higher level, and do not depend on physical level data. Other approaches include Sastry et. al. [91], who use echo latency for localization. In contrast, our solution does not use out of the band communication.

4.3 Preliminaries

Geographic routing is a well researched approach for ad-hoc routing [14, 30, 50, 53]. Nodes are expected to know their own geographic locations, and to share it with their one-hop neighbors through periodic beacons. The periodic beacons allow each node to know the geographic locations of its one-hop neighbors.

Messages carry their target locations as they are routed though the ad-hoc network. Under the assumption of bidirectional connectivity, geographic routing can be efficiently implemented on a planar sub-graph of the one-hop connectivity graph. A number of planarization approaches have been developed, and each can lead to a different subset of neighbors to use for routing. Once the choice of neighbors to use for routing is made, the next hop can be determined by greedily selecting the next hop in order to minimize the remaining distance to the destination location. Greedy forwarding fails if there is no next hop among the neighbors that is closer to the destination. In this case, geographic routing makes progress by entering the perimeter mode. In the perimeter mode, the next-hop is selected so as to traverse the perimeter of the region where greedy forwarding failed. Perimeter mode forwarding continues as long as there is no better greedy next hop neighbor, and the initial location where the perimeter mode started is not visited again. Geographic routing is simple and efficient. The state required at each node depends only on the node density.

The underlying assumptions of bidirectional and distance based connectivity have been challenged through experimental studies in [53]. The same study also notes that these non-malicious errors are solvable through improved planarization techniques. Therefore, we do not consider violations of bidirectional connectivity and transmission range. Prior research has also noted that geographic routing is susceptible to location errors and attacks [55, 59]. These

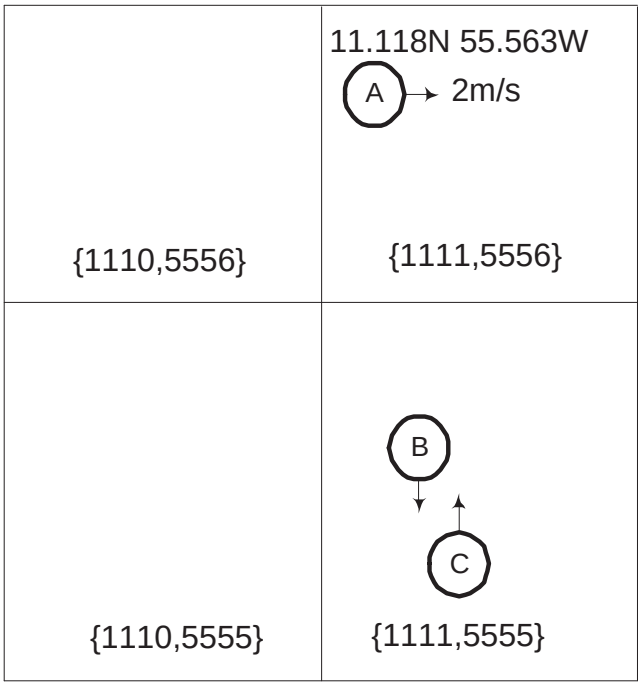

Figure 4.1: An example of geographic location and integer co-ordinates. Co-ordinates are shown in curly brackets, e.g., $\{1110, 5556\}$. 11.118N, 55.563W is a geographic location.

security attacks are the basic focus of our geographical secure path routing protocol (GSPR).

4.4 Model

This section discusses the assumptions, the problems being solved, and the notation used in this chapter. The participants in our geographical secure path routing protocol are referred to as nodes. The nodes are located in an integer vector space as shown in Figure 4.1. The nodes may be stationary or mobile. These two cases are considered separately. The correctness and attack resilience of routing is first established for the stationary case. The effects of mobility on correctness, attack resilience, privacy, and performance are studied thereafter.

4.4.1 Assumptions

We make a number of standard ad-hoc networking assumptions. All the nodes are assumed to know their geographic locations. The GSPR protocol operates on integer co-ordinates, which can be constructed by scaling the geographic co-ordinates with a global constant. For example, as shown in Figure 4.1, the integer co-ordinates $\{1111, 5556\}$ of node *A* can be derived from the geographic location 11.118N, 55.563W by applying a scaling factor of 100. The scaling

factor is assumed to be a predefined constant well known to all the nodes. Nodes generate their own public-private key pairs. We also make standard assumptions about the non-invertibility of popular cryptographic functions.

Nodes are identified through short-lived temporary pseudonyms. These pseudonyms are also used as the physical level node identifiers in order to prevent physical level identification attacks. Pseudonyms are constructed from a pseudo-random number generator. This ensures that pseudonyms can not be used to derive real node identity. Nodes are assumed to silence transmissions while changing pseudonyms. Nodes can also estimate the number of one-hop neighbors either by listening to local transmissions or by knowing the node density. These capabilities are sufficient for using mix zones. Mix zones are regions with sufficient node density that can provide large enough anonymity sets to support the desired level of anonymity [18]. Changing pseudonyms within mix zones prevents node tracking by making it impossible to associate a series of pseudonyms with a physical node. This protects the location privacy of participating nodes.

We assume that promiscuous mode reception is enabled on the networking adapters. Nodes overhear all transmissions in their one-hop neighborhood. This enables detection of malicious activity in the one-hop neighborhood. The underlying networking layer is expected to handle the hidden terminal problem, packet collision, and asymmetrical connectivity issues. Node connectivity is assumed to be symmetrical and resilient to packet losses caused by collisions or jamming.

Node connectivity is assumed to have a range limited one-hop neighborhood. The one-hop neighborhood is bounded by a maximum distance R. Nodes farther apart than R are never one-hop neighbors. The maximum range distance is assumed to be a fixed global constant, which depends on the the link layer technology being used. For example, in an outdoor IEEE 802.11 peer-to-peer wireless network, the maximum range is limited to a few hundred meters. Violations of connectivity radius are detected, and result in the offending node being eliminated from the secure routing protocol. Limited connectivity radius is a reasonable assumption in the context of standardized wireless networking hardware.

4.4.2 Definitions

Consistent with traditional geographic routing [50], all the nodes are located on a plane. Every node p has a geographic location: $\text{Location}(p) \equiv (\mathbf{p_x}, \mathbf{p_y})$. The integer co-ordinates p_x and p_y of the node are computed by scaling the geographic location with the global scaling factor. Each node p has a public key K_p, such that a message m can be secretly sent to p by encrypting with the well known public key as $K_p(m)$. The corresponding private key K_p^{-1} is only known to p, and can not be computed from K_p. All the messages transmitted from p are digitally signed with the private key K_p^{-1}. The notation used in this chapter is summarized in Table 4.1.

$f \circ g(x)$	Function composition of f and g, i.e. $f(g(x))$.
$f^{(k)}(x)$	The function f applied k times on x.
$\{x, y, z\}$	Message containing strings x, y and z.
K_p	Public key of the node p.
K_p^{-1}	Private key of the node p.
$K_p(x)$	A string x encrypted with the public key K_p of p.
$\{x\}_p$	A message consisting of string x signed by node p.
r_p	A pseudo-random number generated by node p.
$(\mathbf{p_x}, \mathbf{p_y})$	The geographic location of node p, also represented as Location(p).
(p_x, p_y)	Integer co-ordinates of node p derived by scaling its geographic location.
$\mathcal{N}(p)$	The set of nodes in the one-hop neighborhood of node p.

Table 4.1: Notation

Our GSPR protocol resists attacks from malicious and faulty nodes. Malicious nodes may intentionally try to give incorrect responses while faulty nodes may be attacked and have incorrect inputs to offer. Honest nodes are distinguished from malicious or faulty nodes as follows:

DEFINITION 6 (Honest Node) *An honest node knows its correct geographical location, follows the maximum range constraint, and executes the GSPR routing protocol correctly. Otherwise, the node is called malicious or faulty.*

The ad-hoc network consists of honest and malicious nodes. These nodes may be placed at arbitrary geographic locations. Nodes become candidates for geographic routing depending on their geographic location. Routing paths consist of sequences of nodes. Each node on the routing path is responsible for forwarding the message towards the geographic destination. Given a node responsible for forwarding the message at a given hop, we define its honest witness node as follows:

DEFINITION 7 (Honest Witness) *An honest node is an honest witness for a forwarding node if it is in the one-hop neighborhood of the forwarding node, the previous-hop node, and the next-hop node for the given geographic routing path.*

The presence of honest witnesses allows the protocol to secure geographic routing while forwarding messages. We define an *honest witness network* as an ad-hoc network that has honest witnesses for all of its nodes.

DEFINITION 8 (Honest Witness Network) *A network is an honest witness network if there is an honest witness for every possible routing path of length* 3*.*

Geographic routing allows packets to be routed to destination locations. The routing protocol will return a failure message if there are no nodes in the one-hop neighborhood of the target

location. If the one-hop neighborhood of the target location has one or more nodes, then each of them is considered a valid destination. Our GSPR protocol operates only on honest nodes. The routed packets are expected to reach target locations by using secure routing paths only.

DEFINITION 9 (Secure Routing Path) *A secure routing path consists of a sequence of honest nodes, each of which is the one-hop neighbor of its predecessor.*

Participation in our protocol allows nodes to find a secure routing path to the destination. The source node requests for a node located near the desired geographic location. GSPR finds an honest node in the one-hop neighborhood of the location if possible. The returned response contains the public key of the discovered node and the secure routing path to it. Message integrity is guaranteed for both query and response messages. Next-hop nodes for the query messages are determined by geographic routing while the response message is source routed on the reverse path. The security properties of GSPR protocol are listed below:

DEFINITION 10 (Properties of GSPR protocol) *Given a message $\mathfrak{M}$ starting at node s with the destination geographic location D, a geographic secure path routing (GSPR) protocol is secure if the following properties hold in an honest witness network:*

- *If there is a secure routing path $\mathcal{S}(s,d)$ from the source node s to a node d located within one-hop distance of the destination location $\mathcal{D}$, then $\mathfrak{M}$ is routed to d.*
- *The destination node d receives the secure routing path $\mathcal{S}(s,d)$.*
- *On receiving the returned response, the source node s gets the correct public key K_d of the destination, and the secure routing path $\mathcal{S}(s,d)$ traversed by the message.*

4.4.3 Threat model

The routing protocol may be attacked for a number of reasons. Adversaries may create the false appearance of being at a location in order to gain additional privileges. For example, access to a classified document may be denied while away from a safe area, but approved within it. Adversaries may also have the simple motivation of rendering the routing mechanism unusable. These motivations are considered in this section. A summary of the threat model is also provided in Table 4.2.

A number of threats are expected in the wireless ad-hoc networking environment. Both the routing mechanism and the routed data may be attacked. Attacks can be mounted on the routing mechanism by dropping or incorrectly forwarding messages. Incorrect forwarding means forwarding messages towards incorrect directions, and includes classical attacks like routing loops and wormholes. These attacks against geographic routing are considered in the security analysis. Attacks on routed data could target the payload or the control data required

Category	Specific Threat
Routing attack	Dropping messages
	Routing in the wrong direction
Data attack	Payload modification
	Control data modification
Malicious node	Reporting false location
	Directional transmission
	Transmission power changes
	Tracking node location

Table 4.2: Overview of the threats handled by GSPR

for protocol operation. Control data susceptible to modification includes node identifiers, node locations, and other data fields governing the routing protocol. The threat of data modification is also considered in our analysis.

Nodes are also vulnerable to attack in ad-hoc networking environment. Compromised nodes can be controlled by an attacker causing them to behave maliciously or incorrectly as defined in Definition 6. Malicious nodes may also collude to continuously track the location of a node, thereby violating its location privacy. In case of nodes having home locations, location privacy violations also make anonymity violations more likely. Threats from compromised or malicious nodes are considered in our analysis along with possible attacks on location privacy and anonymity of nodes.

A number of low level attacks are possible against wireless ad-hoc networks. The jamming attack blocks radio transmissions in a given geographic region, thereby preventing the routing protocol from using that area. Jamming can be tackled with spread spectrum techniques [102]. Other low level attacks include transmission power changes and directional transmissions. These attacks are detected by the protocol and the responsible nodes classified as malicious.

4.5 Geographical secure path routing

The details of our geographical secure path routing protocol are described in this section. Each of the building blocks: geographic hashes, periodic beacons, geographic routing, and malicious node detection are presented below.

4.5.1 Geographic hashes

We develop a novel method, called *geographic hashes*, for encoding relative geographic positions of two nodes. Our solution associates unforgeable transient geographic hashes to relative

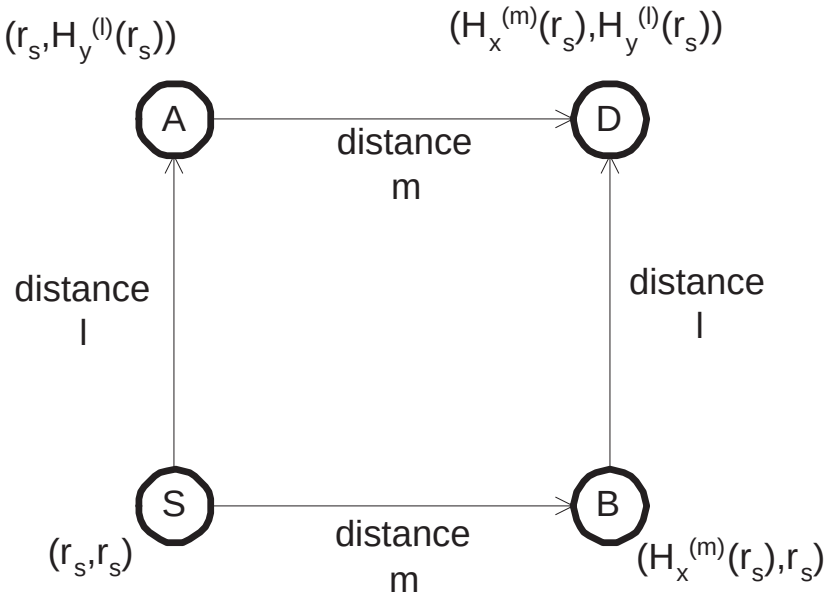

Figure 4.2: Using associative one-way functions to create geographic hashes. H_x and H_y are associative one-way functions which are applied to maintain geographic hash values at different geographical locations. r_s is a random nonce published by node s and serves as the geographic hash of s at s.

geographic positions as shown in Figure 4.2. Geographic hash values are a function of source node and location of calculation. Geographic hashes allows the validation of node locations beyond the one-hop neighborhood. Nodes maintain a set of integer tokens, called geographic hashes, that associate a secret with a geographic location. The secret can not be determined without being in the vicinity of the location, but its knowledge can be verified remotely. Geographic hash values are transient (i.e., are short-lived and keep changing while a node moves). They are computed by repeatedly applying cryptographic one-way functions on a random nonce chosen by the node.

Geographic hash contains a chain of hashed values. It has the following interesting property. Given a geographic hash value corresponding to a time period i, it is infeasible for any polynomial-time adversary to compute the geographic hash values corresponding to any subsequent time period $j > i$. On the other hand, the reverse computation can be easily done, that is, given a geographic hash value corresponding to a time period i, it is easy to compute the geographic hash values corresponding to the preceding time period $j < i$. Conceptually, this property is the opposite of a regular one-way hash function, where computing the hash value from an input is easy while finding the input corresponding to a hash value is hard. We are able to achieve our property by having a node to (1) pre-compute a chain of hash values $(r[1],\ldots,r[n-1],r[n])$, where $r[n]$ is the hash value of $r[n-1]$ and so on, and (2) use the chain

in reverse order $(r[n],\ldots,r[1])$, that is, use $r[n]$ first, then $r[n-1]$ second, etc. Geographic hashes are created through modular arithmetic. Consider a large prime p and a generating number a, such that the function $f(x) \equiv a^x \bmod p$ maps $\mathbb{Z}_p^\star = \{1,\ldots,p-1\}$ onto itself. Each integer $h \in \mathbb{Z}_p^\star$ can be used to represent a one-way function $H(x) \equiv (a^h)^x \bmod p$ since the discrete logarithm problem of finding x, given $y = a^x \bmod p$, is believed to be NP-hard. Formally, geographic hash is defined next.

DEFINITION 11 (Geographic Hash) *Each node A periodically publishes the following geographic hash parameters: a large prime p, a generator a for $\mathbb{Z}_p^\star = \{1,\ldots,p-1\}$, three random integers $\zeta_A, \eta_A, \theta_A \in \mathbb{Z}_p^\star$, and a time interval Δ_A indicating the expiry time for a single version of the geographic hash.*

The geographic hash of A is initialized to (r_A, r_A) at A, where r_A is a random nonce selected by A. Successive versions $r_A[i]$ and $r_A[i+1]$ of the random nonce satisfy:

$$r_A[i+1]a^{\theta_A} \bmod p = r_A[i]$$

Each node B in the neighborhood of A computes the geographic hash of A at B as follows:

$$GH(A,B) = \left(r_A a^{\zeta_A \Delta x} \bmod p, r_A a^{\eta_A \Delta y} \bmod p\right)$$

where Δx and Δy are the differences in the integer co-ordinates related to the geographic location of A and B.

The geographic hash has the following properties:

- *A geographic hash at a remote location is easy to verify.*
- *A geographic hash for a remote location is hard to construct except by being in the one-hop neighborhood of the location.*

The geographic hash is a tuple of integers computed by repeated applications of one-way functions to the locally known geographic hash of a node. The functions are designed to encode the relative position across multiple applications on various nodes. Consider a component GH_y of the geographic hash after two translations Δy_1 and Δy_2 from A:

$$\begin{aligned} GH_y &= r_A.a^{\eta_A \Delta y_1}.a^{\eta_A \Delta y_2} \bmod p \\ &= r_A.a^{\eta_A \Delta y_1 + \eta_A \Delta y_2} \bmod p \\ &= r_A.a^{\eta_A (\Delta y_1 + \Delta y_2)} \bmod p \end{aligned}$$

The final value of the geographic hash component depends only on the initial random nonce r_A and the difference in integer co-ordinates. An example computation is shown in Figure 4.2. The y component of a geographic hash is being computed for the parameters $r_A = 53$, $\eta_A = 10$,

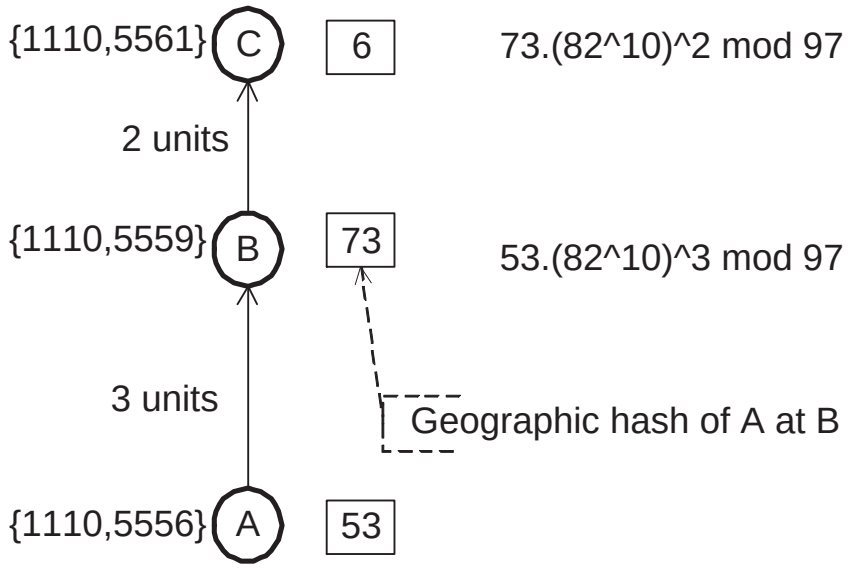

Figure 4.3: Example of geographic hash calculation. The node A selects $r_A = 53$ as its initial random nonce, 82 and 97 are the generating and prime numbers, and $\eta_A = 10$. The calculated y component of the geographic hash is shown in square boxes. An explanation of the value is given on the right.

prime number $p = 97$, and generating number $a = 82$. The final value of 6 encodes a relative translation of 5 units along the y axis because $6 = 53(82^{10})^5 \bmod 97$. Since the construction of geographic hashes encodes the relative geographic locations of nodes, the computed geographic hash values would be identical across different calculation paths.

4.5.2 Beacon

Nodes are required to know one-hop neighbor locations for geographic routing. This is achieved by having the nodes transmit a periodic beacon containing the node identifier and location. Nodes continuously listen for beacons from neighboring nodes. The information gathered from the received beacons is stored in memory in order to support geographic routing. Our protocol extends the beacon to include the public key and the random nonce selected by the node. This ensures that public keys of nodes are well known in the one-hop neighborhood. The beacon also includes locations and geographic hashes of neighboring nodes. This information helps in detecting malicious nodes as discussed in Section 4.6. Beacon messages are digitally signed with the private key of the node and are broadcast to all the one-hop neighbors.

DEFINITION 12 (Periodic Beacon) *Each node p periodically broadcasts a beacon message containing its geographic location, random nonce r_p, public key K_p, neighbor information list $\mathbb{Q}$, and a set of geographic hashes $\mathbb{G}$ known to it. The beacon message is broadcast to all nodes*

in the one-hop neighborhood $\mathcal{N}(p)$ of p as shown below:

$$p \quad \rightarrow \quad \mathcal{N}(p) \qquad \left\{ \{p, Location(p)\}_p, r_p, K_p, \mathbb{Q}, \mathbb{G}, \mathbb{M} \right\}_p$$

where

$$\mathbb{Q} \equiv \left\{ \{q, Location(q)\}_q \mid q \in \mathcal{N}(p) \right\}$$

and

$$\mathbb{G} \equiv \left\{ \left\langle q, \zeta_q, \eta_q, \theta_q, \Delta_q, GH(q,p) \right\rangle \mid Distance(q,p) < 2R \right\}$$

and

$$\mathbb{M} \equiv \{ \langle q,\ Evidence \rangle \mid q \text{ is malicious} \}$$

The periodic beacon permits sharing the information that is used for validating routing actions. Sharing the neighbor information list $\mathbb{Q}$ helps in detecting false location attacks among the set of one-hop neighbors. The geographic hashes of nodes located within twice the maximum one-hop radius *R* are stored in memory. These geographic hashes are shared with neighboring nodes in order to detect malicious routing behavior beyond the one-hop neighbors.

4.5.3 Routing protocol

The GSPR protocol has a two step query-response messaging model. Payload and control data are sent towards the destination location as is done in traditional geographic routing protocols. The returning source routed acknowledgment completes the protocol. The routing protocol consists of the following operations: BEGIN FORWARDING, GEOGRAPHIC FORWARDING, LOCAL RESPONSE, RECURSIVE RESPONSE, END FORWARDING, REVERSE RESPONSE FORWARDING, and VERIFICATION. The additional stage of malicious node detection is treated separately in the next section because it encompasses all the operations described above. Malicious node detection results in nodes being detected as malicious or the neighborhood being classified as a bad neighborhood. Malicious nodes are not used for routing and honest nodes do not forward messages if they are located in bad neighborhoods.

Consider a data payload being routed from source node the *s* to a destination node *d* near the target geographic location. Let $\{p_0, \ldots, p_k\}$ be the intermediate nodes on the geographic routing path. The messages involved in the GSPR protocol are described below. All the messages include source and destination identifiers. These common fields are omitted below for brevity.

- BEGIN FORWARDING
  This operation executes at the source node *s* and begins routing the payload to the destination geographic location $\mathcal{D}$. The message contains a location list, $\mathbb{L}_s = [\text{Location}(s)]$,

which is initialized to contain the source location. The message also contains the random nonce r_s selected by the source node. The message type "forward" is included in order to distinguish it from other messages in the protocol. The following message is sent to p_0, the next-hop node for geographic routing:

$$s \quad \rightarrow \quad p_0 \qquad \{\text{Forward}, \mathcal{D}, r_s, \mathbb{L}_s, \text{Payload}\}_s$$

- GEOGRAPHIC FORWARDING

  This operation executes at each node on the routing path. At node $p_i \in \{p_0, \ldots, p_k\}$, the next-hop p_{i+1} is determined according to geographic routing towards the destination location $\mathcal{D}$. The operation is triggered on receiving a message with the type "forward" from the previous hop p_{i-1}. The operation results in further forwarding of the message if the current node is not the destination node. The forwarded message contains the location list $\mathbb{L}_i$ which is constructed by appending the location of the current node to the location list received from the previous node p_{i-1}, i.e. $\mathbb{L}_i = \mathbb{L}_{i-1} + [\text{Location}(p_i)]$. The node p_i transmits the following message to the next hop p_{i+1}.

$$p_i \quad \rightarrow \quad p_{i+1} \qquad \{\text{Forward}, \mathcal{D}, r_s, \mathbb{L}_i, \text{Payload}\}_i$$

  The operation also causes a "local response" message to be returned to the previous hop node p_{i-1}. The local response contains public key, node identifier, and geographic hash (see Definition 11) information $\mathcal{I}(p_{i+1})$ about the next-hop node:

$$\mathcal{I}(p_{i+1}) \equiv \left\langle K_{p_{i+1}}, p_{i+1}, \text{Location}(p_{i+1}), \text{GH}(p_{i+1}, p) \right\rangle$$

$$p_{i-1} \quad \leftarrow \quad p_i \qquad \{\text{Local Response}, r_s, K_{p_i}^{-1}(r_s), \mathcal{I}(p_{i+1})\}_i$$

  The purpose of the local response message is to help verification of the protocol operation beyond the one-hop region. It permits the previous hop p_i to check the validity of the public key $K_{p_{i+1}}$. The presence of the geographic hash of the next hop node permits the current node to verify that the next to next-hop location is genuine and correct.

- END FORWARDING

  This operation executes at the destination node d. Repeated forwarding of "forward" messages routes the payload to the destination. The destination node does not forward a message of type "forward" but runs the receive operation instead. On verifying the integrity of the received message, the destination sends back the "recursive response" to the source. The recursive response contains the location list $\mathbb{L}_d$ traversed by the geographically routed message. This list is used to route the recursive response back to the source. The recursive response also contains a list of public keys of the nodes on the

reverse routing path, $\mathbb{P}_d = [K_d]$, which is initialized to contain the public key of the destination. The challenge nonce r_s and its response $q_d = K_d^{-1}(r_s)$ are also sent back towards the source.

$$p_{k-1} \quad \leftarrow \quad d \qquad \{\text{Recursive Response}, \mathbb{L}_d, r_s, q_d, \mathbb{P}_d\}_d$$

- REVERSE RESPONSE FORWARDING
  This operation executes at all the nodes on the routing path. The operation is triggered on the node p_i on receiving a "recursive response" message from the node p_{i+1}. The public key of the current node is appended to the public key list received in the incoming message: $\mathbb{P}_i = \mathbb{P}_{i+1} + [K_{p_i}]$. The location list $\mathbb{L}_d$ is checked to ensure the current node is on the reverse route. If so, the following message is transmitted to the next node p_{i-1} on the reverse route:

$$p_{i-1} \quad \leftarrow \quad p_i \qquad \{\text{Recursive Response}, \mathbb{L}_d, r_s, K_{p_i}^{-1}(q_{i+1}), \mathbb{P}_i\}_i$$

- VERIFICATION
  This operation executes on the source node after it receives the recursive response message routed back to it. On receiving the recursive response, the source gets a signed copy of the routing path $\mathbb{L}_d$ taken by its original forward message and the list of public keys $\mathbb{P}_0$ belonging to the nodes on the routing path. The source also gets the response $q_0 = K_{p_0}^{-1} \circ K_{p_1}^{-1} \circ \ldots \circ K_d^{-1}(r_s)$ to its challenge nonce r_s. The correctness can be verified using the public keys in $\mathbb{P}_0$ as:

$$r_s = K_{p_0} \circ K_{p_1} \circ \ldots \circ K_d(q_0)$$

  This verification completes the protocol.

4.5.4 Malicious node detection

Malicious node detection is based on the broadcast nature of wireless communication and is modeled after the watchdog protocol [68]. Nodes listen to the transmissions of their neighbors in order to detect malicious nodes. Malicious nodes are not used for routing. Since the protocol operates in an honest witness network (see Definition 8), an honest witness node is always able to detect malicious activity and warn other neighboring nodes through its periodic beacon. Watchdog protocols are vulnerable to blacklisting attacks. Because the GSPR protocol uses temporary pseudonyms, a blacklisting attack can only have temporary effect. Avoiding temporary blacklisting is not a goal of the protocol, and is not considered further. Malicious nodes are detected both by checking for inconsistencies in the periodic beacons and by checking the correctness of geographically routed messages and their reverse source routed responses. Among the threats identified in Section 4.4.3, false location attacks are

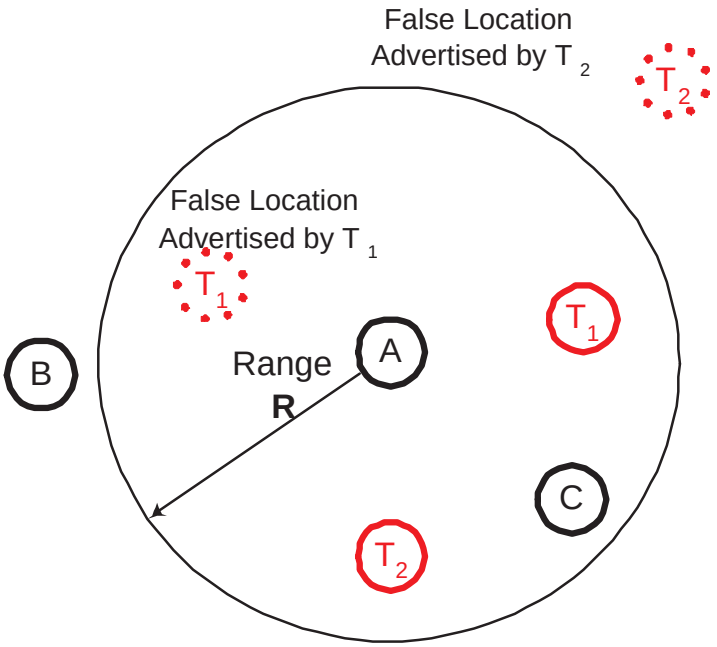

Figure 4.4: Different types of false location attacks

detected through inconsistencies in the periodic beacons. The remaining threats are handled by overhearing query and response messages in promiscuous mode. This permits detection of malicious forwarding.

Beacon validation

Periodic beacons are received from neighboring nodes. According to Definition 12, beacon messages contain location information about one-hop neighbors and the geographic hashes of nodes within two-hops of the sending node. Beacons are stored in memory and their contents used for detecting malicious nodes. Nodes launching false location attacks are detected by applying the range constraint R. Each node constructs a number of mappings from pseudonym to location, one received from each neighbor. Small inconsistencies in location are ignored as location errors. Larger inconsistencies permit the node to conclude that its one-hop neighborhood has malicious nodes. The precise threshold of location error limit depends on the GPS error rate. As shown in Figure 4.4, the node A can detect that the advertised location of T_2 is out of range. This node must be malicious and can be ignored for routing. A related situation occurs for T_1, which appears to be in the direction of B when it is not. The node A detects this malicious behavior with help of the node C. The beacon from C will contain either a different or an infeasible location for T_1. Thus, A can determine that its one-hop neighborhood has malicious nodes.

The detailed procedure for detecting false location attacks based on received beacons is given in the algorithm below:

```
1  malicious ← [] ;
2  neighbor_info ← [] ;
3  bad_neighborhood ← False;
4  while beacon ← ReceiveBeacon() do
5      q ← Sender( beacon ) ;
6      if Distance(q, p) > R then
7          malicious ← malicious + {q} ;
8      end
9      for r ∈ Neighbors(q) do
10         if Distance(q, r) > R then
11             malicious ← malicious + {r} ;
12         end
13         else if Distance(p, r) < R and r ∉ neighbor_info then
14             malicious ← malicious + {r} ;
15         end
16         if not ( neighbor_info [r] ≃ Location(r,beacon) ) then
17             bad_neighborhood ← True ;
18         end
19         else
20             neighbor_info [ q ] ← beacon ;
21         end
22     end
23 end
```

Algorithm 1: Detecting malicious nodes and false location attacks on node p.

The malicious node detection procedure maintains the list of malicious nodes and a flag indicating if it is located in a bad neighborhood. The state associated with malicious node detection is temporary and will be cleaned on expiry of the pseudonym. As shown in Algorithm 1, received beacons permit nodes to update their neighbor information and to decide if the sender is malicious. Malicious nodes are detected in case of infeasible distances between nodes as shown on lines 6, 10, and 13. The node can also decide the neighborhood is bad if there are large disagreements in neighbor node locations, as shown on line 14. While nodes located in bad neighborhoods do not forward messages, they continue to transmit the beacon in order to propagate geographic hashes and their knowledge of malicious nodes in the one-hop neighborhood.

Forwarding validation

Operating in promiscuous mode permits overhearing wireless transmissions of one-hop neighbors. Incorrect forwarding is detected and the incorrectly forwarding nodes classified as malicious. Nodes forward signed messages during GEOGRAPHIC FORWARDING and REVERSE RESPONSE FORWARDING stages. These operations can be abstracted to a simple multi-hop ad-hoc forwarding protocol as follows: Let A, B, and C be successive hops on a routing path. All the transmissions made by B must be received by A because of the one-hop neighbor relationship. Therefore, A can detect if B fails to forward a message by listening for the next-hop transmission. In this case, the honest previous hop node A can identify the malicious node B without the need for other honest witness nodes. Similarly, A can check the overheard next-hop transmission (from B to C) for malicious payload modification or control data modification. Data tampering by B is detectable as the message from B to C is digitally signed with the private key of B, which can be verified by A because of being in the one-hop neighborhood. The node A detects that the next-hop for its message is a malicious node.

In case both B and C are malicious, B can forward the message correctly to C, and later collude by not reporting a malicious forwarding by C. This collusion case is detected by the honest witness node G, which is the one-hop neighbor of A, B, and C. We note that such a node must exist in an honest witness network according to Definition 8. The honest witness G will observe the malicious action of C, and broadcast this malicious action to its one-hop neighborhood through the periodic beacon. Since A is in the one-hop neighborhood of G, it will also mark the node C as malicious. But A also overhears the forwarding of its message from B to C. Thus, A detects that its message is being forwarded through a malicious node. On encountering malicious nodes, the previous hop node A will find another route to the destination or send back a routing failure to the source.

Nodes check the integrity of messages and feasibility of routing paths. Both GEOGRAPHIC FORWARDING and REVERSE RESPONSE FORWARDING stages require messages to carry the geographic location list $\mathbb{L}$ of nodes on the routing path. Forwarding nodes check the integrity of the digitally signed message and the feasibility of the routing path. Since the system has the global upper limit R on the range of nodes, a feasible path can not have successive nodes A and B at a distance greater than R:

$$R \geq \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2} \tag{4.1}$$

If this range constraint is violated, then the message is faulty or the previous hop node is malicious. Messages violating the range constraint are not forwarded and their previous-hop nodes are classified as malicious.

4.6 Security analysis

An adversary can attempt to attack our routing protocol in a number of ways. As outlined in Section 4.4.3, the threats can be classified into two categories: attacks against the beacon and attacks against the routed messages. This section demonstrates how our protocol withstands the threats.

4.6.1 False location attacks

Beacons propagate neighbor information within the one-hop neighborhood. This information is used for geographic routing. Beacons are directly transmitted to one-hop neighbors. Transmission within the one-hop neighborhood obviates the need for protecting against dropping messages and incorrect routing. Direct transmission also protects against payload and control data modification. Since beacons are digitally signed with the private key of the transmitting node, another node can not construct a false beacon. Malicious nodes can use beacons to launched false position attacks and try to cause incorrect routing. The following two lemmas show that these attacks are either detected or are harmless.

LEMMA 1 *Given a pair of honest nodes A and B within one-hop distance of each other, false location attacks that change the expected one-hop relationship to A or B are detected.*

PROOF: Let C be a malicious node with real location l_C. C launches a false location attack by reporting a false location $\acute{l_C} \neq l_C$. Now, consider the one-hop relationship of possible locations to A and B. Let L_A be the set of locations in the one-hop neighborhood of A, and L_B the set of locations in the one-hop neighborhood of B. The reported false location $\acute{l_C}$ can change the expected one-hop relationship to A or B in one of the following ways:

- If $\acute{l_C} \notin L_A \cup L_B$ and $l_C \in L_A \cup L_B$: The beacon of C is heard by at least one of A or B. The node hearing the beacon marks C as malicious because of the violation of range constraint as shown on line 6 of Algorithm 1. The other node learns about C being malicious through the periodic beacon of its peer (see the beacon specification in Definition 12). Thus, both A and B detect the false location attack and mark C malicious.

- If $\acute{l_C} \in L_A - L_B$, then A should hear the beacon and B should not. If $l_C \notin L_A - L_B$, then either A will not hear C's beacon or B will hear it. Since A and B are one-hop neighbors, they share the location information received from beacons. Both A and B will recognize that either A does not hear C's beacon or B does. This happens on lines 10 and 13 of Algorithm 1. In this case, l_C is false because the range constraint is violated for one of two nodes. Beacon exchanges allow both A and B to detect the false location attack and mark C as a malicious node.

- The case when $l_C^{'} \in L_B - L_A$ follows by symmetry from the previous case.
- If $l_C^{'} \in L_A \cap L_B$, then both A and B should hear the beacon of C. If $l_C \notin L_A \cap L_B$, then one of A or B will not hear the beacon directly but receive it indirectly through the other node. It will detect the false location attack at line 13 of Algorithm 1 and mark C as malicious. Beacon exchanges allow both A and B to share this information and mark C as a malicious node.

In each of the possible cases when the false location has a different one-hop relationship with A and B as compared to the real location, the false location attack is detected. □

LEMMA 2 *Given a pair of honest nodes A and B within one-hop distance of each other, false location attacks that do not change the expected one-hop relationship to A or B, do not affect the correctness of routing through nodes A or B in an honest witness network.*

PROOF: Let C be a malicious node at location l_C, and let C report a false location $l_C^{'} \neq l_C$. Let L_A be the set of locations in the one-hop neighborhood of A, and L_B the set of locations in the one-hop neighborhood of B. The reported false location $l_C^{'}$ has the same one-hop relation to A and B as l_C. The following cases can be distinguished by the one-hop relationship of C to A and B:

- If $l_C^{'}, l_C \notin L_A \cup L_B$, then C is not used for routing because of being out of range. Thus, routing through A or B is not affected.
- If $l_C^{'}, l_C \in L_A - L_B$, then A is in the one-hop neighborhood of C, and hears all the transmissions from C. We note that C can only lie on routing paths through A because it is out of the range of B. If the reported false location $l_C^{'} \in L_A - L_B$ is not on a routing path through A, then it can not impact the correctness of routing. If $l_C^{'}$ is on a routing path through A, and X is the next-hop node on it, then X can either be within one-hop of l_C or not. If X is within one-hop of l_C, then the location attack does not affect routing because the next hop receives the message.

  If the next-hop X is not within one-hop of l_C, then by the honest witness assumption, an honest node G must exist within one hop of A, l_C, and X. G will observe the message from C, but not the forwarded message from the next-hop X. The witness G will broadcast the error to its one-hop neighborhood, which includes A. Now, A will either select a different next-hop than C or mark the neighborhood bad and return an error upstream to the previous hop.
- The case when $l_C^{'}, l_C \in L_B - L_A$ follows by symmetry from the previous case.
- The case when $l_C^{'}, l_C \in L_A \cap L_B$ follows by considering the arguments for the previous two cases.

In each of the cases, either the message will reach the destination or an error notification will reach the source. Thus, the location error does not affect the correctness of routing. □

4.6.2 Routing and message integrity

Preserving message integrity and correct geographic forwarding are required for secure routing. As shown in Lemma 1, beacon based malicious node detection detects large location errors, which could affect routing. Undetectable location errors do not affect geographic routing as shown in Lemma 2. This leaves message dropping, routing in the wrong direction, payload modification, and control data modification as the remaining threats to consider. The following two lemmas show that message integrity is preserved, and that messages are routed to the correct destination location.

LEMMA 3 *Message integrity is preserved by the geographical secure path routing protocol operating in an honest witness network.*

PROOF: Consider the message $\mathfrak{M}$ starting at node s with destination location $\mathcal{D}$. Let the sequence of nodes $\mathcal{S}(s,d) \equiv \{s,\ldots,d\}$ denote a feasible geographic routing path from the source node s to a destination node d in the one-hop neighborhood of $\mathcal{D}$. Consider the forwarding of $\mathfrak{M}$ on three successive nodes A, B, and C on the routing path $\mathcal{S}(s,d)$. Let B be the node launching the attack and modifying either the payload or control data.

Recall the routing protocol described in Section 4.5.3. The GEOGRAPHIC FORWARDING operation on node B forwards the message to the next hop C. Since A and B must be one-hop neighbors by virtue of being successive hops on a routing path, A must overhear the transmission of B to C. If B modifies the contents of the message, A will detect the attack, and mark the node B as malicious. The node A can now select another next-hop neighbor, or send back a routing failure if no suitable neighbor exists. In either case, message integrity is not violated. □

LEMMA 4 *In an honest witness network, geographical secure path routing protocol routes the messages to the correct next hop node.*

PROOF: Consider the forwarding of $\mathfrak{M}$ on three successive nodes A, B, and C on the routing path $\mathcal{S}(s,d)$. Let B be the node launching the routing attack. As outlined in Section 4.4.3, the following attacks are possible:

- *Dropping messages*:
  Recall the routing protocol described in Section 4.5.3. The GEOGRAPHIC FORWARDING operation executing on node B forwards the message to the next hop C. Since A and B must be one-hop neighbors by virtue of being successive hops on a routing path, A must

overhear the transmission of B to C. If B drops the message, A will detect the missing transmission and mark B as malicious. The node A either selects another next-hop node or sends a routing error back to the source.

- *Routing in the wrong direction*:
  The GEOGRAPHIC FORWARDING operation is expected to route the message to the correct next hop node C. The malicious node B can launch one of the following attacks on the choice of node C:

  – The node C is not the correct choice among the neighbors $\mathcal{N}(B)$ of node B. Let $D \in \mathcal{N}(B)$ be the correct next-hop neighbor. By assumption of honest witness network, there must be an honest witness node G that is in the one-hop neighborhood of A, B and D. Since the message content can not be modified by Lemma 3, D will overhear the malicious forwarding by B and detect that $\mathfrak{M}$ should have been routed to it, but has been routed to C instead. D will mark B as malicious and share the information with G. G will send the error back to A.

  – Node B manufactures a false location for C. An honest witness G to A, B, and C will pass back the correct geographic hash of C to A through the beacon. On receiving a "Local Response" with incorrect geographic hash for the next hop C (see Section 4.5.3), A will detect it is in a bad neighborhood.

  In either case, A selects another next-hop node or sends a routing error back to the source.

- *Collusion among malicious nodes*:
  If both the nodes B and C are malicious, and if C forwards the message in an incorrect direction. The node B can collude with C and not report the error back to A. However, in the honest witness network, there must be an honest witness node G that is in the one-hop neighbor of A, B, and C. The node G will detect C dropping the message or routing in the incorrect direction. The proof follows from the previous two cases of this lemma.

In all the cases above, the next-hop is selected correctly if it exists. If no appropriate honest node can be chosen, a routing error is sent back to the source. □

4.6.3 Security of geographical secure path routing

The security goals of geographical secure path routing follow from the resistance to false location attacks, correctness of routing, and preservation of message integrity. These properties have been established in the previous lemmas for honest witness networks.

THEOREM 1 *In an honest witness network, our geographical secure path routing protocol provides the following properties:*

- *If there is a secure routing path $\mathcal{S}(s,d)$ from the source node s to a node d located within one-hop distance of the destination location $\mathcal{D}$, then $\mathfrak{M}$ is routed to d.*
- *The destination node d receives the secure routing path $\mathcal{S}(s,d)$.*
- *On receiving the returned response, the source node s gets the correct public key K_d of the destination, and the secure routing path $\mathcal{S}(s,d)$ traversed by the message.*

PROOF: Consider the message $\mathfrak{M}$ being routed on $\mathcal{S}(s,d)$. At each hop the routing path malicious actions of neighboring nodes are either detected or are harmless for geographic forwarding from Lemmas 1 and 2. Hence, the routing path avoids malicious nodes. Lemma 4 ensures that messages are routed to the correct next-hop nodes for geographic routing. Since geographic routing considers all honest candidates for routing at every hop, the routing protocol will find a secure routing path if it exists. Message integrity is preserved at every hop according to Lemma 3. Thus, the message $\mathfrak{M}$ will be delivered to d if there is a secure routing path $\mathcal{S}(s,d)$ from source to destination.

The destination node learns about the secure routing path because the "Forward" messages in the routing protocol (see Section 4.5.3) carry the list of node locations. Because each hop preserves message integrity, the received location list must be correct.

The source node receives the "Recursive Response" message by source routing on the reverse path. The message carries the public keys and locations of the nodes on $\mathcal{S}(s,d)$. Since message integrity is guaranteed at every hop, the received public key and routing path are correct. □

4.6.4 Node density requirement

The existence of honest witnesses is required to guarantee the security of geographical secure path routing protocol. Increasing node density makes it more likely that honest witnesses shall exist for every segment of a routing path. The practical way of determining node density would require a separate information channel because a single physical node can pose as multiple nodes. Nodes are therefore expected to have a side channel that can estimate node density in their one-hop neighborhood.

If a given fraction of nodes can always be assumed to be honest, then honest witness density can be related to node density. Honest witness node requirement can then be met by modifying the forwarding procedure, and requiring a higher node density.

LEMMA 5 *Let R be the node connectivity radius, such that nodes closer than R are always one-hop neighbors. If successive hops are selected within $\frac{R}{3}$ distance of each other, and if the honest node density is greater than $\frac{18}{R^2}$, then the network is an honest witness network.*

PROOF: Consider a square with diagonal $\frac{R}{3}$. Given that the honest node density is greater

than $\frac{18}{R^2}$, each of the four quadrants of the square must have an honest node. By design each of the nodes in the four quadrants are one-hop neighbors of each others. Consider a message that enters the square at node A placed in one of the quadrants. One of the remaining three nodes must be the next hop because geographic routing routes by direction and the next hop must be within $\frac{R}{3}$ distance.

Without loss of generality, suppose the opposite diagonal node B is the next hop. The distance between A and B is at most $\frac{R}{3}$. Let C and D be the remaining two nodes in the square, and let X be the next hop from B. The distance between X and B is at most $\frac{R}{3}$ because successive hops must be selected within $\frac{R}{3}$ distance of each other. The distance between B and C is at most $\frac{R}{3}$ by construction. By triangle inequality, the distance between X and C is at most $\frac{2R}{3}$. Since A, B, and C are one-hop neighbors of each other, C is the honest witness for the routing segment A, B, X. If instead of node B, nodes C or D were chose as the next hop from A, the same argument would apply with B being the honest witness.

The plane can be tiled into squares of diagonal $\frac{R}{3}$, each having 4 honest nodes in order to have the honest node density of $\frac{18}{R^2}$. Therefore, the above argument must apply at all locations. Because every routing segment has an honest witness node, the network is an honest witness network. □

Node density estimation is also required for privacy protection. Changing node pseudonyms in mix zones is sufficient for privacy protection [18]. Mix zones are regions with sufficient node density that can provide large enough anonymity sets to support the desired level of anonymity. The pseudonym changing procedure requires a sufficient number of nodes to silence their transmissions while changing pseudonyms. This prevents an attacker from associating two versions of the pseudonym to the same physical node, thereby providing location privacy.

4.6.5 Overheads

The overhead imposed by the geographical secure path routing protocol can be expressed in terms of the routing and deployment parameters. Consider a geographic routing path consisting of p nodes. Recall from Section 4.5.3 that the "Forward" messages contain the list of node locations. Then, the message size overhead on the forward path is $\mathbf{O}(p)$. The computational cost on the forward path is two public key operations per forwarded message per node. One operation is for digital signature verification on the incoming message, and the second is for signing the outgoing message with the private key. The overhead on the return path is of the same order as that on the forward path.

Secure geographic path routing also introduces an overhead on the periodic beacon transmitted by the node. The beacon carries constant size overhead caused by the nonce, digital signature, and the public key (see Definition 12). It also contains neighbor location list, and the geographic hash list, which linearly scale with the node density. Thus, given a node density of

ρ, the beacon overhead is $\mathbf{O}(\rho)$.

4.7 Performance evaluation

Performance of geographic secure path routing is evaluated by the NS2 network simulator [72]. Although NS2 has support for wireless and mobile ad-hoc network simulation, geographic routing is not available in the standard NS2 code. Therefore, we use the patch provided by Kiess [52, 51], which maintains Karp's original implementation of GPSR [50]. The patch simulates IEEE 802.11 MAC layer with a node range of 500m. It provides support for mobility through the random way point model [12].

Geographic routing takes all routing decisions based on the local one hop neighborhood. Since secure routing detects and avoids malicious nodes, the changes to routing performance can be evaluated by changing the routing behavior to avoid malicious nodes. The remaining protocol operations just authenticate node locations and public keys without affecting routing. Therefore, we change only the local routing behavior of geographic routing in order to assess the performance of GSPR. We modified the simulator code to keep track of malicious or honest nodes. The routing mechanism was also modified to operate in the base mode or to avoid malicious nodes for routing.

The objective of this evaluation is to compare the routing performance and attack resilience of traditional insecure GPSR protocol against our proposed geographical secure path routing protocol. We select percentage rate of data delivery and the routing path length as the indicators of routing performance. The comparative evaluation of the two routing protocols is done for various combinations of node density, mobility, and the presence of malicious nodes.

4.7.1 Node density

GPSR and secure routing were simulated on NS2 using one constant bit rate source per node. Simulation is done with stationary nodes randomly placed on a square area with side 800m. The number of nodes was varied from 20 to 50 to create a number of node density scenarios. The simulation was run for 300 seconds of simulated time. An average of 10 runs was used in the following observations. The delivery rate was calculated by counting the number of application packets sent and received. Path length was computed by tracing GPSR packets through forwardings. As shown in Figure 4.5, the baseline GPSR protocol attains a high percentage of packet delivery. This is consistent with Karp's observations in [50]. It can be observed that introducing 20% malicious nodes severely impacts the effectiveness of GPSR as demonstrated by the reduction in delivery rate. Geographic secure path routing is resilient to the malicious nodes. Its delivery rate closely replicates the rate achieved by GPSR in a benign environment.

The impact of incorrect routing introduced by malicious nodes is shown in Figure 4.6.

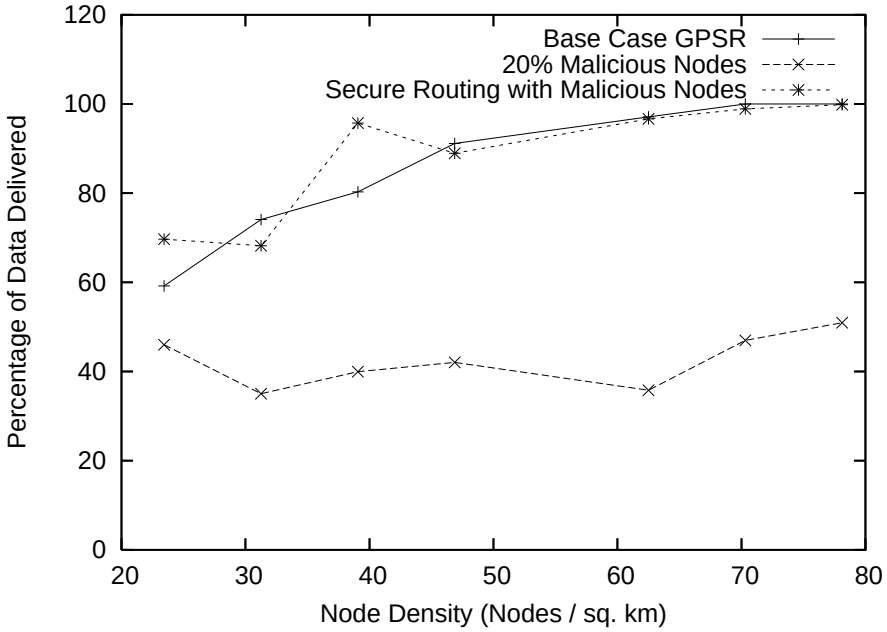

Figure 4.5: Data delivery vs. node density on a square area with stationary nodes.

Given the node range of 500m, and the 0.64km^2 node placement area, we expect nodes to be about 2 hops away in the greedy routing case. This is consistent with our average reading of 3.38 hops for GPSR in benign environment. We also note that secure routing incurs an overhead on the routing path length by routing packets around malicious nodes. The average number of hops for geographic secure path routing with 20% malicious nodes is 10.37, an increase of about three times over GPSR in benign environment.

4.7.2 Effect of malicious nodes

Malicious nodes publish an incorrect location on their beacons. We modify the published location of malicious node to appear as the closest neighbor for the packet being forwarded. This allows malicious nodes to falsely become the next hop neighbors. Malicious nodes also forward the packet to a random neighbor in violation of geographic routing rules. We studied the effect of malicious nodes on GPSR and geographic secure path routing by running an NS2 simulation with varying proportions of malicious nodes on a universe of 42 stationary nodes. These nodes are placed randomly in a rectangular area of 1.5km by 0.5km. Observations are collected by averaging the collected over 10 runs of 300 seconds simulated time each.

The data delivery achieved in presence of malicious nodes is shown in Figure 4.7. It can be observed that insecure geographic routing is very sensitive to malicious nodes. The delivery rate falls rapidly even with a small percentage of malicious nodes. The simulation also indicates that malicious nodes do not affect the delivery rate of our secure routing. The secure delivery

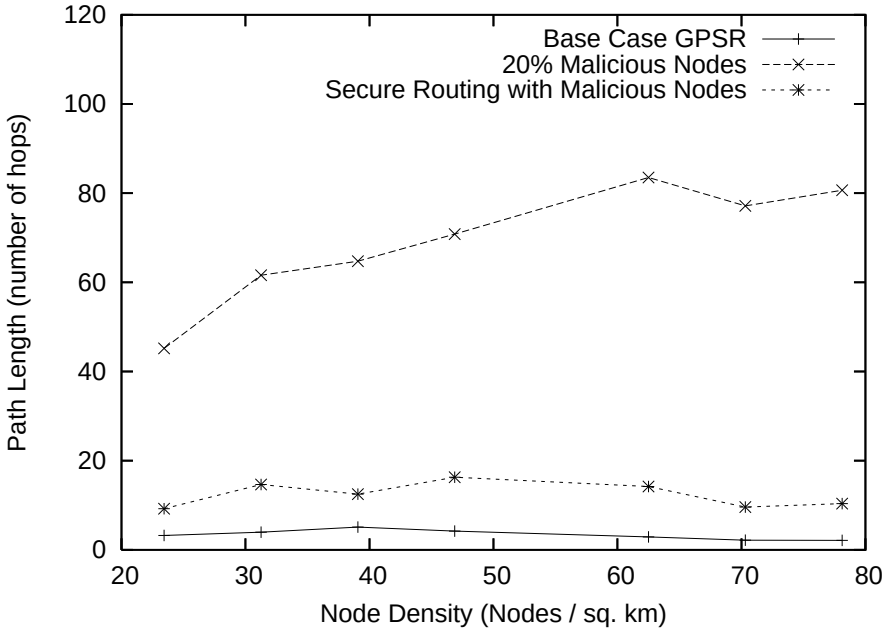

Figure 4.6: Routing path length vs. node density on a square area with stationary nodes.

rate falls from close to 100% in a benign environment to about 90% when 40% of the nodes are malicious. The effect on forwarding path length is shown in Figure 4.8. The path length shown by insecure geographic routing grows by orders of magnitude as the malicious nodes force the insecure protocol to route in incorrect directions. Geographic secure path routing incurs a more modest overhead by rejecting malicious nodes for routing.

4.7.3 Effects of mobility

The effect of mobility on data delivery rate is given in Figure 4.9. Mobility improves the data delivery rate of secure routing because mobility allows nodes to discover new honest nodes. This effect is also found in the insecure geographic routing to a smaller degree. Geographic routing in insecure environment becomes less effective with increasing mobility because of increasing chance of inaccuracy in one hop node locations. We also note that mobility reduces the routing path length of secure routing as shown in Figure 4.10. This happens because mobility increases the chance of discovering new honest nodes in the one hop neighborhood.

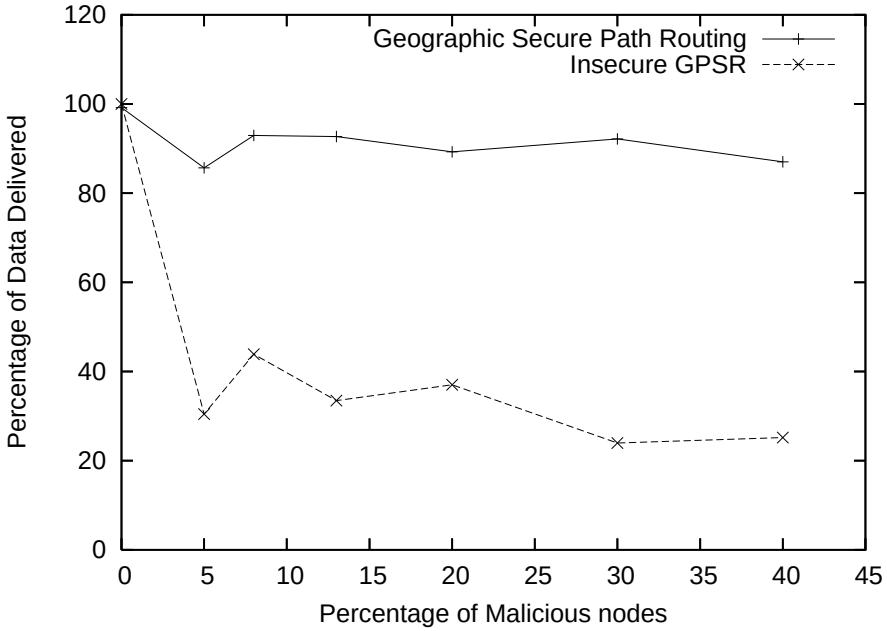

Figure 4.7: Effect of malicious nodes on data delivery.

4.8 Summary

We design and evaluate geographical secure path routing, a privacy preserving ad-hoc routing protocol that geographically routes messages through anonymous nodes to destination locations. The secure routing protocol also authenticates the public key and the geographic location of destination nodes. This enables private communication with nodes located in a given geographic area.

Geographical secure path routing protocol requires associative cryptographic one-way hash functions for security. These hash functions are derived from the discrete logarithm problem, which uses expensive modular arithmetic. This makes our protocol unsuitable for power limited devices. This limitation could be overcome by implementing the protocol with light weight cryptographic primitives or by applying it in resource rich application domains like vehicular ad-hoc networks.

We evaluated the geographical secure path routing protocol using the NS2 network simulator for various values of node density, node mobility, and fraction of malicious nodes. Evaluation results show that the protocol tolerates malicious nodes with an increased routing path length. The geographical secure path routing protocol is also able to maintain a low loss rate even when the majority of nodes are malicious.

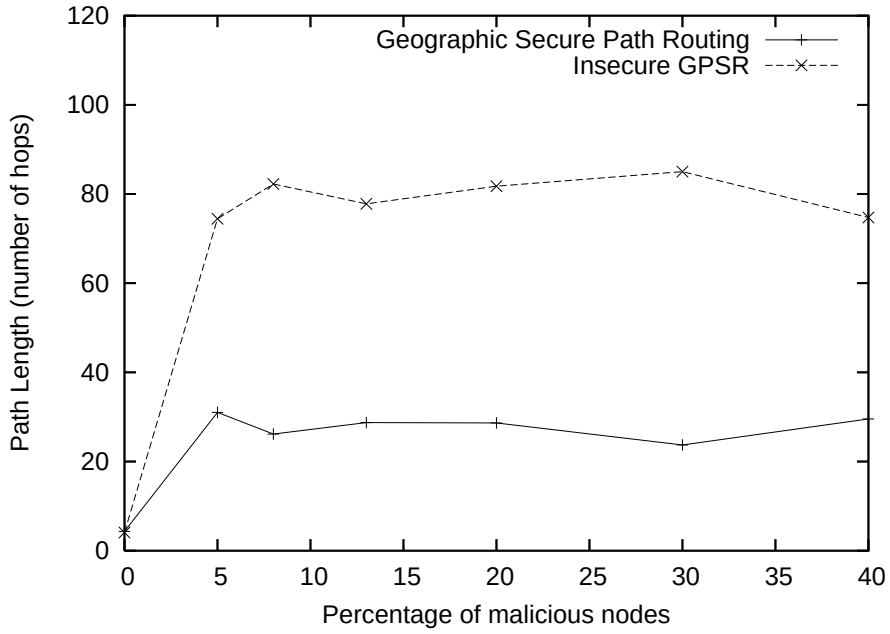

Figure 4.8: Effect of malicious nodes on path length.

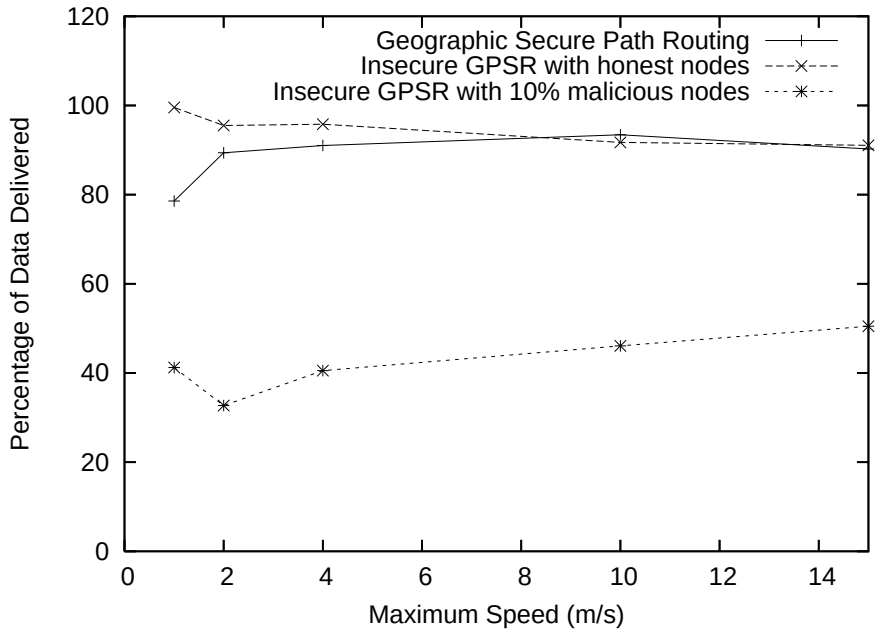

Figure 4.9: Effect of mobility on data delivery for baseline and with 10% malicious nodes.

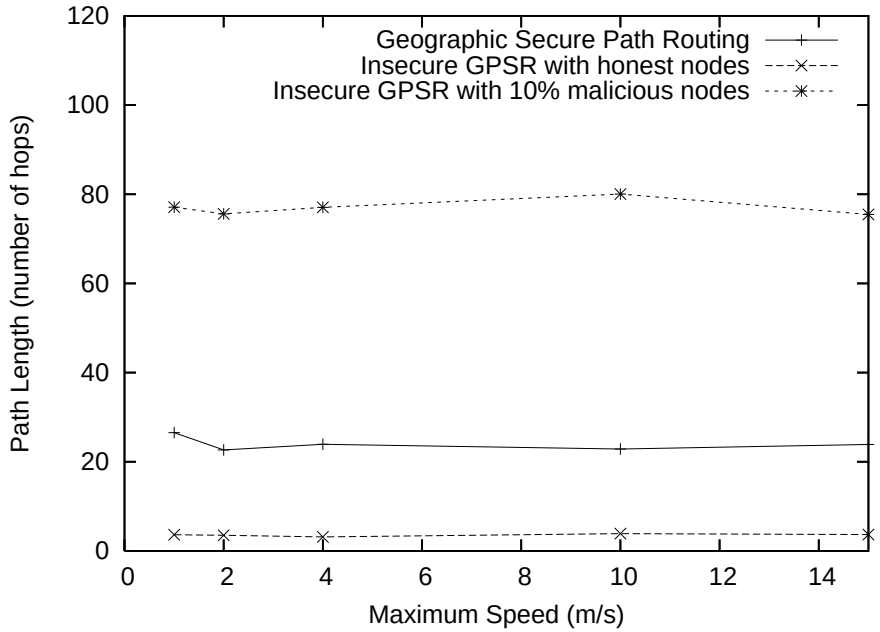

Figure 4.10: Effect of mobility on forwarding path length for baseline and with 10% malicious nodes.

Chapter 5

Conclusions

The ongoing trend towards participative and open systems challenges the traditional centralized view of authentication. Better access along with cost improvements have made it possible for an ever increasing number of players to create and publish content. The trend towards democratized content creation goes beyond the web. Large numbers of real and fictitious email users saturate the Internet with spam emails. Authenticating and trusting peers in these massive open systems is a huge challenge.

Authentication and trust have been studied for a long time in traditional settings. Centralized trusted authorities are commonly used in traditional hierarchical organizations. Their trust model is usually closed, and an explicit permission from the administrator is required to enter the trusted group. Centralized authorities may be used either in a localized manner or on a global scale as being the infrastructure provider. In both of these cases, centralized trust models are inadequate or unsuitable for addressing the trust and authentication needs of open systems. Semi-automatic decentralized authentication methods are also well established. The need for human judgment or infrastructure support makes this class of solutions inappropriate for massive open systems. The users of these systems are relatively unsophisticated. These scenarios therefore need automatic authentication and trust deduction.

With a view towards solving these real life problems, in this dissertation, we show that robust decentralized authentication of public keys and geographic locations can be achieved by assuming the presence of honest participants instead of relying on centralized security infrastructure or human trust judgment. The supporting contributions are described next.

5.1 Contributions

We developed *Byzantine fault tolerant public key authentication* (BPKA), a protocol for automatic authentication of public keys in an open peer-to-peer network. The authentication protocol tolerates byzantine faults and malicious peers without using centralized trusted authorities. It works correctly if a majority of the peers are honest. Our authentication protocol provides a new approach to tackle the authentication problems of distributed and peer to peer systems. Its salient features are lack of total trust and single points of failure. The protocol allows a natural

growth of trust without requiring trustworthy hierarchies of delegating and recommending parties as done in other trust management systems [13, 103]. It is made feasible by a weakened network adversary model. Although weaker than the traditional model in terms of adversary power, our authentication model is stronger in terms of fault tolerance.

We also develop, implement, and evaluate *Social-group key authentication* (SGKA), an automatic, byzantine fault tolerant authentication system for email. Our authentication system operates without trusted third parties, is incrementally deployable, and backward compatible with the existing email infrastructure. It is implemented entirely at the email client in accordance with the end-to-end principle. This enables the creation of user controlled fine grained trust policies that can cross organizational and administrative boundaries. We have implemented the authentication mechanism on the Thunderbird email client. Our authentication mechanism has been evaluated through micro-benchmarks, and with two real life email traces. Evaluation results show that the overheads are acceptable, and that the sender authentication mechanism is effective in real life scenarios. The results also show that most of the peers can be authenticated with only a minor overhead on the email network.

We design and evaluate *Geographical secure path routing* (GSPR), a privacy preserving ad-hoc routing protocol that geographically routes messages through anonymous nodes. Our protocol secures geographic routing against malicious nodes. Geographic locations of anonymous nodes are authenticated to provide location authentication and anonymity simultaneously. Our solution eliminates the need for secure initialization and works correctly if there is a sufficient density of honest nodes. The protocol was evaluated using the NS2 network simulator for various scenarios of node density, node mobility, and proportion of malicious nodes. Evaluation results show that although the protocol tolerates malicious nodes with an increased routing path length, it able to maintain a low loss rate even when a majority of nodes are malicious.

5.2 Vision for the future

Given the increasing ease and economy of participation in the the digital world, it appears that an ever increasing number of identities, documents, relationships, and properties would need to be authenticated in an open trust model. The trust model would permit peers to freely join and leave their trusted groups. It would also allow them to make trust deductions about known and unknown peers in the system. Future authentication models would also take mobility into account, and permit mobile peers the same level of privacy and trust as given to static peers. Since the traditional needs for authentication and trust are unlikely to disappear, hybrid trust models and authentication methods are also likely to be useful. Our peer-to-peer authentication mechanism can be applied for content authentication on the Internet and to achieve collaborative spam control. Automatic authentication of participants can also provide a suitable substrate for economic incentive schemes to handle denial of service attacks and more general economic problems in systems of the future. Authentication and privacy protection in ad-hoc networking

environment can provide adequate support for secure ad-hoc social networking in future mobile computing environments.

References

[1] Sender ID Home Page. http://www.microsoft.com/senderid, 2007. This is an electronic document. Date retrieved: February 1, 2007.

[2] XML User Interface Language. http://www.mozilla.org/projects/xul/, 2007. This is an electronic document. Date retrieved: February 1, 2007.

[3] Home of the Mozilla Project. http://www.mozilla.org/, 2008. This is an electronic document. Date retrieved: February 15, 2008.

[4] Yahoo! Anti-Spam Resource Center - DomainKeys. http://antispam.yahoo.com/domainkeys, 2008. This is an electronic document. Date retrieved: February 15, 2008.

[5] ABDUL-RAHMAN, A., AND HAILES, S. Supporting Trust in Virtual Communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, (2000).

[6] AIELLO, W., LODHA, S., AND OSTROVSKY, R. Fast digital identity revocation. In *Proceedings of 18th Annual International Cryptology Conference, Advances in Cryptology - CRYPTO '98* (1998), vol. 1462 of *Lecture Notes in Computer Science*, Springer, pp. 137–152.

[7] ALLMAN, E., CALLAS, J., DELANY, M., LIBBEY, M., FENTON, J., AND THOMAS, M. DomainKeys Identified Mail (DKIM) Signatures. RFC 4871, May 2007.

[8] ANDERSON, R., CHAN, H., AND PERRIG, A. Key infection: Smart trust for smart dust. In *Proceedings of IEEE International Conference on Network Protocols (ICNP 2004)* (Oct. 2004).

[9] AWERBUCH, B., HOLMER, D., NITA-ROTARU, C., AND RUBENS, H. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *Proceedings of the 2002 ACM workshop on Wireless security* (2002).

[10] BACON, J., MOODY, K., AND YAO, W. Access Control and Trust in the Use of Widely Distributed Services. In *Proceedings of Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms* (2001), vol. 2218 of *Lecture Notes in Computer Science*, Springer, pp. 295–310.

[11] BERESFORD, A. R., AND STAJANO, F. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing 2*, 1 (2003), 46–55.

[12] BETTSTETTER, C. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *SIGMOBILE Mob. Comput. Commun. Rev. 5*, 3 (2001), 55–66.

[13] BLAZE, M., IOANNIDIS, J., AND KEROMYTIS, A. D. Trust Management and Network Layer Security Protocols. In *Proceedings of Cambridge Security Protocols International Workshop* (1999), pp. 103–118.

[14] BOSE, P., MORIN, P., STOJMENOVIC, I., AND URRUTIA, J. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks 7*, 6 (2001), 609–616.

[15] CACHIN, C. Distributing Trust on the Internet. In *Proceedings of International Conference on Dependable Systems and Networks (DSN2001), Göteborg, Sweden.* (June 2001), IEEE.

[16] CANETTI, R., GENNARO, R., HERZBERG, A., AND NAOR, D. Proactive Security: Long-term protection against break-ins. *RSA CryptoBytes 3*, 1 (1997), 1–8.

[17] CAPKUN, S., BUTTYÁN, L., AND HUBAUX, J.-P. Small worlds in security systems: an analysis of the PGP certificate graph. In *Proceedings of the ACM New Security Paradigms Workshop, 2002*. (2002).

[18] CAPKUN, S., HUBAUX, J.-P., AND JAKOBSSON, M. Secure and privacy-preserving communication in hybrid ad hoc networks. I&C Research Report 200444, Ecole Polytechnique Fédérale de Lausanne, May 2004.

[19] CCITT. The Directory Authentication Framework. Recommendation X.509, 1988.

[20] CROCKER, D. H. Standard for the format of ARPA Internet text messages. RFC 822, August 1982.

[21] DAMIANI, E., DE CAPITANI DI VIMERCATI, S., PARABOSCHI, S., SAMARATI, P., AND VIOLANTE, F. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security* (New York, NY, USA, 2002), ACM, pp. 207–216.

[22] DEMERS, A., GREENE, D., HAUSER, C., IRISH, W., LARSON, J., SHENKER, S., STURGIS, H., SWINEHART, D., AND TERRY, D. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing* (New York, NY, USA, 1987), ACM Press, pp. 1–12.

[23] DIFFIE, W., AND HELLMAN, M. New Directions in Cryptography. *IEEE Trans. Info. Theory 22* (1976), 644–654.

[24] DINGLEDINE, R., FREEDMAN, M. J., AND MOLNAR, D. The Free Haven Project: Distributed Anonymous Storage Service. In *Proceedings of International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA (USA) July 2000* (2001), vol. 2009 of *Lecture Notes in Computer Science*, Springer.

[25] DOUCEUR, J. The Sybil Attack. In *Proceedings of the IPTPS02 Workshop, Cambridge, MA (USA)* (2002).

[26] DRAVES, R., PADHYE, J., AND ZILL, B. Comparison of routing metrics for static multi-hop wireless networks. In *Proceedings of SIGCOMM* (2004), R. Yavatkar, E. W. Zegura, and J. Rexford, Eds., ACM, pp. 133–144.

[27] FENTON, J. Analysis of Threats Motivating DomainKeys Identified Mail (DKIM). RFC 4686, September 2006.

[28] FERRAIOLO, D. F., SANDHU, R., GAVRILA, S., KUHN, D. R., AND CHANDRAMOULI, R. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur. 4*, 3 (2001), 224–274.

[29] FIDGE, C. Logical time in distributed computing systems. *Computer 24*, 8 (1991), 28–33.

[30] FINN, G. G. Routing and addressing problems in large metropolitan-scale internetworks. Research Report 180, Information Sciences Institute, March 1987.

[31] FOX, B., AND LAMACCHIA, B. Certificate Revocation: Mechanics and Meaning. In *Proceedings of International Conference on Financial Cryptography*, R. Hirschfeld, Ed., vol. 1465 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998, pp. 158–164.

[32] GARFINKEL, S. *PGP: Pretty Good Privacy*. O'Reilly & Associates, Inc., Cambridge, MA, 1995.

[33] GARFINKEL, S. L., AND MILLER, R. C. Johnny 2: a user test of key continuity management with s/mime and outlook express. In *Proceedings of the 2005 symposium on Usable privacy and security* (New York, NY, USA, 2005), ACM, pp. 13–24.

[34] GARRISS, S., KAMINSKY, M., FREEDMAN, M. J., KARP, B., MAZIÈRES, D., AND YU, H. RE: Reliable Email. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation* (San Jose, CA, May 2006).

[35] GOLDWASSER, S., MICALI, S., AND RIVEST, R. L. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput. 17*, 2 (1988), 281–308.

[36] GOODRICH, M. T., TAMASSIA, R., AND YAO, D. Accredited DomainKeys: a service architecture for improved email validation. In *Proceedings of the Conference on Email and Anti-Spam (CEAS '05)* (July 2005).

[37] GÖRLACH, A., HEINEMANN, A., AND TERPSTRA, W. W. Survey on Location Privacy in Pervasive Computing. In *Proceedings of The First Workshop on Security and Privacy at the Conference on Pervasive Computing (SPPC)* (April 2004).

[38] GUTMANN, P. Why isn't the internet secure yet, dammit. In *Proceedings of the AusCERT Asia Pacific Information Technology Security Conference 2004; Computer Security: Are we there yet?* (May 2004).

[39] HERZBERG, A., JAKOBSSON, M., JARECKI, S., KRAWCZYK, H., AND YUNG, M. Proactive public key and signature systems. In *Proceedings of the ACM Conference on Computer and Communications Security* (1997), pp. 100–110.

[40] HOWELL, J., AND KOTZ, D. End-to-end authorization. In *Proceedings of the 4th Symposium on Operating System Design & Implementation* (Berkeley, CA, USA, 2000), USENIX Association, pp. 151–164.

[41] HOWELL, J. R. *Naming and sharing resources across administrative boundaries*. PhD thesis, Dartmouth College, Hanover, NH, USA, 2000. Chair-David Kotz.

[42] HU, Y.-C., JOHNSON, D. B., AND PERRIG, A. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks 1*, 1 (2003), 175–192.

[43] HU, Y.-C., PERRIG, A., AND JOHNSON, D. B. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks 11*, 1-2 (2005), 21–38.

[44] HUBAUX, J.-P., BUTTYÁN, L., AND CAPKUN, S. The quest for security in mobile ad hoc networks. In *Proceedings of the 2nd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2001, October 4-5, 2001, Long Beach, CA, USA* (2001), ACM, pp. 146–155.

[45] JOHNSON, D., HU, Y., AND MALTZ, D. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4 . RFC 4728, February 2007.

[46] JOHNSON, D. B., MALTZ, D. A., AND BROCH, J. DSR: the dynamic source routing protocol for multihop wireless ad hoc networks. In *Ad hoc networking*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001, pp. 139–172.

[47] KAMINSKY, M., SAVVIDES, G., MAZIERES, D., AND KAASHOEK, M. F. Decentralized user authentication in a global file system. *SIGOPS Oper. Syst. Rev. 37*, 5 (2003), 60–73.

[48] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), ACM, pp. 640–651.

[49] KARGL, F., GEISS, A., SCHLOTT, S., AND WEBER, M. Secure Dynamic Source Routing. In *Proceedings of the 38th Hawaii International Conference on System Sciences* (2005).

[50] KARP, B., AND KUNG, H. T. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom '00)* (New York, NY, USA, August 2000), ACM Press, pp. 243–254.

[51] KIESS, W. Hierarchical Location Service for Mobile Ad-Hoc Networks. Master's thesis, Department of Mathematics and Computer Science, University of Mannheim, 2003.

[52] KIESS, W., FÜSSLER, H., WIDMER, J., AND MAUVE, M. Hierarchical Location Service for Mobile Ad-Hoc Networks. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R) 8*, 4 (October 2004), 47–58.

[53] KIM, Y.-J., GOVINDAN, R., KARP, B., AND SHENKER, S. Geographic routing made practical. In *Proceedings of 2nd Symposium on Networked Systems Design and Implementation* (2005), USENIX, pp. 217–230.

[54] KLENSIN, J. Simple Mail Transfer Protocol. RFC 2821, April 2001.

[55] KWON, S., AND SHROFF, N. B. Geographic routing in the presence of location errors. *Comput. Networks 50*, 15 (October 2006), 2902–2917.

[56] LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM 21*, 7 (1978), 558–565.

[57] LAMPORT, L., SHOSTAK, R., AND PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS) 4*, 3 (1982), 382–401.

[58] LEE, S., SHERWOOD, R., AND BHATTACHARJEE, S. Cooperative Peer Groups in NICE. In *Proceedings of INFOCOM* (2003).

[59] LEINMÜLLER, T., AND SCHOCH, E. Greedy routing in highway scenarios: The impact of position faking nodes. In *Proceedings of the Third International Workshop on Intelligent Transportation (WIT) Hamburg, Germany* (March 2006).

[60] LEKKAS, D. Establishing and managing trust within the public key infrastructure. *Computer Communications 26*, 16 (2003), 1815–1825.

[61] LEVI, A., CAGLAYAN, M. U., AND KOC, C. K. Use of nested certificates for efficient, dynamic, and trust preserving public key infrastructure. *ACM Trans. Inf. Syst. Secur. 7*, 1 (2004), 21–59.

[62] LI, B.-H., HOU, Y.-B., AND ZHAO, Y.-L. A scalable scheme for certificate revocation. In *Proceedings of the International Conference on Machine Learning and Cybernetics, 2005* (Aug. 2005), vol. 6, pp. 3852–3856.

[63] LI, N., MITCHELL, J. C., AND WINSBOROUGH, W. H. Design of a role-based trust-management framework. In *Proceedings of the IEEE Symposium on Security and Privacy* (2002), pp. 114–130.

[64] LI, Z., TRAPPE, W., ZHANG, Y., AND NATH, B. Robust statistical methods for securing wireless localization in sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks* (Los Angeles, California, 2005).

[65] LIU, D., NING, P., AND DU, W. K. Attack-resistant location estimation in sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks* (Los Angeles, California, 2005).

[66] LUNDBERG, J. Routing security in ad hoc networks. Tech. Rep. Tik110. 501, Helsinki University of Technology, 2000.

[67] MAHAJAN, R., RODRIG, M., WETHERALL, D., AND ZAHORJAN, J. Sustaining cooperation in multi-hop wireless networks. In *Proceedings of 2nd Symposium on Networked Systems Design and Implementation* (2005), USENIX, pp. 231–244.

[68] MARTI, S., GIULI, T. J., LAI, K., AND BAKER, M. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of MOBICOM* (August 2000), pp. 255–265.

[69] MONTENEGRO, G., AND CASTELLUCCIA, C. Crypto-based identifiers (cbids): Concepts and applications. *ACM Trans. Inf. Syst. Secur. 7*, 1 (2004), 97–127.

[70] NAOR, M., AND NISSIM, K. Certificate Revocation and Certificate Update. In *Proceedings 7th USENIX Security Symposium* (San Antonio, Texas, Jan 1998).

[71] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Digital Signature Standard. FIPS PUB 186, May 1994.

[72] The Network Simulator ns-2 (v2.1b8a). http://www.isi.edu/nsnam/ns/, October 2001.

[73] P. RESNICK, E. Internet Message Format. RFC 2822, April 2001.

[74] PATHAK, V. Automatic peer-to-peer mail authentication plugin. http://discolab.rutgers.edu/sam/, 2007.

[75] PATHAK, V., AND IFTODE, L. Byzantine fault tolerant public key authentication in peer-to-peer systems. *Computer Networks 50*, 4 (2006), 579–596.

[76] PATHAK, V., YAO, D., AND IFTODE, L. Improving email trustworthiness through social-group key authentication. In *Proceedings of the Fifth Conference on Email and Anti-Spam CEAS* (Mountain View, California, 2008).

[77] PATHAK, V., YAO, D., AND IFTODE, L. Securing geographical routing in mobile ad-hoc networks. Tech. Rep. 638, Department of Computer Science, Rutgers University, July 2008.

[78] PATHAK, V., YAO, D., AND IFTODE, L. Securing location aware services over vanet using geographical secure path routing. In *Proceedings of the International Conference on Vehicular Electronics and Safety ICVES* (Columbus, Ohio, 2008).

[79] PERKINS, C. E., AND BHAGWAT, P. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of SIGCOMM* (1994), pp. 234–244.

[80] R. RIVEST, A. S., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.

[81] RAHMAN, S. M. M., MAMBO, M., INOMATA, A., AND OKAMOTO, E. An anonymous on-demand position-based routing in mobile ad hoc networks. In *Proceedings of the International Symposium on Applications and the Internet (SAINT'06)* (2006), IEEE Computer Society, pp. 300–306.

[82] RAMSDELL, B. S/MIME Version 3 Certificate Handling. RFC 2632, June 1999.

[83] RAMSDELL, B. S/MIME Version 3 Message Specification. RFC 2633, June 1999.

[84] REITER, M. K. The Rampart Toolkit for Building High-Integrity Services. In *Proceedings of Dagstuhl Seminar on Distributed Systems* (1994), pp. 99–110.

[85] REITER, M. K., AND STUBBLEBINE, S. G. Authentication metric analysis and design. *ACM Transactions on Information and System Security 2*, 2 (1999), 138–158.

[86] SAHA, A. K., TO, K. A., PALCHAUDHURI, S., DU, S., AND JOHNSON, D. B. Design and Performance of PRAN: A System for Physical Implementation of Ad Hoc Network Routing Protocols. *IEEE Trans. Mob. Comput. 6*, 4 (2007), 463–479.

[87] SALTZER, J. H., REED, D. P., AND CLARK, D. D. End-to-end arguments in system design. *ACM Trans. Comput. Syst. 2*, 4 (1984), 277–288.

[88] SALTZER, J. H., AND SCHROEDER, M. D. The protection of information in computer systems. *Proceedings of the IEEE 63*, 9 (Sept. 1975), 1278–1308.

[89] SANDHU, R., BHAMIDIPATI, V., AND MUNAWER, Q. The ARBAC97 model for role-based administration of roles. *ACM Trans. Inf. Syst. Secur. 2*, 1 (1999), 105–135.

[90] SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. Role-based access control models. *IEEE Computer 29*, 2 (1996), 38–47.

[91] SASTRY, N., SHANKAR, U., AND WAGNER, D. Secure verification of location claims. In *Proceedings of the Workshop on Wireless Security* (San Diego, California, 2003), W. D. Maughan and A. Perrig, Eds., ACM, pp. 1–10.

[92] SHAMIR, A. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology* (New York, NY, USA, 1985), Springer-Verlag New York, Inc., pp. 47–53.

[93] STREIB, M. D. dtype.org. http://www.dtype.org/keyanalyze/, 2002.

[94] SYVERSON, P. F., AND CERVESATO, I. The logic of authentication protocols. In *FOSAD* (2000), R. Focardi and R. Gorrieri, Eds., vol. 2171 of *Lecture Notes in Computer Science*, Springer, pp. 63–136.

[95] THOMAS, M. Requirements for a DomainKeys Identified Mail (DKIM) Signing Practices Protocol. RFC 5106, October 2007.

[96] V. LEGRAND, D. HOOSHMAND, S. U. Trusted Ambient community for self-securing hybrid networks. INRIA Research Report, 5027, 2003.

[97] WALFISH, M., ZAMFIRESCU, J., BALAKRISHNAN, H., KARGER, D., AND SHENKER, S. Distributed Quota Enforcement for Spam Control. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (San Jose, CA, May 2006).

[98] WAN, T., KRANAKIS, E., AND VAN OORSCHOT, P. C. Securing the Destination-Sequenced Distance Vector Routing Protocol (S-DSDV). In *Proceedings of the 6th International Information and Communications Security Conference* (2004), J. Lopez, S. Qing, and E. Okamoto, Eds., vol. 3269 of *Lecture Notes in Computer Science*, Springer, pp. 358–374.

[99] WHITTEN, A., AND TYGAR, J. D. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium* (August 1999).

[100] WONG, M., AND SCHLITT, W. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408, April 2006.

[101] WU, X., AND BHARGAVA, B. K. Ao2p: Ad hoc on-demand position-based private routing protocol. *IEEE Trans. Mob. Comput. 4*, 4 (2005), 335–348.

[102] XU, W., MA, K., TRAPPE, W., AND ZHANG, Y. Jamming sensor networks: attack and defense strategies. *IEEE Network 20*, 3 (May-June 2006), 41–47.

[103] YAHALOM, R., KLEIN, B., AND BETH, T. Trust relationships in secure systems—a distributed authentication perspective. In *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy* (May 1993), pp. 150–164.

[104] YLÖNEN, T. SSH: secure login connections over the internet. In *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography* (Berkeley, CA, USA, 1996), USENIX Association, pp. 37–42.

[105] ZAPATA, M. G., AND ASOKAN, N. Securing ad-hoc routing protocols. In *Proceedings of the 2002 ACM workshop on Wireless security* (2002).

[106] ZHOU, L., AND HAAS, Z. Securing Ad Hoc Networks. *IEEE Network Magazine 13*, 6 (1999).

[107] ZHOU, L., SCHNEIDER, F. B., AND VAN RENESSE, R. COCA: A Secure Distributed On-line Certification Authority. Tech. Rep. 2000-1828, Department of Computer Science, Cornell University, Ithaca, NY USA, December 2000.

[108] ZHU, B., WAN, Z., KANKANHALLI, M. S., BAO, F., AND DENG, R. H. Anonymous secure routing in mobile ad-hoc networks. In *Proceedings of the 29th Annual IEEE Conference on Local Computer Networks (LCN)* (Tampa, Florida, 2004), IEEE Computer Society, pp. 102–108.

[109] ZIMMERMANN, P. *The Official PGP User's Guide*. MIT Press, Cambridge, Massachusetts, 1995.