Technical Report

UCAM-CL-TR-718
ISSN 1476-2986

Number 718

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Cooperative attack and defense in distributed networks

Tyler Moore

June 2008

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted March 2008 by the author for the degree of Doctor of Philosophy to the University of Cambridge, St. John's College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Dedication

To Jillian

Cooperative attack and defense in distributed networks

Tyler W. Moore

Summary

The advance of computer networking has made cooperation essential to both attackers and defenders. Increased decentralization of network ownership requires devices to interact with entities beyond their own realm of control. The distribution of intelligence forces decisions to be taken at the edge. The exposure of devices makes multiple, simultaneous attacker-chosen compromise a credible threat. Motivation for this thesis derives from the observation that it is often easier for attackers to cooperate than for defenders to do so. I describe a number of attacks which exploit cooperation to devastating effect. I also propose and evaluate defensive strategies which require cooperation.

I first investigate the security of decentralized, or 'ad-hoc', wireless networks. Many have proposed pre-loading symmetric keys onto devices. I describe two practical attacks on these schemes. First, attackers may compromise several devices and share the pre-loaded secrets to impersonate legitimate users. Second, whenever some keys are not pre-assigned but exchanged upon deployment, a revoked attacker can rejoin the network.

I next consider defensive strategies where devices collectively decide to remove a malicious device from the network. Existing voting-based protocols are made resilient to the attacks I have developed, and I propose alternative strategies that can be more efficient and secure. First, I describe a reelection protocol which relies on positive affirmation from peers to continue participation. Then I describe a more radical alternative called suicide: a good device removes a bad one unilaterally by declaring both devices dead. Suicide offers significant improvements in speed and efficiency compared to voting-based decision mechanisms. I then apply suicide and voting to revocation in vehicular networks.

Next, I empirically investigate attack and defense in another context: phishing attacks on the Internet. I have found evidence that one group responsible for half of all phishing, the rock-phish gang, cooperates by pooling hosting resources and by targeting many banks simultaneously. These cooperative attacks are shown to be far more effective.

I also study the behavior of defenders – banks and Internet service providers – who must cooperate to remove malicious sites. I find that phishing-website lifetimes follow a long-tailed lognormal distribution. While many sites are removed quickly, others remain much longer. I examine several feeds from professional 'take-down' companies and find that a lack of data sharing helps many phishing sites evade removal for long time periods.

One anti-phishing organization has relied on volunteers to submit and verify suspected phishing sites. I find its voting-based decision mechanism to be slower and less comprehensive than unilateral verification performed by companies. I also note that the distribution of user participation is highly skewed, leaving the scheme vulnerable to manipulation.

Acknowledgments

I would like to thank my supervisor, Professor Ross Anderson, for his advice and support during my time in Cambridge. I have especially appreciated his encouragement to pursue a wide range of research topics.

I gratefully acknowledge the financial support of the UK Marshall Aid Commemoration Commission and the US National Science Foundation. I am especially indebted to my undergraduate mentor, Professor Sujeet Shenoi, for encouraging me to apply for a Marshall Scholarship which has made possible my studies in Cambridge.

I must also thank the members of the Security Group for their fruitful discussions and collaborations. In particular, I thank Dr. Richard Clayton for proofreading nearly all of my papers, as well as Shishir Nagaraja, Dr. Hao Feng and Dr. Dan Cvrček for patiently listening to many nascent research ideas. I have especially appreciated the contributions from my co-authors, notably Dr. Jolyon Clulow and Dr. Richard Clayton. I would also like to thank my examiners, Dr. Frank Stajano and Professor Peter Ryan, for their helpful comments. Of course, the responsibility for any remaining errors or omissions is mine.

I am deeply grateful for assistance received from the anti-phishing community generally and from the companies we have called *A* and *B* in particular. Without their data, the information they have provided about their clients, and their ongoing advice about how the industry works, the research described in Chapters 6–9 would not have been possible.

Finally, I would like to thank my family, and especially my parents Paul and Rebecca Moore, for their support and encouragement throughout my education. I am especially indebted to my wife, Jillian Moore, for her devotion in agreeing to move to the UK and supporting me during our time spent here.

Contents

Published work

In the course of my studies, I have published the following papers, book chapter, edited volume and commissioned report. Many discuss topics covered in this thesis while others describe distinct research threads. Two of my papers have been given best paper awards: 'New strategies for revocation in ad-hoc networks' (ESAS 2007) and 'Examining the impact of website take-down on phishing' (eCrime 2007). The latter paper was also named 2008 Computer Laboratory Publication of the Year.

Economics of information security

Tyler Moore and Richard Clayton, 'Evaluating the wisdom of crowds in assessing phishing websites', in *12th International Financial Cryptography and Data Security Conference (FC)*, Springer Lecture Notes in Computer Science (LNCS) (to appear), 2008.

Ross Anderson, Rainer Böhme, Richard Clayton, and Tyler Moore. 'Security economics and the internal market', report commissioned by the European Network and Information Security Agency (ENISA), 2008.

Tyler Moore and Richard Clayton, 'Examining the impact of website take-down on phishing', in *2nd Anti-Phishing Working Group eCrime Researcher's Summit (APWG eCrime)*, 2007, pp. 1–13.

Ross Anderson and Tyler Moore, 'Information security economics – and beyond', in *27th Annual International Cryptology Conference (CRYPTO)*, Springer LNCS, vol. 4622, pp. 68–91, 2007.

Tyler Moore and Richard Clayton, 'An empirical analysis of the current state of phishing attack and defence', in *6th Workshop on the Economics of Information Security (WEIS)*, 2007.

Ross Anderson, Tyler Moore, Shishir Nagaraja, and Andy Ozment, 'Incentives and information security', in *Algorithmic Game Theory*, Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay Vazirani, Eds. New York: Cambridge University Press, 2007, pp. 633–649.

Ross Anderson and Tyler Moore, 'The economics of information security', *Science*, vol. 314, no. 5799, pp. 610–613, 2006.

Tyler Moore, 'The economics of digital forensics', in *5th WEIS*, 2006.

Tyler Moore, 'Countering hidden-action attacks on networked systems', in *4th WEIS*, 2005.

Wireless network security

Tyler Moore, Maxim Raya, Jolyon Clulow, Panos Papadimitratos, Ross Anderson and Jean-Pierre Hubaux, 'Fast exclusion of errant devices from vehicular networks', in *5th IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2008 (to appear).

Tyler Moore, Jolyon Clulow, Shishir Nagaraja and Ross Anderson, 'New strategies for revocation in ad-hoc networks', *4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, Springer LNCS, vol. 4572, pp. 232–246, 2007.

Frank Stajano, Catherine Meadows, Srdjan Čapkun, and Tyler Moore, Eds., *Security and Privacy in Ad-hoc and Sensor Networks, 4th European Workshop, ESAS 2007, Proceedings*, Vol. 4572, Lecture Notes in Computer Science, Heidelberg: Springer, 2007.

Tyler Moore and Jolyon Clulow, 'Secure path-key revocation for symmetric key predistribution schemes in sensor networks', in *New Approaches for Security, Privacy and Trust in Complex Environments, Proceedings of the IFIP TC 11 22nd International Information Security Conference (SEC 2007)*, Vol. 232, IFIP, H. Venter, M. Eloff, L. Labuschagne, J. Eloff, and R. von Solms, Eds., Heidelberg: Springer, 2007, pp. 157-168.

Jolyon Clulow, Gerhard Hancke, Markus Kuhn and Tyler Moore, 'So near and yet so far: distance-bounding attacks in wireless networks', in *3rd ESAS*, Springer LNCS, vol. 4357, pp. 83–97, 2006.

Jolyon Clulow and Tyler Moore, 'Suicide for the common good: a new strategy for credential revocation in self-organizing systems', *ACM SIGOPS Operating Systems Review*, vol. 40, no. 3, pp. 18–21, 2006.

Tyler Moore, 'A collusion attack on random pairwise key predistribution schemes for distributed sensor networks', in *IEEE International Workshop on Pervasive Computing and Communications Security (PerSec)*, 2006, pp. 251–255.

Selected presentations and professional activities

In addition to formal publications, I have tried to disseminate my research by giving presentations to research groups at other universities and to conferences without proceedings. I have also helped organize several academic conferences and served on program committees.

Presentations

'The economics of information security', invited talk to the DeepSec In-Depth Security Conference, Vienna, Austria, November 22, 2007.

'Network economics and security engineering', presented at the DIMACS Workshop on Information Security Economics, Rutgers University, Piscataway, NJ, USA, January 18–19, 2007.

'The economics of information security', invited talk to the Institute for Security Technology Studies, Dartmouth College, Hanover, NH, USA, January 16, 2007.

'The economics of information security', invited talk to the Laboratory for Computer Communications and Applications, EPFL, Lausanne, Switzerland, November 30, 2006.

'Economic challenges to improving information security', invited talk to the International Workshop on Cyber-security, Danish Board of Technology, Copenhagen, Denmark, September 14, 2006.

'A survey of recent results in the economics of information security', invited talk to the Conference on Network and Information Security in Cyprus—Policy and Implementation of Standards, Nicosia, Cyprus, April 28, 2006.

'On economics and information security', keynote address to the Network of Networks (NVN) Workshop, Amsterdam, The Netherlands, January 26, 2006.

Professional Activities

General Chair

5th Workshop on the Economics of Information Security, University of Cambridge, UK, June 2006.

13th International Conference on Financial Cryptography and Data Security, Barbados, 2009.

Local Arrangements Chair

4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks, University of Cambridge, UK, July 2007.

Program Committee Member

7th Workshop on the Economics of Information Security, Dartmouth College, Hanover, NH, USA, June 2008.

6th Workshop on the Economics of Information Security, Carnegie Mellon University, Pittsburgh, PA, USA, June 2007.

4th IEEE Workshop on Pervasive Computing and Communications Security, White Plains, NY, USA, March 2007.

2nd IFIP WG 11.10 International Conference on Critical Infrastructure Protection, George Mason University, Arlington, VA, USA, March 2008.

1st IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Dartmouth College, Hanover, NH, USA, March 2007.

4th IFIP WG 11.9 International Conference on Digital Forensics, Kyoto, Japan, Januray 2008.

3rd IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 2007.

2nd IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 2006.

Reviewer

ACM Computing Surveys

Session Chair

12th International Conference on Financial Cryptography and Data Security, Cozumel, Mexico, January 2008.

Work done in collaboration

Chapter 3 is based on two papers, 'A collusion attack on random pairwise key predistribution schemes for distributed sensor networks' [113] and 'Secure path-key revocation for symmetric key pre-distribution schemes in sensor networks' [118]. I am the sole author of the former paper, while I co-wrote the latter with Jolyon Clulow. Some of the attacks in Section 3.4.1 were developed with Jolyon who independently reached the same list of mitigation strategies presented in Section 3.5.

Chapter 4 is based on the paper 'New strategies for revocation in ad-hoc networks' [119]. Jolyon Clulow shares credit for some of the early ideas in Section 4.2, while the implementation of the protocol is mine. The protocol refinement in Section 4.1.2 was suggested by Ross Anderson, while Shishir Nagaraja helped implement the simulations in Section 4.3.3.

Chapter 5 is based on the paper 'Fast exclusion of errant devices from vehicular networks' [120]. Jean-Pierre Hubaux and I independently arrived at the idea to extend the protocol developed in Section 4.2 for use in vehicular networks. Some of the initial ideas were discussed with Jolyon Clulow, Maxim Raya, Panos Papadimitratos, and Jean-Pierre Hubaux. The protocol extension and analysis presented in the chapter, along with the implementation of the simulations, were mine.

Chapters 6–9 are based on papers co-authored with Richard Clayton [116, 117, 115]. Richard implemented the data collection for phishing websites described in Sections 6.3.1. He is also primarily responsible for my understanding of how the rock-phish gang operates as described in Section 6.1.2. Richard and I discussed extensively the data analyzed in Chapters 7 and 8; undoubtedly my present understanding is a product of this collaboration. However, the data analysis conducted in these chapters was mine.

The motivation, data and vulnerability analysis, and attacks presented in Chapter 9 are mine. Jaeyeon Jung corrected an error in an earlier version of the expected overlap formula presented in Section 9.4.3.

Chapter 1

Introduction

Computer networks are becoming increasingly decentralized. No single entity controls the entire system; therefore, no entity can compel compliance. The Internet, for example, is comprised of many service providers, domain registrars and end users following frequently divergent commercial interests.

Increasingly, new network applications are decentralized by design. Peer-to-peer file-sharing systems connect users to exchange information, yet lack a hierarchical organization so they can increase efficiency and avoid single points of failure. Wireless sensor networks are assembled from large numbers of interchangeable, low-cost devices and scattered into an area of interest to perform surveillance or monitoring tasks.

Distributed networks require cooperation among participants to complete even basic tasks. Why? Nodes can perceive only a limited view of system activity, which introduces a degree of interdependence between components. Aggregate measurements require input from many participants, and actions such as routing traffic are likely to be reciprocated in the future. Distributed networks also engender limited trust between participants. In a network of similarly capable peers, none may be recognized as authoritative. This in turn necessitates that cooperative agreements be reached frequently. For example, the Internet's inter-domain routing protocol BGP reflects negotiated policies agreed between service providers.

The opportunities and challenges presented by cooperation extend to attack and defense in distributed networks. It can be easy for attacker-controlled devices to covertly collude to carry out attacks, by sharing key material for instance. For defenders, cooperation can be much harder, since individual participants may have conflicting interests. Trust presents another impediment to cooperation: it can be difficult to determine whether a neighboring network participant is under the control of an adversary. Hence, designing workable mechanisms for cooperatively defending against attacks is hard. This thesis proposes cooperative defensive strategies and empirically examines the adverse effects of non-cooperation in existing applications.

1.1 Prior work

Byzantine Fault Tolerance is the standard view of reliability [93]. Failures are assumed to be random and independent; under these assumptions Lamport, Shostack and Pease showed that a system can be made resilient to the failure of up to one third of its components. Deliberate attacks do not always satisfy these assumptions, however. Albert, Jeong and Barabási showed that some network structures are highly susceptible to targeted removal [7]. Syverson explores how coordination problems and topological heterogeneity can undermine the Byzantine assumptions in practice [153].

A number of distributed network applications suffer from coordinated attacks or require cooperative defensive strategies. Distributed denial-of-service (DDoS) attacks leverage many attacker-controlled devices to overwhelm legitimate services. While DDoS attacks may only target a single entity, from inundating a large company [86] to extorting ransom from gambling websites [24], defensive responses may require help from multiple actors. For example, Ioannidis and Bellovin devise a 'pushback' mechanism for countering DDoS attacks where intermediate routers locally rate-limit attack traffic and request upstream routers to cooperatively take similar action [80]. Notably, this mechanism has not been adopted by the industry precisely because it requires cooperation.

In wireless sensor networks (WSNs), an adversary may compromise the key material from several devices. One option is for an adversary to carry out key-harvesting attacks by compromising enough nodes to increase the chance of reusing keys to eavesdrop other nodes' communications. We describe additional attacks on WSNs in Chapter 2.

The economics literature has a rich discussion of the challenges of cooperation among self-interested principals. For instance, game theory is a tool for modeling strategic behavior where an individual's best response depends on the actions taken by others [59]. Many computing resources, from the provision of bandwidth to overall system reliability, exhibit characteristics of public goods, where consumption by one actor does not limit others and exclusion from consuming the good is impossible. Unfortunately, one common outcome for public goods collected from many sources is the 'free-riding' problem, where participants consume resources without contributing back [73].

Recently, the economics of information security has become a thriving and fast-moving discipline [12, 15]. Economics helps explain why security mechanisms fail and identify ways to strengthen applications requiring cooperation.

Asymmetric information – where one party to a transaction has better information than the other – can be a strong impediment to effective security. The study of this subject was initiated by George Akerlof's Nobel-Prize-winning paper on the 'market for lemons' [5], in which he imagined a town with 50 good used cars for sale (worth $2000 each), along with 50 'lemons' (worth $1000 each). The sellers know the difference but the buyers do not, with the result that the market price ends up near $1000. A lemons market also

affects some security products and services, as their effectiveness is difficult for consumers to ascertain correctly. The consumers refuse to pay a premium for quality they cannot assess, so products and services tend to be of poor quality.

The public has inadequate information about the relative effectiveness of the many security products and services available. Publishing quantitative metrics to a wider audience is essential for reducing information asymmetries. Yet for many years there has been a general lack of adequate statistics on information security – available data are insufficient, fragmented and incomparable [61].

Security breach disclosure legislation is one promising effort to improve information security measurement. The first such law to be enacted in the United States was California's A.B.700 in September 2002 [31]. It applies to public and private entities that conduct business in California and requires them to notify affected individuals when personal data under their control has been acquired by an unauthorized person. The law was intended to ensure that individuals are given the opportunity to take appropriate steps to protect their interests following data theft, such as putting a 'lock' on their file at credit agencies. It was also intended to motivate companies holding personal data to take steps to keep it secure. Indeed, Acquisti *et al.* [1] found a statistically significant negative impact on stock prices following a breach disclosure. Breach disclosure laws have also had the positive effect of contributing valuable data on security incidents to the public domain.

Often, overall security levels depend on the efforts of many interdependent principals. Hirshleifer told the story of *Anarchia*, an island whose flood defenses were constructed by individual families and whose defense depended on the weakest link, that is, the laziest family. He then compared this to a city whose protection against missile attack depended on the single best intercepting shot [74]. Varian extended this analogy to three cases of interest for the dependability of information systems – where performance depends on the minimum effort, the best effort, or the sum-of-efforts [157]. Program correctness can depend on minimum effort (the most careless programmer introducing a vulnerability) while software validation and vulnerability testing may depend on the total of everyone's efforts. Varian's analysis predicts that the principals who stand to gain most will carry out most of the effort, while others will free-ride.

This work inspired other researchers to consider interdependent risk. A recent influential model by Kunreuther and Heal notes that an individual taking protective measures creates positive externalities for others that in turn may discourage them from investment [91]. This insight has implications far beyond information security. The decision by one apartment owner to install a sprinkler system will decrease his neighbors' fire risk and make them less likely to do the same; airlines may decide not to screen luggage transferred from other carriers who are believed to be careful with security; and people thinking of vaccinating their children may choose to free-ride off the herd immunity instead. In each case, several widely varying equilibria are possible, from complete adoption to total

refusal, depending on the levels of coordination between principals.

Given interdependence between principals on many information security challenges, cooperation via information sharing has been recognized as an important way forward. Yet there is a fear that firms might free-ride off the security expenditures of other firms by only 'consuming' shared security information and never providing any data of their own [69]. Even the threat of such free-riding can stymie sharing. Nonetheless, there can also be positive economic incentives for sharing security information. Gal-Or and Ghose developed a model where sharing works by encouraging additional security investment [60]. Where there is a lack of industry awareness to threats, sharing information can certainly foster broader investment to the benefit of security service providers.

Information security practitioners traditionally have assumed two types of user: honest ones who always behave as directed, and malicious ones intent on wreaking havoc at any cost. But systems are often undermined by *strategic* users, who act out of self-interest rather than malice. Many file-sharing systems suffer from free-riding, where users download files without uploading their own. This is perfectly rational behavior, given that upload bandwidth is typically more scarce and file uploaders are at higher risk of getting sued. The cumulative effect is degraded performance. Several network protocols may be exploited by selfish users at the expense of system-wide performance. In TCP, the protocol used to transmit most Internet data, Akella *et al.* find that selfish provision of congestion control mechanisms can lead to suboptimal performance [4]. Buttyán and Hubaux [30] categorize threats to wireless network security as malicious or selfish. Using game theory, they develop several models of selfish behavior in wireless networks, from forwarding packets to sharing spectrum.

Researchers have used game theory to study the negative effects of selfish behavior on systems more generally. Koutsoupias and Papadimitriou termed the 'price of anarchy' as the ratio of the utilities of the worst-case Nash equilibrium to the social optimum [89]. Nash equilibria arise when participants act in their own interest without considering the impact on others. The social optimum, by contrast, is the course of action that maximizes the utility for all participants. The price of anarchy compares these outcomes. When the selfish action falls far short of the best case, the price of anarchy is large.

The price of anarchy has become a standard measurement of the inefficiency of selfish behavior in computer networks. Roughgarden and Tardos studied selfish routing in a congested network, comparing congestion levels in a network where users choose the shortest path available to congestion when a network planner chooses paths to maximize flow [139]. Other topics hindered by selfish activity include network creation, where users decide whether to create costly links to shorten paths or free-ride over longer, indirect connections [55, 16]; wireless spectrum sharing, where service providers compete to acquire channels from access points [71]; and computer virus inoculation, where users incur a high cost for inoculating themselves and the benefits accrue to unprotected nodes [21].

To account for user self-interest, computer scientists have proposed several mechanisms with an informal notion of 'fairness' in mind. To address spam, Dwork and Naor propose a non-cooperative defense mechanism – attaching to emails a 'proof-of-work' that is easy to do for a few emails but impractical for a flood [53]. Laurie and Clayton criticize 'proof-of-work' schemes, demonstrating that the additional burden may be cumbersome for many legitimate users while spam senders could carry out a cooperative attack using botnets to perform the computations [95]. Liu argues that proof-of-work schemes could be made viable by combining them with other measures of trust such as reputation systems [100].

Reputation systems have been widely proposed to overcome free-riding in peer-to-peer networks. The best-known fielded example may be feedback on eBay's online auctions. Dellarocas argues that leniency in the feedback mechanism (only 1% of ratings are negative) encourages stability in the marketplace [45]. Serjantov and Anderson use social choice theory to recommend improvements to reputation system proposals [145]. Feldman *et al.* model such systems as an iterated prisoner's dilemma game, where users in each round alternate between roles as client and server [57]. Recently, researchers have begun to consider more formally how to construct fair systems using mechanism design.

The goal of algorithmic mechanism design is to build network protocols and interfaces that are 'strategy-proof': that is, designed so that no one can gain by cheating [126]. Designing bad behavior out of systems may be cheaper than policing it afterward. Some promising initial results look at mechanism design and protocols. Feigenbaum *et al.* show how combinatorial auction techniques can be used to provide distributed strategy-proof routing mechanisms [56]. Schneidman *et al.* compare the incentive mechanisms in BitTorrent, a popular peer-to-peer file-sharing application, to theoretical guarantees of faithfulness [150].

Another example of how defense must be implicitly cooperative is the impact of network effects. Katz and Shapiro analyzed how network externalities influenced the adoption of technology, following an S-shaped adoption curve in which slow early adoption gives way to rapid deployment after the number of users reaches some critical mass [85]. Network effects can influence the initial deployment of security technology, whose benefit often depends on the number of users who adopt it. If the cost exceeds the benefit for the earliest adopters, then technologies are unlikely to ever be deployed. Ozment and Schechter analyzed different approaches for overcoming such bootstrapping problems [129]. A number of core Internet protocols, such as DNS and routing, are widely viewed as insecure. Better protocols exist (e.g., DNSSEC, S-BGP) but have seen limited adoption. Two widely-deployed security protocols, SSH and IPsec, both overcame the bootstrapping problem by providing significant internal benefits to adopting firms, with the result that protocols could be adopted one firm at a time, instead of moving all at once. In many cases, deploying collective defenses requires strategic design.

1.2 Structure and contribution of this thesis

This thesis is organized around two case studies of distributed network applications. The first case study, that of decentralized wireless networks, examines an emerging application. It identifies attacks that exploit cooperation and proposes defensive mechanisms for collaboratively revoking the credentials of errant devices. The second case study, that of phishing attacks on the Internet, empirically examines real-world attack and defense for one of today's main threats. It finds evidence of cooperation among attackers and non-cooperation among defenders, and then estimates the costs imposed by the attacks. Thus, the second case study empirically validates the importance of cooperation in attack and defense generally by studying ongoing attacks, which can only be done for the first case study through theoretical analysis and simulation.

The contribution of this thesis is to address several key gaps in the literature of distributed network security. First, we identify new cooperative attacks for wireless networks, while we empirically observe cooperative phishing attacks and demonstrate their superior effectiveness. This contributes to the understanding of how cooperative attacks work, in both theory and practice. Second, we propose and evaluate novel cooperative defense strategies in the context of revocation for wireless networks. Finally, we contribute an empirical analysis of cooperative attack and defense for an important class of Internet attacks. We demonstrate a lack of cooperation between defenders and quantify its impact, and we evaluate a voting-based online decision mechanism in the context of phishing.

Chapters 2–5 address cooperation in decentralized wireless networks, the first case study. Chapter 2 describes the operation of 'ad-hoc' networks generically and wireless sensor networks (WSNs) in particular. Sensor nodes have restricted capability in terms of computational power and storage, and each device commands limited authority. Tasks are distributed amongst member nodes which cooperate to provide services and reach decisions. Nodes are pre-loaded with a number of secret keys used for communication. Devices are deployed in unguarded environments without tamper protection, so node subversion is a very real threat that must be tackled. Dealing with errant nodes – either faulty or malicious – requires nodes to cooperatively detect misbehavior, decide on a course of action, and implement punishment.

Chapter 3 outlines two classes of attacks on the key-distribution schemes for WSNs. In the first attack, an adversary collects pre-assigned keys and swaps them between compromised nodes to create fake communication channels. The second attack describes a way to defeat revocation mechanisms by rejoining ejected nodes to the network via colluding intermediaries. The chapter concludes by devising countermeasures. Chapter 4 proposes new decentralized decision mechanisms for revoking the credentials of an errant node. The primary new mechanism, called suicide, is lightweight and simple: a good node, on perceiving another node to be misbehaving, simply declares both of them to be dead.

Other nodes thereafter ignore them both. Suicide exhibits favorable properties compared to traditional voting-based mechanisms for WSNs.

Chapter 5 compares suicide to voting-based removal for a new application, vehicular networks. Compared to WSNs, the radio-equipped devices found in vehicular networks are more capable computationally, but they are also highly mobile and characterized by short-lived interactions. We conduct simulations to identify trade-offs between adapted suicide and voting strategies.

Chapters 6–9 discuss the second case study, an empirical examination of phishing attack and defense on the Internet. Chapter 6 describes how phishing attacks work. In particular, for a large portion of phishing (the so-called 'rock-phish' gang), we find evidence of a cooperative attacker exploiting uncoordinated defenders. We then outline the current state of phishing defense, where companies continuously detect new attacks as they appear and ask the appropriate authorities at ISPs and domain registrars to remove the offending material. Most of the research in these chapters is data-driven, so we conclude Chapter 6 by outlining the data sources and collection methodologies used in Chapters 7–9.

Many different attackers carry out phishing attacks, but even more entities must defend against the attacks – banks, companies specializing in website cleanup, ISPs where the phishing websites are hosted, and registrars when dodgy domains are used. Unsurprisingly, the defense against phishing attacks is uncoordinated and depends on many factors. In Chapter 7 we show one consequence of uncoordinated defense is a long-tailed distribution of phishing website lifetimes. Most phishing websites are dealt with quickly, but a substantial minority go unnoticed and escape removal for many days or weeks.

In Chapter 8 we study different sources of phishing websites. We empirically demonstrate that take-down companies are not sharing information about phishing websites, which causes these overlooked sites to remain for long periods. We quantify the cost of the resulting imprudence, finding that non-cooperation puts at risk an extra few hundred million dollars per annum.

In Chapter 9 we study a community-based effort to verify phishing URLs. Anyone may submit URLs of suspected phishing websites, and may vote on the accuracy of the submissions from others. Hence, the chapter provides an empirical complement to the comparisons by analysis and simulation of the voting and suicide decision mechanisms for ad-hoc networks presented in Chapters 4 and 5. We compare the voting mechanism to unilateral verification performed by a specialist company, and find voting to be significantly slower. We also identify ways in which the voting could be manipulated by an attacker wishing to incorrectly assess malicious phishing websites.

Finally, in Chapter 10 we discuss opportunities for future work and conclude.

Chapter 2

Decentralized wireless networks

The last ten years have seen the invention and deployment of a range of systems which organize themselves out of a collection of nodes in order to perform some task. Communications technologies such as WiFi [42], Bluetooth [27] and Homeplug [96] support short-range networking of disparate devices in home and office environments, and may allow larger networks to be assembled opportunistically. Sensor networks [6] then came along – networks assembled from large numbers of low-cost nodes that could be scattered into an area of interest to perform tasks such as surveillance or environmental monitoring. We describe such communications strategies generically as 'ad-hoc networking'.

These new technologies discard management by central authority in favor of decentralized mechanisms that improve efficiency and robustness. Here, tasks are distributed amongst member nodes which cooperate to provide services and reach decisions. Another common feature of wireless ad-hoc networks is their support for mobile devices. Devices move while remaining connected to the network, breaking links with old neighbors and establishing fresh links with new devices. Finally, resource constraints limit the tools available to the protocol designer: at most a minority of nodes have the capability to create digital signatures or store large amounts of data; symmetric cryptography is preferred for establishing and maintaining key material.

There are various threats to ad-hoc networks, of which the most interesting and important is probably *node subversion*. A node in a military sensor network may be reverse-engineered by the enemy, and replaced by a malicious node that knows its key material and can thus impersonate it. Subverted nodes can perform a number of attacks on the network, for example, decrypting messages, injecting false data and manipulating decentralized operations such as voting. Thus they must be identified and removed quickly.

In this thesis, I study two 'instantiations' of ad-hoc networks: wireless sensor networks (WSNs) and vehicular networks. WSNs are comprised of severely resource-constrained devices. As a result, researchers have designed symmetric key-management schemes that minimize storage, computation and communication costs. In Chapter 3, I describe several

attacks on key pre-distribution schemes that have been over-optimized and provide countermeasures that work even under severe constraints. I then propose several strategies for good devices to protect themselves from subverted nodes through cooperation, first for WSNs (Chapter 4) and then for vehicular networks (Chapter 5). Vehicular networks present different challenges. The devices are more capable, but they are also highly mobile, interactions can be short-lived, and bad devices can wreak havoc much more quickly. This chapter sets out the ground rules for decentralized wireless networks using WSNs as an example. Chapter 5 explains which circumstances change for vehicular networks.

2.1 System model

WSNs are comprised of a large collection of low-cost devices that cooperate to form a communications network. Devices, or *nodes*, are labeled as $A, B, C, \ldots \in N, |N| = n$. Node resources are severely limited in terms of processing power, storage, and battery life. This means expensive cryptographic operations and communications must be minimized. Nodes have a limited communication radius; each node A can reach a set of neighboring nodes $\mathcal{N}(A)$ within A's radio range. The standard topological assumption for WSNs is that nodes are geographically positioned randomly across a space so that each can contact n' other nodes on average. Typical values for n' range from 40 to 60. There may also be one or more trusted base stations S, more capable devices that can be leveraged by the nodes to perform additional security tasks. We assume that nodes adopt a viable medium access control (MAC) protocol (e.g., 802.11 [42], S-MAC [168]) that addresses contention in the wireless channel, thereby overcoming complications such as the hidden-terminal problem [155].

There are four basic events in the life cycle of a distributed, wireless, sensor network: *pre-deployment*, *initialization*, *operation* and *revocation*. In pre-deployment, the network owner programs nodes with keys and authentication values. This is regarded as a secure operation occurring away from the attacker under the owner's control. Nodes are then deployed into the environment where attackers may be present and initialized by establishing keys with their neighbors. When nodes are mobile, key setup is ongoing as they establish links with new neighbors and break links with old ones. At any stage, one or more nodes may find another node misbehaving, prompting a decision mechanism to determine whether the node should be removed from the system. Revocation invalidates any credentials shared between the revoked node and honest nodes.

The notation used in this chapter and throughout the rest of the thesis is summarized in Table 2.1.

Symbol	Meaning
A, B, C	normal nodes
S	base station node
M_i	malicious node i
N	set of nodes in network
n	network size, $n = \|N\|$
$\mathcal{A}$	set of malicious nodes
n'	expected number of neighbor nodes in radio range
$\mathcal{N}(A)$	set of neighbors of node A
$\mathcal{V}_A$	set of voting members for node A
v	size of voting member set, $v = \|\mathcal{V}_A\|$
d	threshold of votes required for decision
p	probability of two nodes sharing a pairwise key
$\mathcal{U}(A)$	set of usable pairwise keys for node A
$h()$	pseudo-random hash function
k_{AB}	key shared between nodes A and B
P	large pool P of l keys
m	size of key ring

Table 2.1: Notation.

2.1.1 Threat model

The attacker can compromise a small minority of nodes $\mathcal{A} = \{M_1, M_2, \ldots, M_i\}$, since devices may be unprotected and deployed into hostile environments. Good nodes adhere to their programmed strategy including algorithms for routing and revocation. A bad node can communicate with any other node, good or bad. Bad nodes may have access to the keys of all other bad nodes, whom they can therefore impersonate if they wish. They do not execute the authorized software and thus do not necessarily follow protocols to identify misbehavior, revoke other bad nodes, vote honestly or delete keys shared with revoked nodes. Notably, we do not assume active node compromise is immediately detected. In fact, node compromise may never be detected.

In particular, we consider three threat models in order from most to least restrictive:

T.0 We assume a global passive adversary upon deployment. Here we allow the adversary to actively compromise a small minority of nodes during initialization. This threat model is adopted by most key-distribution schemes for sensor networks [54, 38, 52, 101, 37]. Unless mentioned otherwise, threat model **T.0** applies.

T.1 Again, we assume a global passive adversary upon deployment. However, no nodes are actively compromised until after the initialization phase, when a small minority may be targeted.

T.2 A third threat model is even more relaxed. The opponent is assumed to be capable of monitoring at most a small minority of communications during the initialization phase [175, 14]. After initialization, a small minority may be actively compromised. Easing the assumption of a global passive adversary to a passive one during initialization means that no, or fewer, keys need to be pre-loaded; instead, nodes set up path keys with neighbors immediately after deployment.

2.2 Key-management infrastructure

The key-management infrastructure plays an important role in establishing trust between devices in a WSN. Beyond enabling secure communications between devices, keys demonstrate membership within a network and enable participation in protocols for removing bad nodes from the network.

Keys and associated authentication values are computed by the network owner and loaded onto devices during the pre-deployment phase – key generation and exchange is too expensive for devices to carry out directly. If public-key cryptography is used [47], then the device must store the public and private key pair, plus a certificate of the public key. Certificates are signed statements binding an identity or role to a key. They can be signed by a universally-recognized certification authority (CA), so any certificate can be verified using the CA's public key, which is stored in all nodes. Nodes demonstrate network membership by using the private key corresponding to the certificate issued by the CA.

However, public-key cryptography is computationally expensive, so its use in WSNs must be extremely limited, if allowed at all. While several researchers have demonstrated the feasibility of public-key cryptography on sensor nodes [161, 70, 66, 25, 156] and argued that it enables cost-saving protocol improvements [65], symmetric key cryptography remains indispensable for most WSN applications. In this section we describe symmetric key-management strategies for WSNs, followed by the ensuing complications and attacks.

2.2.1 Key pre-distribution schemes

The simplest architecture is a single shared key known to all nodes. This scheme is vulnerable to the compromise of a single node, and revocation is impossible. At the other extreme is the *complete pairwise scheme*, where every node stores a unique pairwise key for each of the $n-1$ other nodes in the network. Here all nodes can confidentially communicate with each other, and any individual node can be revoked. However, the scheme is infeasible when considering large networks of low-cost nodes with limited storage space. We now review a number of proposals seeking a middle ground where a limited number of keys are assigned to nodes while maintaining a high likelihood of node confidentiality.

Eschenauer and Gligor [54] propose two related techniques for reducing the number of keys pre-loaded onto nodes: *key pools* and *random key assignment*. Here each node is randomly assigned m keys from a large pool P of $|P|$ keys (where $m << n$ and $|P| >> n$). Nodes determine which keys are shared by querying each other for the identifiers of keys held. These *link keys* are used to secure and authenticate messages between nodes. Using results from random graph theory to identify suitable choices of $|P|$ and m, the network is probabilistically guaranteed to be connected, with nodes sharing a link key with a sufficient number of neighbors in communication range. This pooling mechanism and random key assignment have inspired several extensions and variations. Chan, Perrig and Song generalize the scheme to require q shared secrets to establish a link key [38]. Du *et al.* [52] and Liu and Ning [101] independently propose creating a large polynomial pool to offer threshold secrecy.

Eschenauer and Gligor randomly assign keys to nodes; thus the only way for nodes to determine whether they share keys is to exchange lists of key identifiers, or by challenge-response. Zhu *et al.* [176] propose a deterministic algorithm that calculates the key identifiers known to a node from the node identifier. This increases efficiency, but it also helps an attacker obtain desired keys by targeting nodes for compromise. To counter this, Di Pietro *et al.* [48] describes a method for nodes to check whether they share keys without revealing all keys held by every node.

While undoubtedly reducing storage requirements, key pools also introduce a new set of security challenges. First, it is hard to authenticate identity based on the keys held by a node, since several nodes may legitimately possess the same keys. Second, pool keys make it hard to revoke a misbehaving node's keys without negatively impacting legitimate nodes. Removing compromised keys is onerous since many nodes across the network could be assigned some keys in common with a revoked node; yet removing too many keys could deplete the key pool, causing inadvertent denial-of-service attacks. Finally, pool keys make harvesting attacks attractive, where an attacker compromises enough nodes to increase the chance of reusing keys to eavesdrop other nodes' communications.

Chan, Perrig and Song propose a *random pairwise* scheme [38] as an alternative to key pools combining aspects of the complete pairwise scheme with the storage-saving random distribution approach of Eschenauer and Gligor [54]. Nodes are pre-loaded with pairwise unique keys, but rather than allocate $n - 1$ keys per node, a fraction of the keys are randomly assigned with probability p. Typically, p ranges from 0.2 to 0.4; therefore, each node stores a random set of $n * p$ pairwise keys and identifiers rather than $n - 1$ keys.

Pre-distributing pairwise unique keys prevents a key-harvesting attacker from compromising the confidentiality of any pre-assigned key shared between uncompromised nodes. It also enables mutual authentication between nodes sharing a key. This forms the basis of a revocation scheme whose details we describe in Section 2.3.2. One disadvantage of the random pairwise scheme is increased storage cost: nodes are pre-loaded with keys

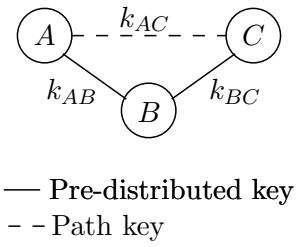


Figure 2.1: Path key k_{AC} established between A and C using B as an intermediary.

totaling a significant fraction of n (generally $\frac{1}{5}$ to $\frac{1}{3}$).

In every pre-distribution scheme, nodes are pre-loaded with more keys than they can use. This is because neighboring nodes cannot be predicted before deployment. The set of usable keys, $\mathcal{U}(A)$, for node A is determined by nodes (i) being in proximity and (ii) sharing the right keys. For example, a pairwise key k_{AB} is added to $\mathcal{U}(A)$ only if one of node A's neighbors $B \in \mathcal{N}(A)$ also holds k_{AB}.

2.2.2 Path-key establishment

In all of the schemes outlined above, there must exist neighboring nodes that are not pre-assigned a common key but wish to communicate. These nodes must discover a path between each other using a number of intermediate nodes, where each hop is secured by a link key. One of the nodes chooses a new *path key* and sends it to the other node, encrypted using link keys between intermediaries.[1] For instance, suppose A wishes to communicate with C, but they have not been pre-assigned a common key (Figure 2.1). Suppose also that A shares a pre-assigned key with B, and that B shares a pre-assigned key with C. A can create a path key k_{AC} and send it to C as follows: $A \rightarrow B : \{k_{AC}\}_{k_{AB}}$, $B \rightarrow C : \{k_{AC}\}_{k_{BC}}$.

Intermediate nodes are selected for setting up path keys in two ways. In *random* path-key establishment, nodes discover paths to other nodes using locally broadcast messages. The average path length depends on the scheme used. In random pool-key deployment with plausible values for $|P|$, m and n, path keys to most neighboring nodes can be established within three hops [54]. Path keys to distant nodes are more expensive, however, requiring an average of eleven link keys for the simulations in [54]. Random path-key establishment is simple but has relatively high communication costs. Schemes using random path-key establishment include [54, 38, 52, 101].

Alternatively, in *deterministic* path-key establishment, link keys are assigned based on a node's identifier so that nodes can unilaterally select the intermediaries used to set up path

[1] To clarify terminology, *link keys* are pre-assigned keys or derived directly from them. *Path keys* are established between devices not directly sharing link keys.

keys. Given a node identifier as input, a pseudo-random function returns the identifiers of keys held by that node. A node checks the key identifiers held by some of its randomly selected neighbors to see if any match the keys held by the destination node. The process can repeat using several intermediaries until a path to the destination is found. Nodes do not need to know anything about the network's topology – if the intermediaries on the selected path are not within communication range, alternative paths are selected until the path completes. Deterministic path-key establishment reduces the communication cost of searching for suitable intermediate nodes while slightly raising the computational costs. Schemes using deterministic path-key establishment include [37, 176].

Path-key establishment is vulnerable to malicious intermediaries, since only link-level encryption is used to establish an end-to-end key. All nodes along the path learn the path key agreed upon by the end points. Consequently, several papers explore multi-path-key reinforcement for strengthening path keys [14, 38, 176].

2.2.3 Attacks on WSN key management

The keys assigned to nodes are effectively also their identities. As a result, the uniqueness of a node's identity is tied to the secrecy of the keys it has been assigned. A message encrypted under a symmetric key assigned to a group of nodes could have originated from any node in the group. Encrypting under a pairwise unique key, by contrast, unambiguously demonstrates a node's identity to the other node that shares the key. Therefore, attacks on identity are important, and they are usually carried out as attacks on preloaded keys.

Sybil identities [49], where a malicious node pretends to be multiple distinct nodes in the network, can facilitate attacks against routing, voting, misbehavior detection, distributed storage, resource allocation and data aggregation in sensor networks. Key pool schemes, for example, are especially vulnerable to Sybil attacks: an attacker can create many fake identities from a few known pool keys since many nodes would be expected to hold these keys.

A Sybil variant is node replication [133], where many copies of a subverted node are introduced. Node-replication attacks are used in preference to Sybils whenever the network owner has managed to restrict the number of recognized keys operational in the network.

Chapter 3 presents two new attacks on key management for WSNs. In the first, attacker-controlled devices use each other's pairwise keys to overwhelm good nodes with bad connections. In the second, path keys are used to carry out Sybil attacks and let nodes rejoin the network after being revoked. In both cases, the adversary leverages cooperation between compromised devices to make the attack more effective.

2.3 Defending against bad nodes

Three stages occur when revoking a bad node: detecting misbehavior, deciding whether to revoke, and implementing punishment.

2.3.1 Detection mechanisms

The first step in defending against bad nodes is detecting them. Detection can be done by passively monitoring or actively testing neighboring nodes. Examples of passive monitoring include the wireless monitoring schemes Watchdog [104] and CONFIDANT [29]. In these schemes, nodes promiscuously listen to their neighbors' routing actions to confirm that appropriate behavior is taken, e.g., that messages are not modified when passed onwards.

Several active monitoring mechanisms have been proposed to verify node claims. For example, distance bounding as part of a secure localization protocol can confirm a node's position in the network [34]. Newsome *et al.* propose a Sybil detection mechanism where honest nodes challenge each other for the expected pre-loaded keys associated with claimed identities [124]. Two nodes that share a common key can directly challenge each other to prove that they have knowledge of the key. As more nodes challenge a given node in this way, confidence increases that this node is not a Sybil identity. Parno *et al.* [133] detect node replication by requiring all nodes to periodically broadcast signed location claims; good nodes then keep a record of claimed locations and check for conflicting claims.

The trouble with detection mechanisms is that they rarely produce evidence that is universally non-repudiable. When such mechanisms do exist (e.g., geographic packet leashes for detecting wormholes [77] and Parno *et al.*'s node-replication-detection technique [124]), they require extensive use of costly signed messages. Furthermore, situations where a bad node is forced into self-incrimination are limited. It is hard to incriminate a node that drops occasional messages since messages vanish for many reasons unconnected with malice. More typically, evidence is gathered which is non-repudiable to a single party. For example, a message authentication code (MAC) generated with a key shared between two nodes guarantees authenticity to the other node. Detection mechanisms of this type include temporal packet leashes [77], Newsome *et al.*'s Sybil attack detection by querying for possessed keys [124] and distance-bounding protocols [28, 72, 33]. Still other mechanisms rely on evidence that is entirely repudiable (e.g., the wireless monitoring scheme Watchdog [104]). Repudiable evidence enables bad nodes to falsely accuse good nodes. Hence, it would be foolish to design a mechanism for removing bad nodes that ejects any node accused of misbehavior following a single unsubstantiated claim of impropriety.

A second problem with detection mechanisms is that untrusted nodes, not central authorities, are often in the best position to detect misbehavior. If node A accuses node B of

making inconsistent statements about its location and B denies making them, a trusted base station can only determine that one of them is misbehaving. Hence cooperation between untrusted devices is necessary. These problems mean that a *decision mechanism* is required where nodes collectively agree to punish bad nodes.

2.3.2 Decision mechanisms

Decision mechanisms can be left to a central authority such as a base station only when the authority is capable of detecting the misbehavior directly, or to the devices themselves when the evidence used to reach a decision is universally non-repudiable. When such conditions are met, the centralized scheme described by Eschenauer and Gligor [54], where the base station determines which keys are tied to a compromised node and instructs all nodes holding these keys to delete them, is sufficient.

Otherwise, untrusted nodes must be relied upon to cooperate and make the decision. Existing proposals for collective decision-making have been voting-based. Threshold voting is a natural choice to implement revocation as it conceptually distributes the decision-making process while considering the observations of others. In a *blackballing* scheme, once the number of negative votes cast exceeds a specified threshold, then the target node is deemed malicious and revoked from the network. Liu *et al.* leave the task of tallying votes from devices and issuing the revocation order to the base station [103]. Several others have proposed using threshold cryptography to split the task of signing a revocation order between a threshold of users [88, 87, 174].

Reputation systems collect historical ratings of interactions and can be thought of as weighted voting schemes [145]. When nodes only rarely interact, a collective shared history could identify bad nodes more efficiently. Proposed decentralized reputation systems for decentralized wireless networks include [29, 109, 62]. Reputation systems struggle whenever systems are too large, mobile or transiently connected for nodes to anticipate interacting with the same nodes repeatedly. In these circumstances, stored history does not reliably inform new node interactions; misbehaving nodes can therefore avoid detection [114]. Furthermore, keeping node histories incurs significant storage and computational overhead. Unweighted voting is often a simpler, more effective solution.

In [38], Chan, Perrig and Song propose a distributed revocation mechanism for the random pairwise scheme, where nodes sharing pre-assigned pairwise keys vote to remove a node. Their blackballing scheme is extended and generalized in [36]. Here, each node B that shares a pairwise key with A is assigned to the set of *voting members of* A, $\mathcal{V}_A$. Nodes in $\mathcal{V}_A$ can vote for each other's removal. Every node A is assigned a unique revocation secret rev_A, which is divided into secret shares according to a (d, v) threshold scheme [147]. Shares $\mathsf{rev}_{A,B}$ (indicating B's vote against A) are given to every $B \in \mathcal{V}_A$ along with an authentication value for the revocation secret, $h^2(\mathsf{rev}_A)$. Nodes vote for another's removal

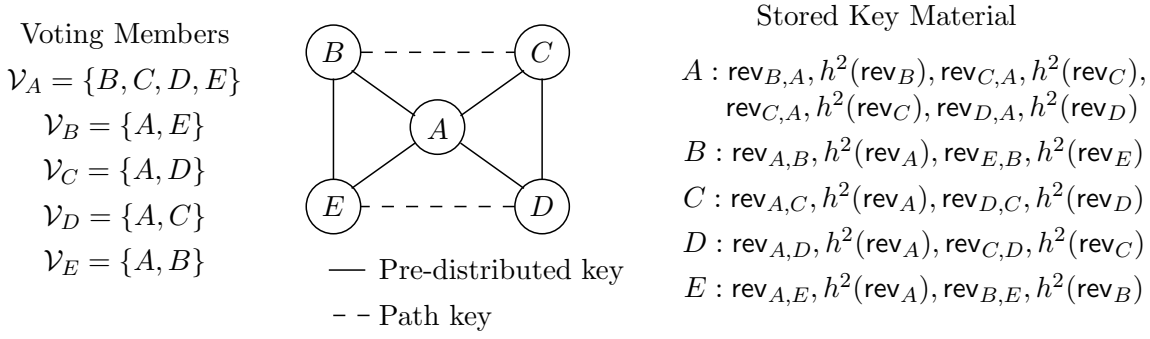


Figure 2.2: Chan *et al.*'s blackballing revocation mechanism.

by revealing their share. If d shares are revealed, then rev_A can be reconstructed and $h(\mathsf{rev}_A)$ is broadcast across the network. Every node $B \in \mathcal{V}_A$ deletes its key shared with A upon verifying the broadcast. Figure 2.2 shows the voting members and key shares assigned to nodes for an example network.

One problem with voting by revealing secret shares is that cast votes are permanent: a slow trickle of votes against a node over its lifetime is equivalent to a burst in a short period. To avoid stale votes, Chan *et al.* create T revocation sessions each with a unique revocation secret $\mathsf{rev}_{A,i}, i \in \{1, \ldots, T\}$ and associated shares $\mathsf{rev}_{A,i,B}$; thus a revealed share only counts as a vote for a single period i.[2]

Chan *et al.*'s blackballing scheme also makes use of Merkle hash trees [108], whose construction we now review as they are used extensively in Chapters 3 and 4. Hash trees provide a space-efficient way to authenticate values using hash operations: n values can be universally verified if devices individually store just $\log n$ hashes. To authenticate values $x_1, \ldots, x_n$, a binary balanced tree is constructed with $h(x_i)$ as leaves. Each parent node is computed as the hash of its two concatenated child nodes. The root of the tree is distributed to all verifying parties, and any leaf x_i can be verified using the $\lceil \log_2 n \rceil - 1$ intermediate authentication values of sibling nodes along the path. We refer to these hashes as *path-authentication values*. For brevity, we write $\lceil \log_2 n \rceil - 1$ as $\log n$.

Consider the tree in Figure 2.3, which can be used to authenticate values x_1, x_2, x_3 and x_4. The path authentication values for x_1 are $h(x_2)$ and $h(h(x_3)||h(x_4))$. Given x_1, the root value and the path-authentication values, any device can verify x_1's authenticity by checking whether a computed root matches the stored root value.

In Chan *et al.*'s blackballing scheme, hash trees are used to authenticate the secret shares distributed for voting. Hash trees are constructed for every node $A \in N$ using the v secret shares of $\mathsf{rev}_{A,i}$, setting $\mathsf{rev}_{A,i,B}, \mathsf{rev}_{A,i,C}, \ldots$ as leaf preimages. Devices store root values for trees containing shares they need to verify. For instance, node A stores the vT roots of trees containing shares of $\mathsf{rev}_{B,i}$ $\forall B \in \mathcal{V}_A, i \in \{1, \ldots, T\}$. A must also store

[2]Note that for simplicity we used a single session for the example in Figure 2.2.

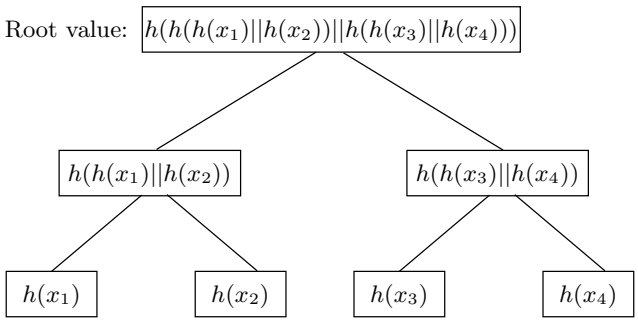


Figure 2.3: Merkle hash tree for preimages x_1, x_2, x_3 and x_4.

$\log v$ path-authentication values to prove the legitimacy of its own shares to the other voting members. Any time a device broadcasts its share it must also transmit path-authentication values so that the other voting members can authenticate the revealed share. For example, if A votes against B during time period 3, A must broadcast $\mathsf{rev}_{B,3,A}$ along with $\mathsf{rev}_{B,3,A}$'s path-authentication values. All nodes in $\mathcal{V}_B$ may then compare the computed root to their own copy of the root value for the tree containing $\mathsf{rev}_{B,3}$'s secret shares as leaf preimages.

To summarize, each node A is loaded with information to do the following:

1. **Vote against each node $B \in \mathcal{V}_A$:** Secret share $\mathsf{rev}_{B,i,A}$ $\forall B \in \mathcal{V}_A, i \in \{1, \ldots, T\}$ (storage cost vT)
2. **Prove to all $B \in \mathcal{V}_A$ that vote is valid[3]:** $\log v$ path-authentication values for each vote $\mathsf{rev}_{B,i,A}$ (storage cost $vT \log v$)
3. **Verify votes from others:** Hash tree roots $\forall B \in \mathcal{V}_A, i \in \{1, \ldots, T\}$ (storage cost vT)
4. **Verify revocation secrets for all $B \in \mathcal{V}_A$:** Authentication values for revocation secrets $h^2(\mathsf{rev}_{B,i})$ $\forall B \in \mathcal{V}_A, i \in \{i, \ldots, T\}$ (storage cost vT)

Any blackballing scheme must deal with a number of fundamental issues: which nodes are eligible to vote, how individual votes are verified, how votes are tallied and how the outcome of a vote is verified. They can also be slow, expensive and prone to manipulation. Chapter 4 proposes alternative decision mechanisms and evaluates their performance compared to blackballing.

2.3.3 Punishment mechanisms

Once a decision is reached, the bad node is punished. Typically, bad nodes are kept from interacting with good nodes by instructing every node to delete all keys shared with

[3]By 'valid', we mean the voting share is the one pre-assigned to the node.

the bad node [54, 38, 175]. Alternatively, nodes could be implicitly removed by routing around the bad node [104] or by maintaining a blacklist.

Simply deleting keys is insufficient if an attacker can reenter the network by other means. Chapter 3 describes an attack on Chan *et al.*'s blackballing mechanism where bad devices can rejoin the network, and fixes it in Section 3.5.

Chapter 3

Attacks on key-distribution schemes

In this chapter, we describe two classes of attacks on key-management mechanisms for WSNs. First, we describe a key-swapping collusion attack where attackers cooperate by sharing pairwise keys to create many fake communications channels. Although the use of pairwise keys eliminates the eavesdropping attack that afflicts other key pre-distribution mechanisms, nodes are still loaded with a large number of globally-applicable secrets. A collusive attacker can share its pre-assigned keys between compromised nodes, enabling each to present multiple 'authenticated' identities to neighboring nodes while escaping detection. We show that such an attacker can establish enough forged communication channels to outnumber legitimate ones using a relatively small group of colluding nodes.

Second, we identify weaknesses in mechanisms for the cooperative removal of keys held by bad devices. Existing revocation proposals fail to remove any path keys associated with a revoked node. We describe a number of resulting attacks which allow a revoked node to continue participating in a network. We then propose techniques for ensuring complete revocation: universal notification to remove keys set up with revoked nodes, path-key records to identify intermediaries that are later revoked, and blacklists to prevent unauthorized reentry via undetected malicious nodes. Path keys also undermine identity authentication, enabling Sybil attacks against random pairwise key pre-distribution.

Both attacks affect all key pre-distribution schemes described in Chapter 2. In fact, the attacks apply to any mechanism where fewer than complete pairwise keys are assigned.

3.1 Key-swapping collusion attack

Traditionally, the threat from node compromise is measured by its impact on confidentiality – whether secret keys shared between uncompromised nodes can be obtained. In contrast, we have expanded the threat model to include integrity and availability by considering the proportion of attacker-controlled communication channels. Notions of confidentiality are moot if an attacker commands the vast majority of transmission paths.

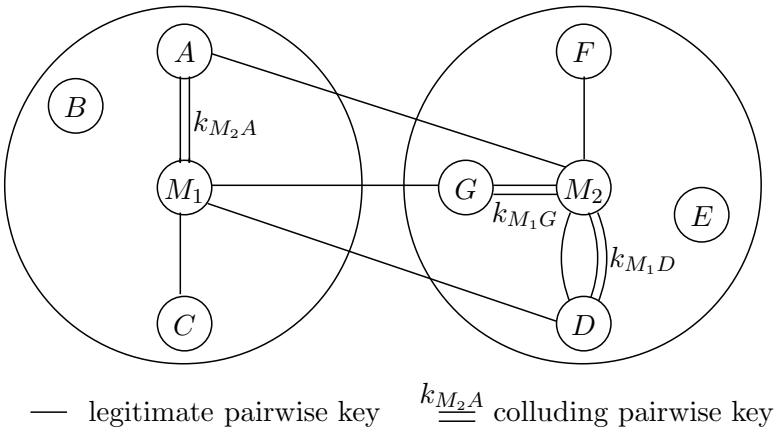


Figure 3.1: Example key-swapping collusion attack between attacker nodes M_1 and M_2.

We define a novel attack that exploits the combination of pre-loaded keys and localized interaction of sensor nodes. Consider two nodes controlled by the attacker, $M_1, M_2 \in \mathcal{A}$. If M_1 tells M_2 its secrets, then M_2 can masquerade as M_1 to all of M_2's neighbors that M_1 shares pairwise keys with, and vice versa. The keys from each subsequently obtained node can be reused by other attacker-controlled nodes, cascading the impact of compromise.

The attack is similar to a Sybil attack [49] in that single nodes present multiple identities; however, these identities are not randomly generated but instead are reused according to available pairwise keys. The attack also resembles a node-replication attack, where copies of a node are inserted into a network. However, the *key-swapping collusion attack* is distinguished from these other attacks in that attacker-controlled nodes pretend to be different nodes to different neighbors.

Consider the example sensor network in Figure 3.1. Node M_1's neighbors are A, B and C, while M_1 shares pairwise keys with C, D and G. Node M_2's neighbors are D, E, F and G, while M_1 shares keys with A, D and F. Thus, M_1 can legitimately communicate directly with node C, while node M_2 can communicate legitimately with nodes D and F. But if M_1 and M_2 collude to share each other's secrets, then M_1 can communicate with A by pretending to be M_2, and M_2 can pretend to be M_1 when communicating with D and G. Note that even though node M_2 can already communicate with D, colluding with M_1 enables M_2 to present multiple identities to D. Attacker nodes can communicate beyond radio range using the network's existing routing mechanism or an out-of-band channel.

Here are the sets of usable pairwise keys when M_1 and M_2 act independently and collude:

	Independence	**Collusion**
$\mathcal{U}(M_1)$	$\{k_{M_1C}\}$	$\{k_{M_1C}, k_{M_2A}\}$
$\mathcal{U}(M_2)$	$\{k_{M_2D}, k_{M_2F}\}$	$\{k_{M_2D}, k_{M_2F}, k_{M_1G}, k_{M_1D}\}$

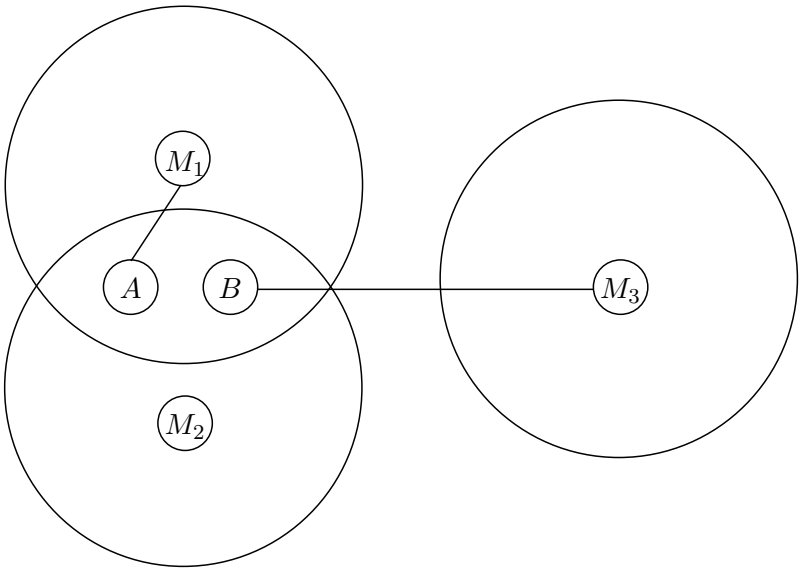


Figure 3.2: Overlap between attacker nodes M_1 and M_2.

As more nodes are compromised, overlap between node communication ranges must be taken into consideration. Overlap becomes unavoidable as the number of attacker-controlled nodes $|\mathcal{A}|$ approaches $\frac{n}{n'}$. Two colluding nodes gain nothing by both pretending to be the same node to a common neighbor. In Figure 3.2, nodes M_1 and M_2 can both masquerade as M_3 to node B, but only one of them should do so. Also, node M_2 achieves nothing by pretending to be M_1 to A, since A already shares a pairwise key with M_1.

The collusion attack utilizes a devastating combination of globally-applicable secrets and locally-communicating nodes. Pairwise keys can be used throughout the network, yet ordinary sensors can only communicate with the small fraction of nodes within radio range. An attacker can readily exploit this lack of coordination between nodes. The smaller the ratio $\frac{n'}{n}$ between average node neighborhood and the overall network size, the greater is the level of uncoordination between nodes. Likewise, as the average fraction p of pairwise keys stored by each node increases, each compromised node offers more potentially usable pairwise keys to the attacking node.

We have noted that the confidentiality of communications between uncompromised nodes is not affected by increasing node capture. However, the integrity and availability of interactions between uncompromised nodes are certainly threatened. For instance, attacker-controlled nodes increase their chances of partitioning the network or counteracting redundant routing whenever ordinary nodes believe they are dealing with many nodes instead of just one.

Clearly, authenticating nodes based upon possession of a particular pairwise key is also affected. For a blackballing scheme where votes are assigned according to pairwise keys (see Section 2.3.2), the attacker-controlled devices could vote to remove good devices.

If path keys are used to set up communication between neighboring nodes not sharing

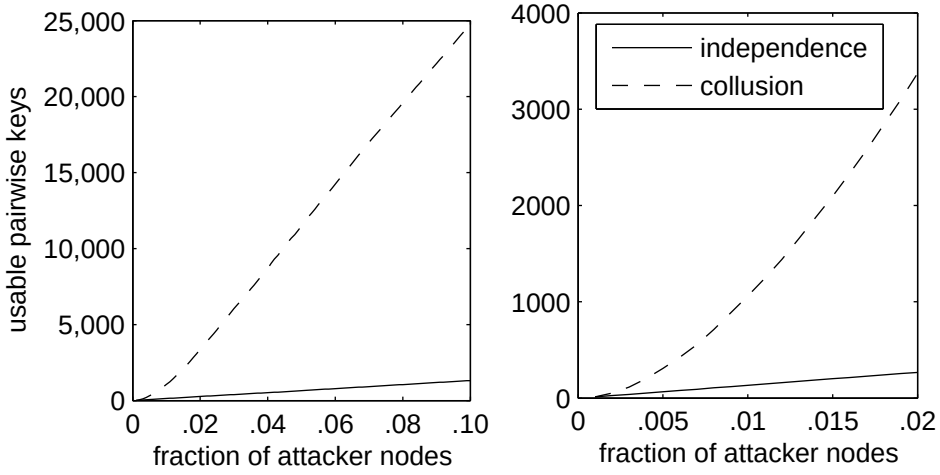


Figure 3.3: Total usable pairwise keys available to attacker-controlled nodes.

any pairwise keys, then the attacker could also create many fake channels using path keys. However, pre-assigned pairwise keys are regarded as 'more secure' and used in special operations like path-key establishment and voting for revocation. So the impact of creating extra identities using pairwise keys, rather than path keys, is greater.

3.2 Analysis

We simulated a sensor network comprised of nodes uniformly distributed over a plane, setting $n = 1000$, $n' = 60$, $p = .25$ and varied $|\mathcal{A}|$, averaging results from 20 rounds.[1]

To quantify the collusion attack's impact, we focus on the pairwise secret keys stored by each node. We first compare the number of usable pairwise keys available when the attacker-controlled nodes act independently versus when they collude. Second, we compare the number of pairwise keys available to the attacker relative to the number of legitimate usable keys available to the attacker's neighbors. This measure quantifies the level of network penetration achieved by the attacker.

3.2.1 Increased usable pairwise keys

Figure 3.3 compares the number of usable pairwise keys available to attacker-controlled nodes, $|\bigcup_{A\in\mathcal{A}} \mathcal{U}(A)|$ for increasing $|\mathcal{A}|$. An attacker can make use of a pairwise key if it is shared between nodes in communication range and it is not already in use within this range. Colluding attackers can access each other's pairwise keys; thus it is no surprise

[1]The graphs in Figures 3.3–3.5 appear smooth because they plot the average from 20 rounds of simulations.

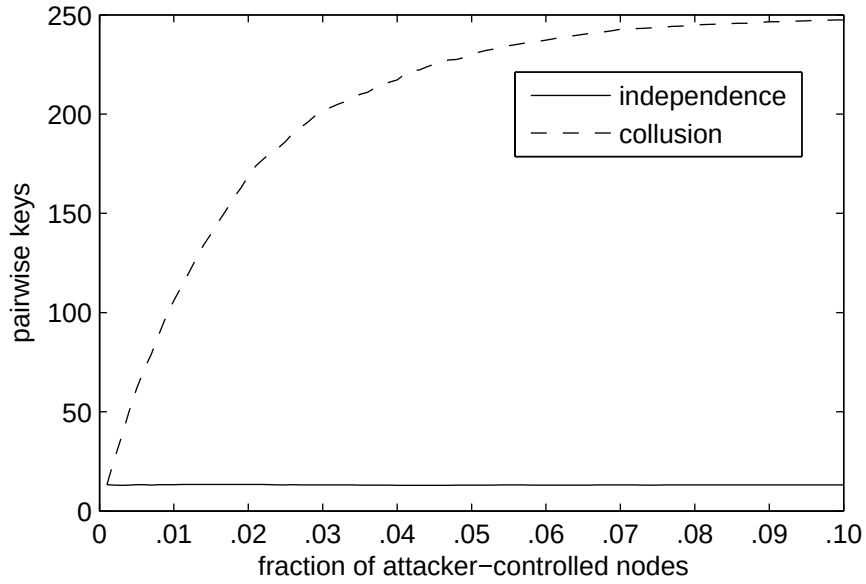


Figure 3.4: Average number of usable pairwise keys per attacker-controlled node.

that the figure indicates many more usable pairwise keys available to attacker-controlled nodes when they share secrets.

We now devise formulas for the total expected number of keys available to attackers and verify the formulas' accuracy through simulation. If an attacker does not share keys, each newly compromised node creates an additional $n' * p$ usable pairwise keys on average. In this case, the total number of usable pairwise keys grows linearly in the number of attacker-controlled nodes, or $|\mathcal{A}| * n' * p$.

Determining the expected number of keys available to a colluding attacker is more subtle. Ignoring overlap, the total number of usable pairwise keys under collusion grows with the square of the number of attacker-controlled nodes, or $|\mathcal{A}|^2 * n' * p$. This is because each newly-compromised node can be used to communicate with $n' * p$ of the neighbors for *each* attacker-controlled node. For example, when there are two attacker-controlled nodes M_1 and M_2, M_1 can use $n' * p$ of M_2's pairwise keys in addition to $n' * p$ of its own pairwise keys, while M_2 can do the same with $n' * p$ of M_1's keys. More generally, each of the $|\mathcal{A}|$ attacker-controlled nodes can impersonate $|\mathcal{A}| * n' * p$ nodes. Consequently, the total number of keys available under collusion (ignoring overlap) is $|\mathcal{A}| * |\mathcal{A}| * n' * p = |\mathcal{A}|^2 * n' * p$.

Taking node overlap into account does significantly slow the growth of attacker-controlled pairwise keys. The right graph in Figure 3.3 shows a close-up view of the left graph for smaller proportions of attacker-controlled nodes. One can see that the number initially grows quadratically before quickly slowing to linear growth; note the coefficient is much larger than in the independent case.

To demonstrate the limiting value for the total number of keys under collusion, Figure 3.4

plots the number of usable pairwise keys per attacker-controlled node for various values of $|\mathcal{A}|$. As $|\mathcal{A}|$ grows larger, each colluding node possesses an average of $n * p$ usable pairwise keys (note that the graph appears to asymptotically approach $1\,000 * .25 = 250$ pairwise keys per attacker node). Thus the average total number of pairwise keys for $|\mathcal{A}|$ compromised nodes is $|\mathcal{A}| * n * p$ (accounting for overlap). Note that n is typically much larger than n' and that n' stays constant even as n grows. For our simulations, $\frac{n}{n'} = 16.67$. Therefore, even at its limiting growth rate, colluding attackers obtain $\frac{n}{n'}$ times as many usable pairwise keys as when acting alone.

The following table restates the formulas obtained for the total number of usable keys available to an attacker:

Independence	**Collusion (initial)**	**Collusion (limiting)**
$\|\mathcal{A}| * n' * p$	$\|\mathcal{A}|^2 * n' * p$	$\|\mathcal{A}| * n * p$

3.2.2 Quantifying attacker penetration

We have just explained how many keys an attacker might expect to obtain when compromising several nodes in a sensor network. But how do the number of keys brought under adversary control compare to the number of legitimate keys available? We have devised a telling measure:

$$\mathcal{I}(\mathcal{A}) = \frac{|\bigcup_{M \in \mathcal{A}} \mathcal{U}(M)|}{|\bigcup_{M \in \mathcal{A}} \bigcup_{B \in \mathcal{N}(M)} \mathcal{U}(B)|}$$

$\mathcal{I}(\mathcal{A})$ compares the number of usable pairwise keys available to attacker-controlled nodes (numerator) to the number of usable pairwise keys available to the neighbors of the attacker-controlled nodes (denominator). When the attacker's aim is to control as many of its neighbor's communication channels (i.e., shared keys) as possible, then $\mathcal{I}(\mathcal{A})$ is meaningful since it measures the average proportion of keys under the attacker's control. Consequently, $\mathcal{I}(\mathcal{A})$ reveals the degree to which an attacker's geographical neighbors are outnumbered by masquerading attacker nodes.

Figure 3.5 measures $\mathcal{I}(\mathcal{A})$ for increasing $|\mathcal{A}|$. It demonstrates that a small collection of colluding nodes can overtake a large fraction of its neighbors' communications. Without colluding, attacker-controlled nodes gain no added influence over their neighbors: compromising 2% of the network yields control over just 2% of their neighbors' pairwise keys. With colluding, attacker-controlled nodes quickly gain considerable influence over their neighbors: compromising 2% of the network yields control over 27% of their neighbors' pairwise keys; compromising 5% enables power over approximately half of the valid communication channels.

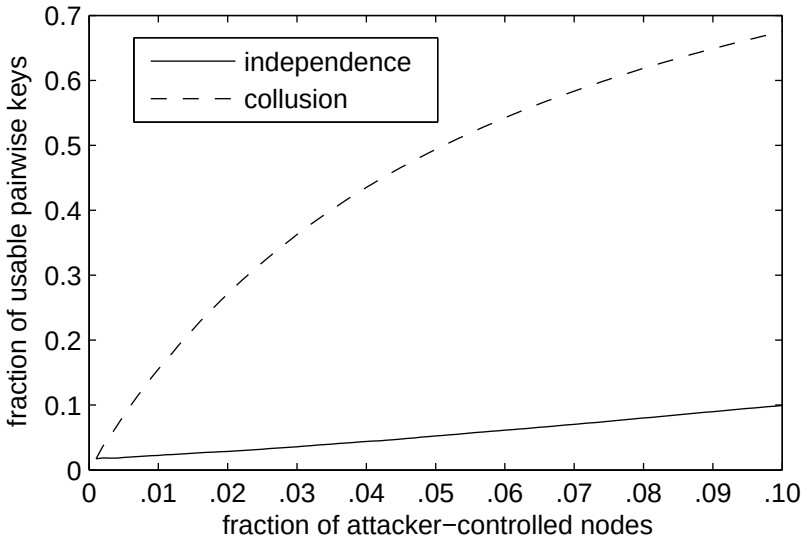


Figure 3.5: $\mathcal{I}(\mathcal{A})$, fraction of communication channels controlled by attacker.

Therefore, a colluding minority, while not capable of eavesdropping, can nonetheless swallow much of the network's interactions. A distributed voting scheme can be undermined by a 5% colluding minority since half of the votes are cast by the attacker. Any application that requires honest interaction with the majority of node neighbors is susceptible. Unfortunately, most sensor network applications do require non-malicious majorities, from routing to data aggregation.

3.3 Discussion

3.3.1 Storage requirements

We have not considered whether colluding nodes can store and transmit all applicable pairwise keys. From Figure 3.3, each attacker-controlled node receives at most $n * p$ usable pairwise keys from collusion in addition to $n * p$ pre-distributed pairwise keys; thus, these nodes need to store up to $2n * p$ keys. Given the severe memory constraints placed on sensor nodes, storing extra keys could prove too onerous.

However, an observation about the effects of overlap on available keys reveals an alternative requiring no hardware modification to meet added storage demands. Each pairwise key can only be used once: when two nodes collude, the only usable keys for one node are those that could not be used by the other. Pairwise keys could help two nodes only when overlap exists in the communication range. As discussed in Section 3.1, only one node may use such keys.

A resourceful attacker can exploit this fact by adding another step to the collusion. After

a node shares a pairwise key with another attacker-controlled node, it can delete the key and replace it with any keys provided by the other node. Thus key-sharing becomes key-swapping. In the end, each attacker-controlled node still stores $n * p$ keys, but now each key can be used to communicate with neighbors.

3.3.2 The cost of pre-distributing secrets

The problem with assigning general-purpose keys to nodes prior to deployment is that such secrets often prove more useful to attackers than to ordinary nodes. A node should store keys for communicating with its neighbors; holding any more, as advocated by pre-distribution schemes, only serves a collusive attacker. Until now, the only costs attributed to pre-distribution have been storage-related. We have shown how pre-distributing secrets also raises security costs.

In fact, any scheme that pre-assigns global secrets to locally-communicating nodes is at risk of similar attacks. Instead of pre-distributing pairwise keys directly as in Chan *et al.*'s approach, Du *et al.* [52] and Liu and Ning [101] randomly pre-distribute capabilities for computing pairwise keys. Nodes may collude just the same, spoofing keys from shared seeds. More recently, PIKE [37] reduces node storage requirements to $O(\sqrt{n})$ pairwise keys, yet remains susceptible to a collusive minority of key-swapping nodes.

3.3.3 Countermeasures

There are two approaches to countering collusion attacks: either reducing the utility of compromised nodes to attackers or detecting the reuse of pairwise keys. The former can be limiting, while the latter is often quite expensive.

One option for reducing utility is for nodes to discard unused keys after an initialization phase, but this inhibits mobility and prevents new nodes from joining the network once initialization is complete. Another is to reduce the number of pre-loaded keys. This can be achieved in many ways, though each technique introduces its own limitations. Some have proposed just pre-loading keys that are geographically close [102, 51]. However, it is not always reasonable to assume the existence of topological knowledge prior to deployment, especially in mobile applications. A more radical approach is key infection, which scraps pre-distribution altogether in favor of simply transmitting keys in the clear for a brief period during network initialization [14]. Key infection schemes are therefore not susceptible to the attack presented here, though they are more vulnerable when a powerful adversary is present (a global passive adversary in threat models **T.0** or **T.1**, rather than a restricted passive adversary during initialization as in **T.2**).

If a sensor network is deployed with a uniform density, then its nodes can detect whether they are under attack by tracking how many connected neighbors they have. Nodes have

n' neighbors on average, but an attacked node may be misled into believing it has an additional $|\mathcal{A}| * p$ neighbors. However, determining which of these neighbors are fake can be difficult, especially if colluding attackers do not reuse the same keys in overlapping node neighborhoods.

One way to identify misbehaving nodes is to require all nodes to transmit their locations. While this too can be faked, key reuse can be detected if nodes recursively ask their neighbors for the location of their transaction partners. Because even a faked location must be within communication range of a duped node, an attacker has no choice but to transmit multiple locations for a single identity. In [133], Parno *et al.* propose a similar detection scheme for node-replication attacks on sensor networks.

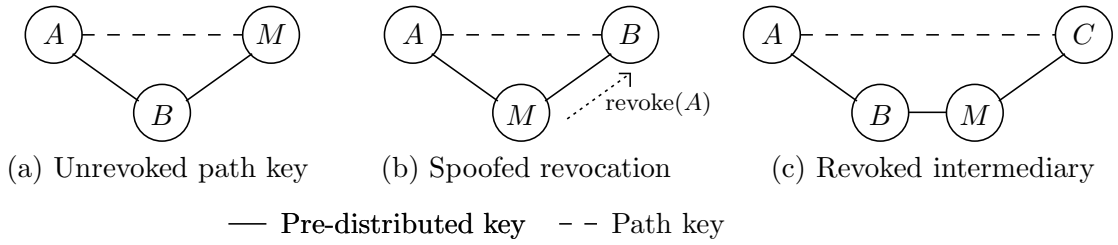
Such detection schemes do have drawbacks, though. Requiring nodes to transmit their location helps an attacker target new nodes for compromise. It is also quite costly to do so: even Parno *et al.*'s 'efficient' technique requires a further $O(\sqrt{n})$ storage per node and $O(n\sqrt{n})$ messages.

Strategies for defending against Sybil attacks given in [124] do not apply since nodes present multiple identities by reusing legitimate identifiers and pairwise keys.

3.4 Path-key-enabled attacks

Most key-distribution schemes pre-load a limited number of secret keys into permanent memory so that nodes can either communicate directly using a shared key or, failing that, set up a *path key* using intermediaries they do share keys with. Path keys are a necessity for any scheme that minimizes storage costs prior to deployment. But path keys must also be considered in the later stages of credential revocation. Existing revocation proposals [54, 38, 36] fail to remove path keys established during operation. This oversight enables attackers to wreak havoc in a number of ways: rejoining the network after dismissal, issuing spoofed revocation messages, and retaining access to path keys established for others. Safeguarding revocation mechanisms from these attacks is essential.

We demonstrate that existing proposals used in conjunction with path keys are vulnerable to a number of attacks, defeating attempts to revoke bad nodes and enabling Sybils. We propose *path-key records*, which detail the identifiers of proxy nodes that help establish each path key. These records are used to identify and remove path keys tainted by a bad node. We show that the combined use of path-key records and blacklisting can secure a centralized revocation mechanism such as Eschenauer and Gligor's. We also show how to modify decentralized revocation schemes to make revocation decisions verifiable to the entire network. Finally, we show that naïve instantiations of Sybil detection mechanisms where results are not verifiable by third parties leave the network vulnerable to path-key-enabled Sybil attacks.



(a) Unrevoked path key (b) Spoofed revocation (c) Revoked intermediary

Figure 3.6: Path-key attacks on revocation mechanisms.

3.4.1 Path-key attacks on revocation mechanisms

Incomplete revocation of path keys

In a centralized revocation scheme, the base station issues revocation orders verifiable by all other nodes which then delete any paths keys shared with the revoked node. However, under Chan *et al.*'s distributed blackballing scheme, the only nodes that can verify votes are those that can participate in a revocation vote. Therefore, only these nodes, which have been pre-assigned keys, know to revoke keys; therefore, only the pre-assigned keys are removed during revocation. Notably, *nothing is done to remove any path keys established with the revoked node.*

Any revocation scheme that does not remove path keys is vulnerable to the attacks in Figure 3.6. Consider the network in Figure 3.6(a). Suppose a malicious node M has been identified and a revocation order issued to all nodes sharing pairwise keys with M. B knows to remove the key shared with M, but A does not, so the path key established between M and A continues to function.

It is not possible to counter this attack by allowing A to accept forwarded revocation claims from B, since A cannot verify the veracity of the claim from B of M's revocation (apart from that B made it). Accepting an unauthorized forwarded message permits the attack shown in Figure 3.6(b) where the undetected malicious node M could lie to B, falsely claiming that honest A had been revoked.

Malicious intermediaries and path keys

The threat of malicious intermediaries during the establishment of path keys has been investigated by a number of authors [14, 38, 176]. These authors have focused on making the path-key establishment mechanism as robust as possible under threat model **T.0** and include techniques such as using multiple disjoint paths. These methods make it harder, but not impossible, for an attacker to compromise a path key, requiring that more intermediary nodes be compromised.

However, these authors do not consider what to do if, as a result of such an attack, a path key is, or might be, compromised. For example, suppose that M has served as an

intermediary to establish a path key between A and C as in Figure 3.6(c). Suppose further that M is subsequently identified as malicious and revoked from the network. While C could observe a revocation order for M, it is unaware that its path key to A was set up via M. M could use its knowledge of the path key to eavesdrop on or rejoin the network. Clearly, the path key should also be revoked.

The use of multiple disjoint paths during path-key establishment simply changes the threshold at which the path key should be revoked. Once an intermediary on each path has been compromised, the resulting path key should also be revoked. A conservative network may require this to happen earlier (e.g., once half of the paths are compromised).

Note that both threat models **T.0** and **T.1** are relevant. In the former case, the attacker has actively compromised M prior to path-key setup and can immediately determine k_{AC}. However, a path-key recovery attack is still possible under threat model **T.1**, where an adversary eavesdrops traffic during path-key setup but does not compromise the intermediary until after path-key setup. Suppose A establishes a path key to C, using M as an intermediary. Here, A sends $\{k_{AC}\}_{k_{AM}}$ to M , which then transmits $\{k_{AC}\}_{k_{MC}}$ to C. When the attacker subsequently compromises M, she can recover k_{AM} or k_{MC} and decrypt the message containing k_{AC}.

Compromised but unrevoked pool keys

The centralized revocation scheme of Eschenauer and Gligor [54] is susceptible to an additional path-key attack. The authors advocate that nodes select unused keys from their key rings as path keys. These keys are, of course, pool keys. A malicious node can establish as many path keys with neighbors as possible, requiring them to provide an unused pool key (and therefore, a key that the attacker does not already possess).

Notably, the network owner is never informed that the node knows additional pool keys. Therefore, should the network owner subsequently revoke the node, none of the path keys are removed. The malicious node retains not only the path-key-enabled links to its neighbors, but also the pool keys for establishing communications back into the system. However, addressing this attack by tracking and removing path keys (and thus pool keys) could enable a denial-of-service attack whereby the adversary deliberately depletes the key pool by setting up many unused pool keys as path keys prior to revocation.

Unauthorized reentry of revoked nodes

One problem with implementing revocation by simply deleting shared keys is that the removal can be reversed when multiple undetected compromised nodes are present. Suppose malicious nodes M_1 and M_2 both share link keys with honest node A, and a revocation order is issued for M_1 but not M_2. A deletes its link key with M_1, as do all honest

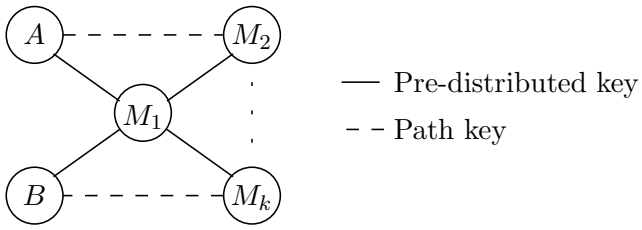


Figure 3.7: Path-key-Sybil attacks on revocation mechanisms.

neighbors of M_1. Yet M_1 can rejoin the network by establishing path keys using M_2 as an intermediary. Unless honest nodes are required to maintain a network-wide blacklist of all revoked nodes, A will not remember that M_1 has already been revoked. Under random path-key establishment, M_1 can rejoin via any colluding node. For deterministic path-key establishment schemes, M_1 can only rejoin via nodes that have been pre-assigned a shared key. This is because any node can compute the identities of keys shared between nodes to deduce who M_1 is eligible to establish a path key with.

3.4.2 Path-key-enabled Sybil attacks

Pool-key-based pre-distribution schemes are susceptible to Sybil attacks since shared keys are not guaranteed to be unique. Pairwise-key-based pre-distribution schemes, by contrast, should be Sybil-resistant since the keys are unique, which enables authentication between nodes sharing keys. However, consider the scenario given in Figure 3.7. Node M_1 shares a pairwise key with A but fabricates the existence of several fake nodes $M_2, \ldots, M_k$ and requests path keys for each of them. Under Chan *et al.*'s distributed revocation protocol, these Sybil nodes are unrevokable! Their sets of voting members, $\mathcal{V}_{M_2}, \ldots, \mathcal{V}_{M_k}$, are all empty since the identities are fake, yet A has no way of knowing this. Node M_1 can use each of these fake identities to carry out attacks, while otherwise behaving honestly to its real neighbors.

Newsome *et al.*'s widely-cited paper [124] claims that 'an adversary cannot fabricate new identities' under the random pairwise scheme, making it immune to Sybil attacks. We have just demonstrated this claim to be incorrect by showing how path keys can be used to create Sybil identities under the random pairwise scheme. Consequently, a mechanism such as Newsome *et al.*'s must be employed to detect Sybils for random pairwise schemes, but their direct key validation Sybil defense cannot do so. Instead, we require the hypothetical indirect validation protocol discussed by the authors. An indirect validation protocol would enable nodes not sharing keys with the target node to verify claims from other nodes that do share keys. While such a protocol would be extremely useful, it requires transitive trust between nodes which cannot normally be guaranteed. In fact, transitive trust has enabled Sybil attacks in the first place: in Figure 3.7, node A and B mistakenly trust that M_1 is telling the truth about nodes $M_2, \ldots, M_k$.

Consequently, the papers outlining revocation proposals and Sybil detection mechanisms for use in sensor networks suffer from irreconcilable cross-dependencies. While Chan *et al.*'s blackballing revocation scheme described in Section 2.3.2 assumes the existence of adequate Sybil attack detection and refers to Newsome *et al.*'s techniques as an example, path-key-enabled Sybil attacks remain an unaddressed impediment to effective revocation, and to pairwise key pre-distribution in general.

Note also that the degree-counting mechanism proposed by Chan, Perrig and Song [38] cannot detect Sybil attacks. It detects attackers using more pairwise keys than allowed, but the attacks just described create path keys without using any pre-assigned pairwise keys.

3.5 Secure path-key revocation

We now present three techniques for securing revocation mechanisms from the path-key attacks outlined above: (1) complete notification of node revocation, (2) path-key records to identify malicious intermediaries and (3) blacklists to prevent unauthorized reentry via path keys. We propose both centralized and decentralized solutions where appropriate.

3.5.1 Complete notification of node revocation

Every node capable of establishing a path key with another node must be notified of that node's revocation order, should one be issued. In sensor networks where the topology is unknown before deployment, path keys could conceivably be established between any two nodes. Thus, every node must be notified of every revocation.

A centralized revocation mechanism for removing pre-distributed keys, similar to the one proposed by Eschenauer and Gligor, can be trivially augmented to revoke path keys. Here, the central authority unicasts a message to every node (signed by the pairwise key shared between the authority and the node) instructing them to remove any path keys established with the revoked node.

For a decentralized revocation mechanism, nodes must be able to verify revocation messages sent by other nodes even when the verifying node does not necessarily trust the sender. To do so, each node is loaded with an authentication value for the revocation secrets of all n nodes in the network. (Recall from Section 2.3.2 that revocation secrets are reconstructed from voting shares broadcast by nodes eligible to decide when it is best to revoke a node.) This is in contrast to Chan *et al.*'s distributed revocation mechanism, which equips nodes with only the ability to verify the revocation of nodes sharing pre-assigned keys. This $O(n)$ storage cost (nT for Chan *et al.*'s blackballing scheme) could impede deploying revocation with symmetric keys for large networks.

We can reduce the storage costs to $O(v \log n)$ using a couple of tricks. Since a node is only revoked once, the revocation secret can be kept constant over all T sessions even if the revocation shares change. Using Shamir's polynomial-based sharing scheme [147], we can select a new polynomial function for each session while leaving the free term unchanged. So long as the threshold is not reached, shares revealed during previous sessions offer no assistance in determining the current secret. Using a constant revocation secret over all sessions reduces storage obligations for authenticating revocation secrets to n from nT.

Costs may be reduced even further by storing authentication values as leaves in a single hash tree [108] (see Section 2.3.2 for an explanation of hash trees). The tree's leaves are assigned $h(\mathsf{rev}_B)$ as preimages for every $B \in N$. Every node stores a copy of the tree's root value. However, we must be careful in distributing the $\log n$ path-authentication values. We cannot rely on the node being revoked to provide the information to verify its own removal; a safer alternative is for every voting member to keep a copy of the path-authentication values for each node it might revoke. Since $|\mathcal{V}_B| = v \ \forall B \in N$, the final storage requirement for complete revocation notification is $1 + v \log n$ per device.

To summarize, under our proposed *secure blackballing* scheme, each node A is loaded with information to do the following:

1. **Vote against each node $B \in \mathcal{V}_A$:** Secret share $\mathsf{rev}_{B,i,A}$ $\forall B \in \mathcal{V}_A, i \in \{1, \ldots, T\}$ (storage cost vT)
2. **Prove to all $B \in \mathcal{V}_A$ that vote is valid:** $\log v$ path-authentication values for each vote $\mathsf{rev}_{B,i,A}$ (storage cost $vT \log v$)
3. **Verify votes from others:** Hash tree roots $\forall B \in \mathcal{V}_A, i \in \{1, \ldots, T\}$ (storage cost vT)
4. **Verify revocation secrets for all $B \in N$:** From hash tree with leaves $h^2(\mathsf{rev}_B)$ $\forall B \in N$, store tree root and path-authentication values matching leaves $h^2(\mathsf{rev}_B)$ $\forall B \in \mathcal{V}_A$ (storage cost $1 + v \log n$)

3.5.2 Path-key records to identify malicious intermediaries

Recall that under threat model **T.1**, an attacker may collect traffic that passes through a node, then compromise it and determine all path keys established with the node as intermediary (Figure 3.6(c)). However, if nodes periodically update their link keys using a one-way function, e.g., $k'_{AB} = h(k_{AB})$, then an attacker cannot recover any path keys established prior to node compromise. The attacker can only determine the most recent link keys.

To address the case where nodes are compromised during path-key establishment (under threat model **T.0**), nodes must keep track of the intermediate nodes used to establish each

path key so that affected path keys can be removed once node compromise is detected. To do so, nodes can build a list of all node identifiers used as intermediaries in conjunction with path-key establishment. When node A establishes a path key with node B, it stores a *path-key record*

$$B, k_{AB}, N_1, N_2, \ldots, N_l$$

where k_{AB} is the path key, and $N_1, N_2, \ldots, N_l$ are the identifiers for the l intermediate nodes. Whenever a revocation order is issued, nodes must check their path-key records for the revoked nodes, discard affected path keys and reinitiate transmission to discover a new path key.

Path-key record generation and verification should remain decentralized, even when access to a central authority is used for other components of path-key revocation. Because path-key records are constructed every time a path key is established, it is unreasonable to always consult a base station. If this frequency of communication with base stations were allowed, then nodes would be better off using the base station to set up the path keys in the first place.

For the path-key records to hold value, nodes must be capable of ensuring that the neighbors in the record are in fact the intermediaries used. It is for this reason that random path-key establishment techniques, where nodes simply ask their neighbors to serve as intermediaries, are inappropriate. Nodes along the path cannot be allowed to help build the record since an undetected malicious node M_1 can trivially modify the record's contents to its own end. For instance, M_1 could replace its identifier with that of an honest neighbor C so that if M_1 is subsequently removed its path key will not be.

Instead, we advocate using a deterministic key-discovery technique (e.g., [176, 48, 37]) to remove the potential for manipulation during path-key record construction. Under deterministic path-key establishment, nodes can compute the list of key identifiers for any node. Whenever a node must set up a path key, it unilaterally decides which nodes to use as intermediaries and builds the path-key record without consulting other nodes. Deterministic key-discovery techniques provide a degree of authentication to the identifiers selected for the path; we exploit this when constructing a secure path-key record.

Suppose A wants to establish a path key with D. By computing several lists of key identifiers, A determines that it shares a link key with B, which shares a key with C, which in turn shares a key with D. A stores this information in the path-key record and attempts to set up the path key. In the event that B and C or C and D are not within communication range of each other, the key setup will fail and A will have to select a new path. Now suppose that one of the nodes A selects happens to be malicious (say B). B cannot add or remove identities, including its own, to the path-key record.

Minimizing the number of intermediaries used to establish path keys reduces the likelihood of selecting a malicious node as an intermediary. Furthermore, using multiple

disjoint paths for establishing path keys [14, 38, 176] decreases the likelihood of path-key compromise. Here, path-key records can be modified to include sets of node identifiers for each of the k paths:

$$\begin{aligned} B, K_{AB}, \{N_{11}, N_{12}, \ldots, N_{1l_1}\}, \\ \{N_{21}, N_{22}, \ldots, N_{2l_2}\}, \\ \vdots \\ \{N_{k1}, N_{k2}, \ldots, N_{kl_k}\}. \end{aligned}$$

3.5.3 Blacklists to prevent reentry via path keys

As described in Section 3.4.1, a previously revoked node can reenter the network by setting up path keys via an undetected malicious intermediary. To counter this, nodes must maintain an up-to-date blacklist of all revoked nodes.

A centralized blacklist maintained by the base station is undesirable, since nodes set up path keys without first consulting the base station. However, a consistent distributed blacklist is easy to construct. Since nodes already observe every revocation order, they simply store the identifiers of each removed node.

So long as revocations are infrequent, keeping such a blacklist does not present a storage problem. An alternative to maintaining a blacklist that still identifies unauthorized reentry via path keys is to combine reentry detection with node-replication detection. In [133], nodes periodically transmit signed copies of their identifiers and locations across the network. Nodes check these messages for multiple claims originating from different places. Similarly, nodes could be required to remember only a subset of revoked nodes (e.g., each node remembers nodes it has decided to revoke). Nodes check the identifiers transmitted in the node-replication-detection messages against their own subset of the blacklist. If detected, nodes forward the message to the base station, which issues a new revocation order. This scheme can be less expensive than requiring nodes to maintain a complete blacklist when node-replication detection is already in frequent use.

3.5.4 Cost summary

In summary, path keys impose the following additional costs to those outlined in [36] for revocation to be effective:

1. authenticate revocation secrets for all nodes in the network;
2. maintain path-key records listing all intermediaries on every path key;
3. maintain blacklists of all revoked nodes in network.

These previously unaccounted costs reflect the difficulty in designing efficient revocation mechanisms using pre-distributed symmetric-key cryptography. We believe these costs are unavoidable whenever anything fewer than complete pairwise keys are pre-loaded.

3.6 Conclusion

Any symmetric-key-management scheme pre-distributing less than complete pairwise keys necessarily weakens notions of identity. Complications inevitably ensue. In this chapter we presented a novel collusion attack on the class of pairwise key pre-distribution schemes and demonstrated how the attack can undermine secure communications in a sensor network. We also identified problems with revocation mechanisms and Sybil identities caused by path keys. We proposed effective countermeasures to ensure that keys shared with or exposed to revoked nodes are removed, and recommended blacklists to prevent the unauthorized reentry of revoked nodes. We noted that exposure to path-key attacks may be limited by employing deterministic path-key establishment mechanisms and minimizing the number of intermediaries used. We also showed that, contrary to prior understanding, path keys make incomplete pairwise key-distribution schemes vulnerable to Sybil attacks.

More generally, efficiency gains made at one stage in the network's life cycle may cause unforeseen problems that are expensive to remedy at other stages. We showed that trade-offs made to improve the efficiency of bootstrapping keys to devices opened the door to attacks that were costly to handle during the maintenance phase of revocation, counteracting gains obtained from earlier trade-offs. This resonates with Anderson's argument that protocol designers have long underestimated the maintenance costs of security mechanisms [11]. One could accept these costs as unavoidable and mitigate any resulting attacks with countless patchwork mechanisms. However, we question whether efficient key establishment coupled with inefficient or insecure revocation is desirable. Instead, we should perhaps explore alternatives, from selective use of asymmetric cryptography to more innovative revocation mechanisms, as proposed in the next chapter.

Chapter 4

Decision mechanisms for removing bad devices

Responding to misbehavior in a decentralized network environment is difficult. Since no single device has authority over anyone else, the most common solution is to let the devices vote on whether to eject errant devices. We have already described one such scheme for WSNs, called blackballing (see Section 2.3.2). In Chapter 3 we fixed some problems with blackballing to arrive at secure blackballing, which prevents network reentry via path keys. Even so, many more fundamental problems remain.

Voting schemes like blackballing are slow, expensive and prone to manipulation. They are susceptible to false accusations – any set of collusive attackers that exceeds the voting threshold can eject good devices at will. Delays between the first detection of misbehavior and a device's revocation can occur while waiting for a threshold of good users to reach consensus. Voting mechanisms also do not cope well with node mobility and churn – more interactions mean more opportunities for false accusations. Finally, blackballing is expensive in terms of computation, communication and storage.

Therefore, in this chapter we propose new decision mechanisms with the aim of improving security and performance. We use the secure blackballing decision mechanism as a basis for comparison.

Recall from Section 2.3 that the defender's goal is to exclude malicious nodes from the network. Decision mechanisms enable nodes to eject misbehaving devices in a decentralized way, based on input from detection mechanisms. We anticipate two primary goals for the attacker. First, an attacker aims to continue participating on the network by avoiding a removal decision. Second, an attacker might also wish to disrupt the network by abusing the decision mechanism to eject non-malicious devices.

4.1 Reelection

Existing proposals for decision and punishment require action by the honest members of the network to remove misbehaving nodes. All nodes must follow the procedures of voting, blacklisting and key removal in order to stop a malicious node from rejoining the network. This represents a significant computational and communications burden shared by all good nodes that can be shirked by bad ones. In contrast, we propose a mechanism that turns the computational liability on its head by requiring additional effort for good nodes to continue participating on the network but no effort to remove bad devices.

We propose a system where a node, upon joining the network and periodically thereafter, must demonstrate that it is still authorized to be on the network. Revocation becomes a matter of preventing a bad node from renewing its membership. Conceptually, this corresponds to a voting scheme with positive votes instead of negative ones: good nodes reelect each other to the club once in each time period.

We first present a robust protocol for remaining on the network using threshold-secret-sharing mechanisms. We then propose a lightweight reelection mechanism using hash operations exclusively. In Section 4.3.2 we discuss some trade-offs between blackballing and both incarnations of the reelection decision mechanism.

4.1.1 Reelection for semi-capable devices

We define a *network access token* $\mathsf{access}_{A,i}$ that allows node A onto the network during time period $i \in \{1, \ldots, T\}$. A must present the token $\mathsf{access}_{A,i}$ to its neighbors to continue interacting with them. Tokens are created using a hash chain [92] where $\mathsf{access}_{A,i-1} = h(\mathsf{access}_{A,i})$, for $i = 1, \ldots, T$. The end-of-chain authentication value $\mathsf{access}_{A,0}$ is distributed to every voting member $B \in \mathcal{V}_A$, which can authenticate $\mathsf{access}_{A,i}$ for time period i by verifying that $\mathsf{access}_{A,0} = h^{(i)}(\mathsf{access}_{A,i})$.

Each token $\mathsf{access}_{A,i}$ is divided into v shares using a (d, v) threshold-secret-sharing scheme. The shares are distributed to the voting members $B \in \mathcal{V}_A$. In particular, B is assigned shares $\mathsf{access}_{A,i,B}$ for each $i = 1, \ldots, T$. The responsibility for reconstructing tokens rests with A, which asks its voting members for their shares. Once d of A's neighbors reveal their share, A can reconstruct $\mathsf{access}_{A,i}$ and send it to A's neighbors. So B casting $\mathsf{access}_{A,i,B}$ is an affirmation of A's honesty rather than a claim of impropriety. Hence, the threshold of votes d may be larger than for blackballing, if more positive votes are required than negative ones. Note that if the voting members are those pre-assigned a pairwise key (as in Chan *et al.*'s blackballing scheme), then nodes should delete any voting shares for non-neighbors following neighbor discovery. Alternatively, we could reduce the average number of voting members v by choosing the voting set upon deployment (under threat models **T.1** or **T.2**).

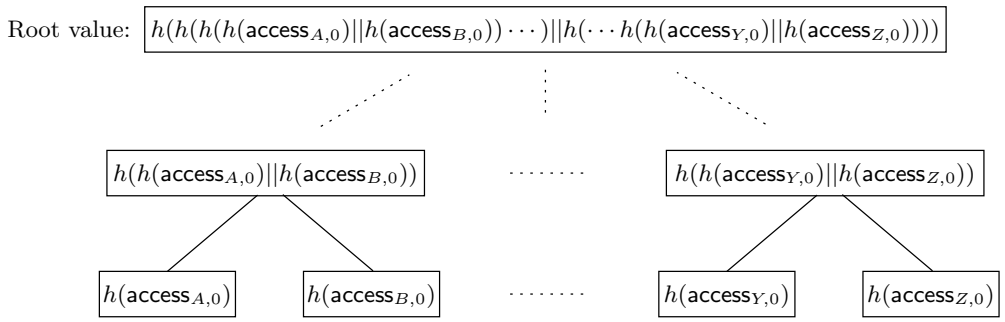


Figure 4.1: Merkle hash tree used for authenticating end-of-chain values $\mathsf{access}_{A,0}$ for every $A \in N$.

Nodes must store additional information to verify transmitted votes and tokens. To verify received votes, node A can store a hash of share $h(\mathsf{access}_{A,i,B})$ for every $B \in \mathcal{V}_A$ and $i = 1, \ldots, T$. To authenticate reconstructed tokens, the owner creates a hash tree where the leaf preimages are the end-of-chain authentication values $\mathsf{access}_{A,0}$ for every $A \in N$ (see Figure 4.1). Each node A stores the tree's root-authentication value, its own end-of-chain authentication value $\mathsf{access}_{A,0}$ and the $\log n$ path-authentication values required to authenticate $\mathsf{access}_{A,0}$.

For example, during time period 3, once node A receives enough shares from neighbors to reconstruct $\mathsf{access}_{A,3}$, A broadcasts $\mathsf{access}_{A,3}$, the end-of-chain authentication value $\mathsf{access}_{A,0}$, and $\mathsf{access}_{A,0}$'s respective path-authentication values. Other nodes authenticate $\mathsf{access}_{A,0}$ using the received path-authentication values and their own stored copy of the hash tree root. They then compute $h^{(3)}(\mathsf{access}_{A,3})$ and verify that it equals $\mathsf{access}_{A,0}$.

To generalize, here is the reelection protocol for a node during time period i:

1. $A \longrightarrow * :$ A, i
2. $B \longrightarrow A :$ $A, B, \mathsf{access}_{A,i,B}$
3. $A \longrightarrow * :$ $A, i, \mathsf{access}_{A,i}, \mathsf{access}_{A,0}$, path-authentication values
4. $* :$ verify $h^{(i)}(\mathsf{access}_{A,i}) = \mathsf{access}_{A,0}$, verify $\mathsf{access}_{A,0}$

A asks each neighbor B for its share $\mathsf{access}_{A,i,B}$ (step 1).[1] If d voting neighbors cooperate (step 2), then A can reconstruct $\mathsf{access}_{A,i}$, which is then broadcast to A's neighbors (step 3). The neighbors verify $h^{(i)}(\mathsf{access}_{A,i}) = \mathsf{access}_{A,0}$ (step 4).

If node B wishes to vote against A, then it simply deletes all the stored shares $\mathsf{access}_{A,i,B}$, $i = 1, \ldots, T$. Once fewer than d of A's neighbors retain their shares, A can no longer reconstruct tokens. Revocation is final and absolute: an adversary cannot reconstruct the tokens even by subsequently compromising all neighboring nodes. Alternatively, the basic reelection protocol can be changed to temporarily punish A by deleting only a subset of the tokens corresponding to a few time periods.

[1] '$A \longrightarrow *$' means that A broadcasts a message.

To save storage, nodes can delete the revealed shares of good nodes once the following round commences; otherwise an attacker could ask for a neighbor's share so that the intended node does not observe the response. Also, note that step 1 is optional; any node loaded with secret shares for a node A can reveal the share without being asked. Dropping this broadcast step forces nodes to continuously listen for neighbors revealing their shares.

To summarize, each node A is loaded with information to do the following:

1. **Vote for each node $B \in \mathcal{V}_A$:** Secret share $\mathsf{access}_{B,i,A}$ $\forall B \in \mathcal{V}_A$, $i \in \{1, \ldots, T\}$ (storage cost vT)
2. **Verify received shares:** Hash values of all token shares for A (storage cost vT)
3. **Prove to all that token is valid:** Root-authentication value from hash tree and path-authentication values matching $\mathsf{access}_{A,0}$ (storage cost $1 + \log n$)

4.1.2 Lightweight reelection with buddy lists

Reconstructing secret shares can be too demanding for devices with severely limited processors. In addition, the effort involved in pre-assigning, swapping and storing vT shares per node may be unattractive for many applications. More fundamentally, designing voting protocols using secret shares is rigid: the threshold of votes needed to take action remains constant for all nodes and for all time. Because network applications and populations can be dynamic, diverse strategies for taking action should be possible. Risk-averse nodes may prefer to shun a neighbor as soon as one of its other neighbors has done so, while more relaxed nodes could continue interacting with any node still supported by two of its neighbors. A diverse population of risk-averse and risk-seeking nodes might allow the network to perform well in normal circumstances and still operate acceptably under serious attack. Hence, it makes sense to disentangle the voting mechanism as far as possible from the strategy.

We therefore consider a lightweight reelection mechanism that is general enough to support diverse strategies. Here, nodes periodically transmit a *buddy list* of approved neighbors across their local neighborhoods. Since many node neighbors overlap, they can cross-reference received lists to determine whether enough nodes have also approved their buddies. If so, they continue to interact with the nodes during the next time period. The definition of 'enough' is made independently of the protocol mechanism described here.

Approved buddy lists are authenticated using hash chains. However, since the buddies can be chosen without constraints, there is no need to distribute secret shares or store hashes and hash trees to authenticate them. Rather, each user can construct their own hash chain, sign the buddy list with the current period's secret, and reveal the prior round's

secret for verification. This approach is similar to the Guy Fawkes protocol proposed by Anderson *et al.* [13].

Upon deployment, node A computes a hash chain with T values so that $\mathsf{access}_{A,0} = h^{(T)}(\text{seed}, A)$ and $\mathsf{access}_{A,i-1} = h(\mathsf{access}_{A,i})$, for $i = 1, \ldots, T$. Buddy lists are signed with a session authentication key $\mathsf{access}_{A,i}$ during time period i, and $\mathsf{access}_{A,i}$ is not revealed until the start of period $i + 1$. Here is the protocol:

1. $A \longrightarrow *:$ $\mathsf{access}_{A,i-1}, A, i, \mathsf{buddies}, \text{HMAC}_{\mathsf{access}_{A,i}}(A, i, \mathsf{buddies})$
2. $*:$ verify $\text{HMAC}_{\mathsf{access}_{A,i-1}}(A, i-1, \text{buddies})$, delete offending neighbor's keys

Each node A broadcasts a list of approved node identifiers called buddies for time period i, while revealing period $i-1$'s signing key $\mathsf{access}_{A,i-1}$. Neighboring nodes note the buddies claimed for time period i, while also verifying the accuracy of the prior period's buddy list. Any neighbor that does not appear on enough buddy lists is ignored in future rounds.

Notably, no pre-assigned storage or topological information is required, yet buddy lists work even under the conservative threat model **T.0**. They also support extremely general strategies for maintaining a network's trusted membership. Nodes' risk aversion could change over time, according to news from other nodes, or as part of an evolutionary game; one could even implement dynamic games similar to Conway's game of 'Life' [63]. Separating trust strategies from the underlying protocol, and implementing it using lightweight and purely local mechanisms, is the strength of this option.

4.2 Suicide for the common good

Revocation by blackballing has turned out to be complex and costly. Matters were improved by reelection, whereby each node has to persuade a quorum of its neighbors to support its continued membership at regular intervals, and still further by the buddy-list mechanism. We now introduce a radical, simpler and in some ways even cheaper method: suicide.

Decisions are much simpler when taken by a single node. Should a node believe another has misbehaved, it can unilaterally remove the offender. Of course, a malicious node could falsely accuse legitimate ones. Therefore, the act of punishment must be made costly for the deciding node. Suicide removes both the accused *and* accuser from the network. Sacrificing future participation is so costly that it unequivocally demonstrates the veracity of the node's claim. We present three cases in order of increasing complexity: using a central trusted authority; using limited asymmetric cryptography without access to a trusted authority; and using only conventional cryptography without access to a trusted authority.

4.2.1 Suicide using a central authority

The simplest way to implement suicide uses a central authority such as a base station. Upon detecting a node M engaging in illegal activity, node A sends a *suicide note* $\mathsf{suicide}_{A,M}$ identifying both A and M to the base station. The note is authenticated by the pairwise unique key shared between the node and base station. The base station S confirms that node A is entitled to revoke node M and informs other nodes in the network by sending individually authenticated messages. Note that the decision mechanism remains distributed: it is the nodes who decide to revoke because nodes are better positioned to detect misbehavior than far-away base stations.

4.2.2 Distributed suicide using signatures

Nodes may not have access to a trusted base station; instead node A broadcasts a signed note $\mathsf{suicide}_{A,M}$ with the identities of both A and M. The other nodes in the network verify the signature and, if correct, revoke both A and M by deleting all keys shared with them and/or adding both identities to a blacklist.

Public-key cryptography works when nodes are sufficiently capable. The owner generates a new public-private key pair for each node and signs the public key. The key pair, certificate and owner's public key are stored on the node. When a node issues a suicide note, it broadcasts its public-key certificate along with the suicide note for other nodes to verify the public key and suicide note.

In constrained devices, one-time signatures using only pseudo-random functions may be substituted [107]. Each node is pre-loaded with a single private signing key and the associated public key: this key might be certified by the network owner, or a hash of the key might be the device's name, depending on the deployment model and computational constraints. Nodes verifying a signed suicide note must be able to authenticate the public key. Thus the owner constructs a hash tree with the public keys as leaves suitably ordered to tie a node's identity to its position in the tree. Each node stores the root-authentication value and the $\log n$ path-authentication values required to verify its own public key. The path-authentication values are subsequently broadcast along with the signed suicide note.

4.2.3 Flypaper and trolling attacks

One challenge for a decentralized suicide scheme is ensuring that multiple nodes do not issue suicide notes for a single misbehaving node. In a *flypaper attack*, a malicious node in a fixed location presents widely observable misbehavior to attract many simultaneous suicides. A base station S can trivially resolve multiple suicide offers for the same node

by accepting just one of them:

1. A : detects M misbehaving
2. $A \longrightarrow S$: $A, M, \text{HMAC}_{K_{AS}}(\mathsf{suicide}_{A,M})$
3. S : verify signature, wait for duplicates
4. $S \longrightarrow B$: $A, M, \text{HMAC}_{K_{SB}}(\mathsf{suicide}_{A,M})\ \forall B \in N$
5. $*$: verifies signature, deletes keys shared with A, M, adds to blacklist

In a decentralized scheme, where each node must be able to reach a decision independently, two precautions can mitigate a flypaper attack. First, a node can wait a random back-off period ($0 \leq t_{\text{r}} < t_{\max}$) before transmitting an offer. If it observes another suicide note for the same node while waiting for its own timer to expire, the node abandons its offer in favor of the already-published one. If its timer does expire, the node transmits a suicide message. Larger values of $t_{\max}$ lower the probability of a collision at the expense of slower revocation. This back-off can significantly reduce the number of simultaneous transmissions; however, duplicate offers are still possible if a second timer expires before the first transmitted suicide message is received by the second node.

To address this possibility, a tie-breaking mechanism is required. If loose time synchronization exists in the network, nodes can append a timestamp to their signed suicide message. Nodes then wait long enough for all offers to be broadcast (t_{bcast}) and honor the suicide note with the earliest timestamp. Alternatively, time synchronization can be avoided by selecting the smallest random number transmitted along with each suicide message. However, using time stamps to resolve conflicts is more efficient since earlier offers are likely to propagate faster. One consequence when using one-time signatures is that we must now store Q key pairs per node, or generate signing keys on the fly from a root secret. Here is the distributed protocol for two nodes A and B detecting M misbehaving:

1a. A : detect M misbehaving; start random timer t_r
1b. B : detect M misbehaving; start random timer $t_{r'}$
2. A : Timer t_r expires (assuming $t_r < t_{r'}$)
3. $A \longrightarrow *$: $A, M, t_A, \{\mathsf{suicide}_{A,M}, t_A\}_{K_A^{-1}}$
4. $*$: waits t_{bcast} for earlier offers, verifies signature, deletes keys shared with A, M and adds them to blacklist

Trolling is where a node presents itself in several locations, either reusing identities (node replication) or presenting different ones (Sybil). For example, colluding malicious nodes could present the same misbehaving identity in multiple locations. Alternatively, using a powerful transmitter or flying over an area achieves the same effect. We have assumed in this chapter that other mechanisms exist for detecting and preventing Sybil and node-replication attacks. However, our multiple-offer resolution mechanism prevents trolling with reused identities even when node-replication detection is not available, provided the

network is connected and an adequate timeout is used to allow multiple notes to traverse the network.

If the adversary is capable of partitioning the network, then a single malicious node can kill multiple good nodes either by issuing different suicide notes in each partition, or by misbehaving in different partitions with the aim of prompting multiple suicide notes. The number of good nodes affected is proportional to the number of partitions. A potential countermeasure is *resurrection*: once the network is reconnected, several suicides on a single node can be converted into the resurrection of all but the first sacrificed node along with revoking the replicated node. In this case, suicide notes must be stored, and a blacklist operated in preference to deleting keys. (Note that if revocation is reversible, then all the mechanisms compared in this chapter become more complex.)

4.2.4 Extensions: probabilistic suicide and suicide pacts

Suicide requires detection mechanisms that identify malicious nodes with few false positives. Yet the basic mechanism can be extended to cope with uncertainty. Suppose a node detects behavior that is probably malicious, and can assign a probability to this (e.g., bad with $p = 0.7$). Then the node can maintain a running total for each node it observes. When the total exceeds a specified threshold (e.g., $\sum p_i \geq 1$), a suicide is triggered. Consequently, the input from several nodes is taken into account before one issues a suicide. A stateless alternative is for the node to commit suicide with probability p. One limitation of these approaches is that each node operates in isolation, gaining no benefit from the collective knowledge of its neighbors.

We could also modify the suicide offer to include probability p as $\langle \mathsf{offer}_{A,M}, t_A, p, \{\mathsf{offer}_{A,M}, t_A, p\}_{K_A^{-1}} \rangle$. Thus revocation decisions can be made using collective knowledge of uncertain observations, with the participating nodes entering into a *suicide pact* against the suspect. Deciding which member of the pact must carry out the suicide should reflect the probability claimed in the suicide offer, whether based on a weighted, verifiable coin toss, or simple probabilistic suicide in the second round of the protocol.

4.3 Analysis and comparison

4.3.1 Storage and communication costs

A comparison of the storage costs is presented in Table 4.1.

Reelection is more efficient than blackballing in terms of storage: access shares ('positive votes') need only be verified by one node, and only one node (the target node) need store the authentication information for the recovered token. Thus, storage costs for reelection

	Storage per node
Blackballing	$3vT + v \log Tv$
Secure blackballing	$2vT + v \log Tv + 1 + v \log n$
Reelection	$2vT + 1 + \log n$
Suicide (symmetric)	$O(Q) + Q \log nQ$
Suicide (asymmetric)	$O(1)$

Table 4.1: Node storage costs for alternative schemes.

	# sessions	Setup per session
Secure blackballing	T	0
Reelection	T	d unicasts +2 broadcasts
Suicide (symmetric)	1	0
Suicide (asymmetric)	1	0

	Communications per revocation	Min accrued communication without revocation	Max communications without revocation
Sec. blackballing	$d + 1$ broadcasts	0	dT
Reelection	0	$2dT$ unicasts $+T$ broadcasts	$2dT$ unicasts $+T$ broadcasts
Suicide (sym.)	1 broadcast	0	0
Suicide (asym.)	1 broadcast	0	0

Table 4.2: Communication costs for alternative schemes.

are $O(vT + \log n)$, compared to $O(vT + v \log n)$ for blackballing. However, reelection arguably may require more, shorter time periods (and hence larger values for T) since a revoked node does not immediately lose access to the network but only at the end of the current time period. As with blackballing, we can also reduce the number of eligible voters v using a weakened threat model **T.2**.

Both blackballing and reelection increase storage well beyond the initial costs of key distribution. This is not easily borne, particularly for large networks of constrained devices. In contrast, suicide using one-time signatures is not affected by the number of keys that are pre-assigned. Here, a node only needs the ability to transmit a very small number (Q) of offers. But the size of each public and private key for one-time signatures can be very large, requiring two hash values per signed bit. Suicide using asymmetric cryptography requires far less storage as nodes keep only their own private and public keys (small when elliptic curve cryptography is used) as well as the owner's public key and certificate.

Table 4.2 shows the respective communication costs associated with each scheme. Reelection is unique in that there is a fixed cost per session when a node asks for and receives its access shares. This requires a node to broadcast a request to its immediate neighbors,

while the d shares can be returned as unicast messages. The reconstructed token is also broadcast. However, communication costs do not increase as nodes get revoked. In contrast, the communication costs of blackballing increase with the number of votes cast. $d + 1$ locally broadcast votes are required to remove a node, where each vote comprises the vote and $\log v$ path-authentication values. These messages may need to be forwarded by other nodes since we are not guaranteed that all voting members are within communication range. The final revocation order must be broadcast across the network (along with $\log n$ path-authentication values) to ensure that everyone revokes the malicious node. Since votes are only valid in a given session, it is possible for up to d votes to be cast in each session without revoking a node. Thus, while blackballing is more efficient than reelection under low rates of misbehavior, reelection fares better when more attackers are present.

Asymmetric suicide is noted for its low communication costs (only one network-wide broadcast), though these energy savings are offset by increased computational expense due to the use of signatures when compared to symmetric-key-based schemes. Of course, voting schemes using asymmetric cryptography face even higher costs since signatures are required for every vote.

Suicide using one-time signatures can face high communication costs, particularly if the public key must be transmitted along with the signature instead of being pre-loaded onto devices. In fact, recent work has demonstrated that one-time-signature schemes perform only slightly better than elliptic curve algorithms when considering both the communication and computational overhead [146]. Thus, the limited asymmetric cryptographic operations needed by suicide may be preferable to the higher complexity and storage requirements imposed by one-time signature schemes.

4.3.2 Strategic trade-offs between decision mechanisms

We now discuss several underlying design considerations for each of the strategies we have presented in this chapter, along with the resulting trade-offs.

Among voting-based schemes, there is a dichotomy between negative votes (blackballing) and positive votes (reelection). Which approach is better depends on the rate of attacker activity. As mentioned in Section 4.3.1, under low rates of misbehavior, blackballing is more efficient because fewer votes need to be transmitted. Reelection requires devices to transmit votes only when devices are good. Secret-sharing based reelection has slightly lower storage requirements compared to blackballing, since voting shares do not need to be proven to the other voting members.

A second design consideration is whether to pre-assign a limited number of votes for nodes to use (as is done by secret-share-based blackballing and reelection) or to leave voting unrestrained (as is the case for buddy lists). Part of the rationale of pre-assigning

secret shares stems from a fear of voting abuse – if a single node can vote infinitely often, it could cast many votes in an attempt to remove bad devices. Consequently, voting shares can be pre-distributed in a sensor network just as keys already are.

However, there are significant downsides introduced by pre-assigning votes. Foremost, pre-assigning votes restricts mobility since voting shares may be blinded to prevent abuse. More generally, pre-assigning votes without knowledge of network topology allows votes to fall into the control of attackers when they compromise a node (we have already shown some negative consequences of unnecessarily pre-assigning keys in Chapter 3).

Buddy lists, by contrast, do not inhibit mobility because devices are free to choose which identifiers get placed in the list. One could design a blackballing complement to buddy lists. However, abusing negative votes in such a scheme might be easier. Buddy lists implicitly construct a reputation system – devices list the other nodes that they trust. This in-built reputation protects against abuse. An unrestricted blackballing scheme would need to build similar measures of trust to counteract malicious votes. Note that suicide, like buddy lists, does not suffer from the perils of pre-assigned credentials.

A third design consideration for the strategies presented is whether to take decisions collectively (as in blackballing and reelection) or unilaterally (as in suicide). A unilateral strategy frees suicide from the problems of stale votes and delays before revoked nodes are removed from the network. Any collective decision mechanism requires several devices to observe misbehavior. In addition to slowing the decision process, multiple devices must detect attacks in order to revoke bad nodes. As discussed in Section 4.3.1, unilateral decisions such as suicide require less communications and storage overhead.

On the other hand, unilateral mechanisms can readily be abused. This is why suicide is made costly for the initiating device. Consequently, suicide require good nodes to value the social welfare of the network over individual utility. This condition is reasonable whenever the nodes are deployed by a single entity (e.g., a sensor network deployed on a battlefield) but may be less so when nodes are individually controlled (e.g., a peer-to-peer file-sharing system) [44].

Suicide enables precision denial-of-service (DoS) attacks since adversaries can remove any node. Network topology differences increase the importance of some nodes due to their location or number of neighbors. Even unsophisticated attackers can wreak havoc by taking out high-value nodes with low-value-node suicides.

But suicide is arguably less susceptible to DoS attacks than threshold voting schemes. Threshold voting schemes become totally vulnerable once the attacker gains sufficient numerical advantage (exceeding the threshold) in a region. Here the adversary can vote out all good nodes in the area. This is of particular concern when devices are mobile as an attacker can use the minimum number of compromised devices, moving them around the network and ejecting good nodes unchallenged. Suicide, by contrast, bounds the

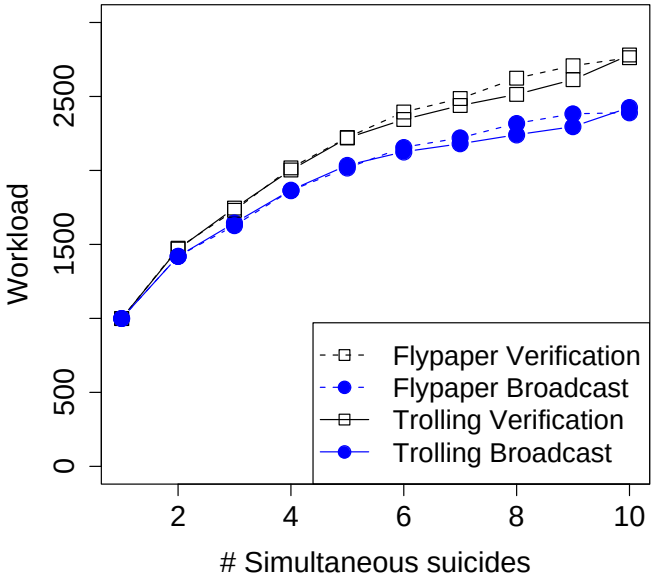


Figure 4.2: Workload versus simultaneous suicides.

maximum amount of damage a set of malicious nodes can do but requires twice as many nodes to be removed from the network – one good node for every bad node.

4.3.3 Quantifying suicide abuse

While protections against flypaper and trolling attacks ensure that only one good node is sacrificed per bad node, reconciling several suicide notes due to these attacks triggers increased communication and computational complexity. An attacker may still attempt flypaper or trolling attacks to consume resources (e.g., battery life or network capacity) by forcing multiple offers to be resolved.

We quantify the increased workload due to these attacks compared to the transmission of a single suicide note. We use simulation since this analysis probabilistically depends on the topology of the network as well as the random back-off period. We consider three scenarios of increasing complexity: normal operation where only a single suicide note is issued; a simple collision where two suicide notes are issued simultaneously; and a trolling attack where misbehavior is presented to nodes across the network simultaneously in the absence of node-replication detection to trigger multiple suicide offers. We compute two quantifiable measures: the number of broadcasts attributed to all suicide offers and the number of signature verifications attributed to all suicide offers.

Using the same parameters as in Section 3.2, we simulated a wireless network comprised of 1000 nodes uniformly distributed over a plane, where the communication radius of nodes ensures an average of 60 immediate neighbors within communication range. We averaged our results over 10 iterations on a network sample. Each suicide note is embedded with a

timestamp. A node rebroadcasts a received suicide note to its immediate neighbors if it is either the first one received, or has the earliest timestamp. In this way, suicide notes are propagated throughout the network until the note with the earliest timestamp completely dominates.

When one suicide note is broadcast, every node in the network broadcasts and verifies once. Two simultaneous suicide notes increases the workload by approximately 50%, a manageable rise. But what is the effect of additional simultaneous suicides? Figure 4.2 plots workload (number of broadcasts and verifications) during trolling and flypaper attacks as a function of the number of simultaneously issued suicide notes. As expected, the computational and communication burden increases. Notably, the function is mainly marginally decreasing, so that additional suicides increase the workload less than previous ones. At most, resource-consumption attacks increase system workload by a small multiple and can therefore be managed under most circumstances.

4.4 Conclusion

A major challenge for ad-hoc networks is removing nodes that are observed to be behaving badly. Existing threshold voting proposals for node revocation enfranchise too many of the wrong nodes, undermining their efficiency and security. They are susceptible to manipulation, particularly if nodes are mobile.

So we switched from voting against bad nodes to a protocol where good nodes reelect each other at regular intervals. Reelection reduces storage costs by shifting the responsibility of verifying votes from a node's neighbors to the node itself. This is a significant improvement, but simple reelection remains infeasible for severely constrained devices. We then proposed a lightweight reelection mechanism requiring no pre-assigned storage and using just hash operations: each node broadcasts a buddy list of trusted neighboring nodes locally at each time period. This can support a much wider range of membership strategies and is significantly cheaper. However, some communication costs remain, and like the other voting-based mechanisms, buddy lists struggle with node mobility and uneven network topologies.

We then showed that the most effective revocation scheme in general ad-hoc networks is suicide. A node observing another node behaving badly simply broadcasts a signed message declaring both of them to be dead. This is cheap; it scales well; it is not affected much by mobility; and it works across interesting parameter ranges. Such strategies are well known in nature, from bees attacking an intruder to the operation of helper T-cells in the immune system. They even echo some human societies, such as the dueling culture of the eighteenth century and the US Wild West. We believe that suicide is attractive for a wide range of distributed system applications.

Chapter 5

Excluding bad devices from vehicular networks

We now turn attention to vehicular networks, another promising application of decentralized wireless networking. Recently, consortia of automobile manufacturers in the US [158], Europe [40] and Japan [3] have begun investigating ways to equip vehicles with wireless radios for communicating safety information. For instance, cars might send each other collision warnings or traffic congestion notifications.

Vehicular networks inherit many of the same challenges faced by WSNs: distrust of other devices, expense of communications, and threat of node subversion. In some respects, the task of protecting vehicular networks is easier since cars are equipped with processors capable of some public-key cryptography, and certification authorities are likely to be available. In other ways, though, the circumstances are even tougher than those for WSNs: devices are highly mobile, interactions are short-lived, and delays in handling errant behavior cannot be tolerated.

Ensuring the integrity of safety communications is paramount. Compromised transmitters might send bogus information for reasons that are selfish (e.g., pretending there is an automobile accident to divert traffic away from the chosen path and enjoy an uncongested ride) or malicious (e.g., faking location information to encourage collisions). Alternatively, the transmitters may simply be broken, which is a less sinister but entirely plausible threat to message integrity.

Whenever a device starts sending bad information, the long-term solution is for the certification authority (e.g., the Department of Motor Vehicles) to revoke the credentials of the offending device. However, this process takes time, from the collection of evidence to the resolution of disputed claims. In the interim, ongoing attacks could endanger passenger safety. Thus, there is a need to rapidly isolate such errant devices and prevent them from spreading incorrect data. One solution is for the cars observing misbehavior to temporarily exclude the responsible bad device until the certification authority is notified

and takes appropriate action. In this chapter, we consider ways to reach such a local decision while at the same time maximizing efficiency and security.

We first describe an already proposed local decision mechanism, called LEAVE, where nodes vote to exclude errant devices by exchanging signed claims of impropriety [136]. We then propose a new protocol, called Stinger, in which a node can unilaterally remove a perceived misbehaving neighbor by limiting its own participation. Stinger is a tempered adaptation of the suicide protocol proposed for ad-hoc networks in Chapter 4.

We then set out to contrast the LEAVE and Stinger mechanisms, finding that they often exhibit complementary security and performance properties. This comparison is done through extensive simulations and involves a detailed comparative framework which supports varying attacker capabilities, characteristics of detection mechanisms, and traffic conditions. We demonstrate the circumstances under which voting-based LEAVE and unilateral Stinger perform best.

5.1 Vehicular networks

We now describe the operational characteristics of vehicular networks.

5.1.1 System model

Existing automotive authorities are likely to become certification authorities (CAs). Each would be responsible for the identity management of all vehicles registered in its respective geographic region. Vehicles register with exactly one CA. Each node has a unique identity, a pair of private and public cryptographic keys, and a certificate issued by the CA.

Messages are transmitted periodically, e.g., every 0.3 s for safety messages, or triggered by in-vehicle or network events. Most traffic is broadcast to limited regions of the network. All safety-related messages include the time and geographical coordinates of the sender, in addition to other application-specific information. Each message is also signed and accompanied by the sender's certificate. It is widely accepted that asymmetric cryptography is feasible for vehicular networks [135].

Safety messages may need to propagate across multiple hops. In this case, they are signed and include the coordinates and timestamp of the last relaying node, along with the originator's signature, coordinates and timestamp. This chain of signatures helps ensure the freshness of the information while limiting the propagation of illegitimate information. A received safety message is discarded if the difference between its timestamp and the timestamp of the receiver is greater than a system-specific constant accounting for clock drift, propagation and processing delays. Moreover, a message is discarded (by a receiver) if the coordinates of its sender/relay indicate that the receiver is outside the sender's

maximum nominal wireless communication range. These validations are applied at each hop.

At the data link layer, the Dedicated Short Range Communications (DSRC) protocol [22], currently being standardized as IEEE 802.11p, provides transmission ranges of typically 300 to 1000 m, with data rates in the 6-27 Mbps range. Beyond DSRC, vehicular networks could also leverage other wireless communication technologies. In this chapter, we assume that 802.11p is used.

A subset of network nodes form the infrastructure, comprised of the short-range DSRC base stations and mobile units. The latter include public safety vehicles (e.g., highway assistance and fire-fighting vehicles), police vehicles, and public transport vehicles (e.g., buses, trams). Infrastructure nodes serve as the gateway of the CA to and from the vehicular network; the connection of the CA to the static infrastructure nodes is over wired secure links. However, we do not assume that the CA must be accessible from the vehicular network at all times. The LEAVE and Stinger exclusion mechanisms are carried out by ordinary vehicles, not infrastructure nodes.

5.1.2 Threat model

An adversary may control a number of nodes that deviate from the legitimate vehicular network protocols. Nodes can also be faulty due to equipment failures. A detailed discussion of adversary and fault models is given in [131]. As our proposed mechanisms apply to both misbehaving and errant devices, we use both terms interchangeably. We emphasize that we are concerned with misbehaving nodes equipped with valid credentials.

We consider two types of attacker strategy. *False information dissemination* may be a very effective attack, when compared to deviations from networking protocols. The motivation for false information dissemination attacks may be malicious (e.g., sending fake braking information to trigger an accident) or selfish (e.g., claiming an accident has occurred to clear congestion). The adversary could either manipulate the sensory inputs or compromise the protocol stack and the computing platform [131]. An attacker may also control incoming communication, e.g., selectively erasing messages received by its on-board platform.

A second attacker strategy is *exclusion mechanism abuse*. In the next section, we discuss two proposals for excluding bad devices from participating on the network. These strategies, along with any other mechanism that attempts to exclude bad devices, may be abused by an adversary trying to remove good devices instead of bad ones.

5.2 Excluding errant devices

In this section we describe the LEAVE and Stinger mechanisms.

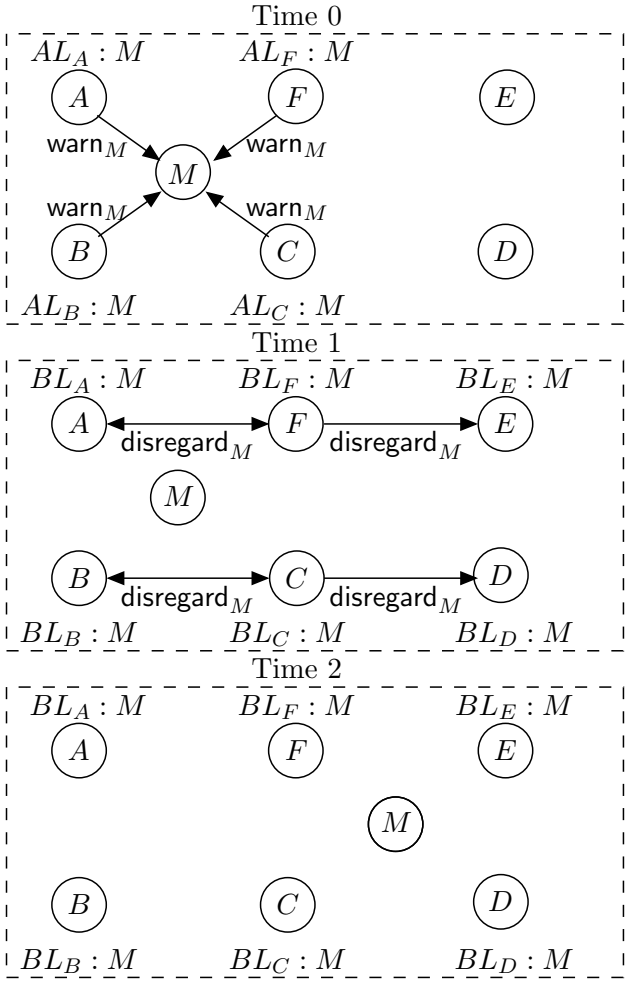


Figure 5.1: The LEAVE protocol. Vehicles A, B, C, and F accuse vehicle M and put it in their respective accusation lists AL. Once the exclusion threshold is reached, disregard messages are broadcast. M is then added to the blacklist BL of all accusing vehicles plus the vehicles receiving disregard messages (D and E). At time 0, D and E are not in M's transmission range and cannot detect its misbehavior.

5.2.1 LEAVE

LEAVE (Local Eviction of Attackers by Voting Evaluators) [136] is illustrated in Figure 5.1. Vehicles detecting an errant device broadcast *warning* messages to all vehicles in range. Any vehicle receiving a warning message adds the warned device to an *accusation list*. Once enough warning votes against a node are collected, its identifier is added to a local blacklist kept by each node. After nodes are added to the blacklists, additional *disregard* messages are repeatedly broadcast to the local neighborhood instructing the receiving nodes to ignore the attacker's messages. Hence, vehicles using LEAVE can be made aware of bad vehicles before interacting with them. Finally, the evicted nodes are reported to the CA once within reach of an infrastructure node.

Deciding when to warn a node using LEAVE is actually more subtle than surpassing a

simple numerical threshold of warning votes. Rather, it is based on exceeding an *exclusion quotient*, a sum of weighted accusations relative to the size of a vehicle's neighborhood (the LEAVE paper used an exclusion quotient of 0.5). The exclusion quotient discounts accusations from users who have themselves been accused by others, as proposed by Crépeau and Davis in [41]. For disregard messages, a simple threshold is used (the LEAVE paper used a threshold of 4 votes). To demonstrate their legitimacy, disregard messages include supporting signatures from this threshold of users.

LEAVE requires an *honest majority* of nodes for the neighbors of every good node to guarantee its security. If the number of attacker-controlled devices exceeds the threshold for sending disregard messages, they can quickly eject any device at will.

5.2.2 Stinger

Chapter 4 proposes several strategies that enable nodes to remove compromised devices from an ad-hoc network. We now discuss how one strategy already presented – *suicide attacks* – can be adapted for use in vehicular networks.

Recall from Section 4.2 how the suicide mechanism works. Upon detecting a node M engaging in some illegal activity, node A sends a *suicide note* $\mathsf{suicide}_{A,M}$ with the identities of both A and M. The other nodes now disregard both A and M.

The environmental assumptions considered in Chapter 4 do not directly correspond to those present in vehicular networks. The modified mechanism is called Stinger, and suicide notes are called stings. Stinger deviates from suicide in the following respects:

1. Stinger temporarily prohibits devices from transmitting messages, but allows them to continue receiving and forwarding messages;

2. Stinger allows multiple good nodes to be ignored by a smaller number of devices in order to exclude a single bad node;

3. Stinger permits good devices to continue accusing bad ones even after they have issued one sting (the sting is only accepted as valid by devices that did not observe the prior sting).

We now describe each of these changes, and their motivation, in greater detail.

First, the original suicide mechanism proposed permanent ejection from participating on the network. Such harsh punishment is inappropriate for vehicular communications that transmit safety information. By contrast, temporary removal could be used to rapidly ignore an errant transmitter. Since most interactions are short-lived, temporary removal is equally effective in tackling misbehavior as it happens without inhibiting communication occurring much later. While the sting instruction prevents the bad and good device from

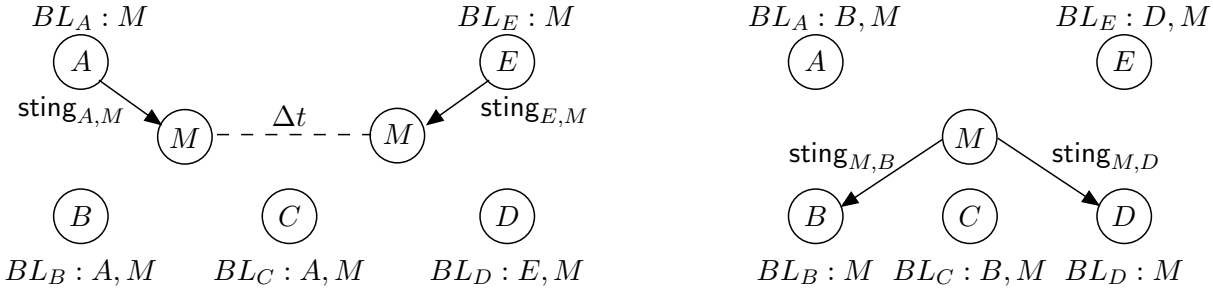


Figure 5.2: Multiple stings for bad node M as it moves over time (Left). Bad node M stings two good nodes B and D simultaneously (Right). Each good node adds M and either B or D to its local blacklist. Node C, which observes both stings, adds B but not D.

sending out additional warnings to cars the good device has already warned, both still *receive* safety instructions from other cars. This minimizes the noticeable impact on the sacrificing driver while still penalizing a malicious device.

Indeed, we anticipate that drivers will remain completely unaware of any execution of the exclusion mechanism, be it LEAVE or Stinger. The interface presented to the driver should only include notifications derived from processing received messages. Consequently, drivers are unlikely to be tempted to abstain from issuing stings whenever the car's software deems it necessary.

Second, the original suicide mechanism assumed a completely connected network. Suicide notes were broadcast throughout the network so that just one good device is removed for each bad device. Vehicular networks will be comprised of disconnected islands. High-traffic areas in cities remain separate from each other and from highways in between. Furthermore, connections are ephemeral: cars on a motorway may only be in communication range for a few seconds. Thus, it is impractical to transmit stings across a country in a short time. Instead, stings must remain localized, rebroadcast at most a few times. This keeps the response quick and minimizes communications overhead. It also means that there will be times where more than one good node has sacrificed itself for the same bad node. Yet the impact is still limited: rather than removing one good node for one bad node, several nodes may be independently removed for a single bad node. Since good nodes maintain a local blacklist, they only ignore the first sting's sender for each accused device. Crucially, no single device will ignore two honest nodes for the same bad node.

Figure 5.2 (left) illustrates how the Stinger protocol works as cars move. Bad node M is detected by A, which broadcasts $\mathsf{sting}_{A,M}$ to instruct vehicles near A to ignore M. Hence, nodes B and C add both A and M to their local blacklists, while D and E do not because they did not receive $\mathsf{sting}_{A,M}$. As M moves into range of D and E, E issues a new removal for B, $\mathsf{sting}_{E,M}$. D adds E and M to its local blacklist, but C does not because it has already ignored M from A's sting.

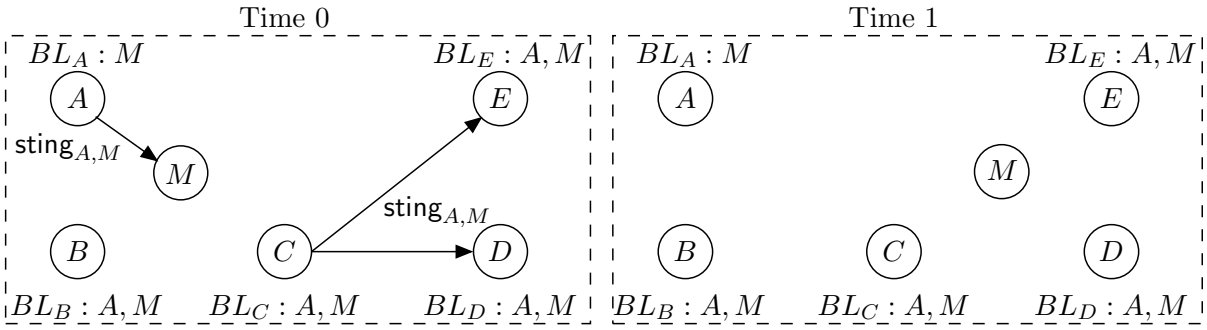


Figure 5.3: Rebroadcasting A's sting against M.

An attacker can also issue many stings against different neighbors. In Figure 5.2, bad node M stings B and D simultaneously. The impact is the same as for the previous example: M is ignored by everyone, and one good node (either B or D) is ignored. While Stinger can be abused, the abuse is confined to adding a single identity to good nodes' local blacklists.

This discussion motivates the third difference between suicide and Stinger: good devices continue to accuse bad ones even after they have issued one sting. Devices receiving the earlier stings ignore the later ones – subsequent stings are intended for and only accepted by devices that did not observe the earlier stings. This change is necessary to prevent a so-called *motorway attacker* who widely broadcasts misbehavior and moves around quickly to attract many stings and prevent good nodes from excluding subsequent attackers.

Sting messages are locally transmitted, and they may also be rebroadcast to warn devices in case the bad device later moves in other directions. The effect of sting retransmission is shown in Figure 5.3. At time 0, bad node M is detected by A which transmits $\mathsf{sting}_{A,M}$. Nodes B and C then retransmit the message, notifying D and E. When M moves near to D and E at time 1, M is already ignored by them.

So what is the cost of Stinger besides message and transmission overhead? Good devices can successfully warn their neighbors of only one bad device; any subsequent stings for other devices will only be adhered to by devices that did not observe the first sting. In Figure 5.4, bad nodes M_1 and M_2 are present in different areas. Nodes A and E issue stings to locally remove them. However, when M_1 moves into the area previously occupied by M_2, E is powerless to warn its neighbors. E may try to remove M_1 using another sting, but it has no effect since its neighbors already ignore E's messages. So F is left to issue sting_{F,M_1}. In the following analysis, we quantify the adverse impact of excluding honest devices by measuring any delays introduced when removing bad devices.

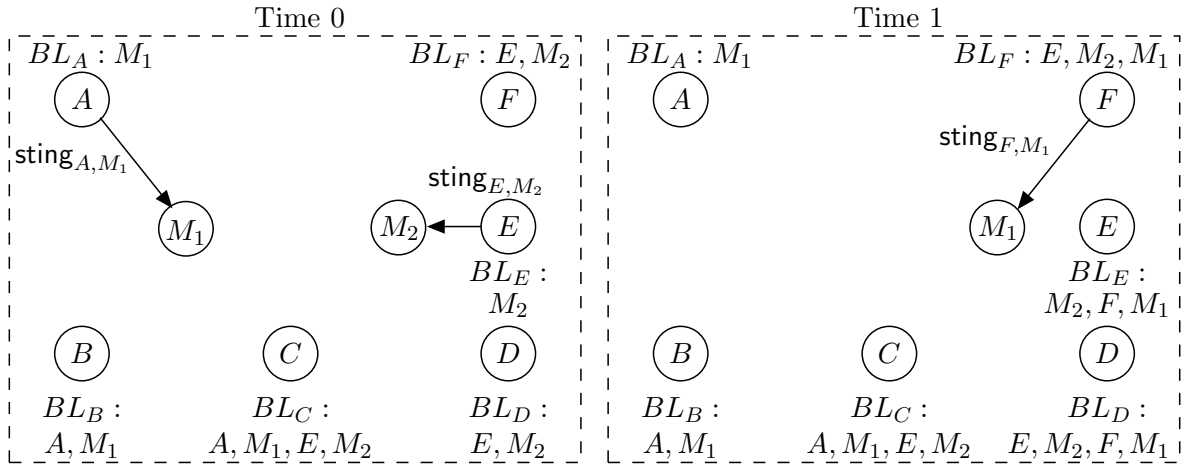


Figure 5.4: Adverse impact of Stinger. Two bad nodes M_1 and M_2 are removed by A and E; E cannot warn his neighbors about M_1 at time 1 since he has already done so for M_2.

5.3 Simulation framework

We evaluate the performance of LEAVE and Stinger under the stringent conditions required by vehicular networks. As both protocols rely on the ad-hoc operation of vehicles within short time delays, we simulate it using ns-2 [125] with the message access control layer parameters of IEEE 802.11p. Consequently, our simulation takes into account subtleties such as non-symmetric message reception and the timing of messages as devices move out of range. LEAVE is implemented with an exclusion coefficient of 0.5 and a disregard threshold of 4, as in the original LEAVE paper. Each simulation run lasts 200 seconds. While stings are designed to be temporary, for the purposes of the short simulation, their effects are treated as permanent.

We now describe how we realistically model dynamic traffic conditions, attacker behavior, and the operation of detection mechanisms. We identify key characteristics which we vary in the simulations in order to better understand their impact on LEAVE and Stinger's security and performance.

5.3.1 Modeling dynamic traffic conditions

To simulate different traffic conditions, we vary:

1. average vehicle speed,
2. average vehicle density.

Presently, we use a city traffic model proposed by Saha and Johnson [141]. In Sections 5.4.2 and 5.4.3, we simulate 150 vehicles traveling at 60 km/h on a 2.4 km by 2.4

km area modeled after a city. We then vary the density and speed of cars in Section 5.4.4. The ns-2 framework used for developing the simulations is available at [144].

5.3.2 Modeling errant behavior and its detection

In the original LEAVE paper [136], attacker behavior and detection is modeled in a simple fashion. Only one bad node participates in the system during simulations. Any device within transmission range of the bad node is deemed vulnerable to attack. Furthermore, good devices can detect bad ones as soon as they are within transmission range. We improve the simulation framework by generalizing the attacker model to allow for a more capable adversary, as well as arriving at a more realistic approximation of attacker impact and detection.

We also note that as part of its system model, the original LEAVE paper assumes an honest majority. In our simulations we allow for circumstances where this is not true. Even though the majority of all cars is likely to remain honest, it is quite reasonable for adversary-controlled devices to reach temporary, localized majorities.

To vary misbehavior, we tweak the following parameters:

1. the number of attacker-controlled devices,
2. false information dissemination versus exclusion mechanism abuse,
3. attacker impact range.

We allow the adversary to simultaneously compromise a number of devices. Attacker-controlled devices can cooperate and share information. Under *false information dissemination* attacks, the adversary attempts to cause accidents or divert congestion by sending fake safety messages. Malicious nodes disregard all bad messages originating from other nodes (i.e., they do not accuse other malicious nodes). Under *exclusion mechanism abuse*, the adversary additionally tries to disrupt the transmission of safety messages using the exclusion mechanism itself (LEAVE or Stinger). Malicious nodes falsely accuse all non-malicious nodes in communication range. As the nodes move, they discover who their new neighbors are and then vote against them. In this way, honest nodes may be implicated. Bear in mind that once an honest node has detected a malicious node as such, it ignores its votes. However, there may be a window of time where an undetected bad node can trick good nodes into believing that other good nodes are bad. Exclusion mechanism abuse may not be in the interest of an attacker whose aim is to remain undetected.

One of the most difficult aspects of simulating attack is determining which devices are made vulnerable by compromised devices. We approximate vulnerability to attack by proximity to compromised devices. The closer a device is to an errant transmitter the

likelier it is to be harmed. Hence, in our simulations we vary the maximum distance from a bad node where a good node is still vulnerable.

To model detection mechanisms, we vary these parameters:

1. range of revealed misbehavior,
2. false positive rate,
3. false negative rate.

These properties are general to all detection mechanisms, and we can vary each without restricting our simulations to using a particular type of detection mechanism.

The simplest approach (taken by the original LEAVE paper) assumes that good nodes can detect bad ones with 100% accuracy so long as they are within the bad node's maximum transmission range. This behavior does not correspond to how detection mechanisms might actually work. In many cases, misbehavior can only be detected much closer to an adversary (e.g., if another car's sensors can directly observe the other car or its environment). To account for this, we can restrict detection to nodes within a specified distance. Typically, the maximum range of detection will be less than the maximum range of attacker impact. For example, a car sending a fake crash warning message is likely to be detected by its immediate neighbors, but cars further away will still be affected precisely because they believe the misinformation and react to it.

Another unfortunate property of detection mechanisms is their susceptibility to error. A false positive occurs whenever the detection mechanism flags a good device as bad. Similarly, a false negative happens when the detection mechanism mistakes a bad device for a good one. In our simulations we vary the likelihood of false positives and negatives over time, since the detection mechanism is continuously operating. For simplicity, we vary the false positive rate per minute of interaction. So a false positive rate of 10% means that there is a 10% probability of a false accusation after interacting with other devices for one minute.[1]

5.4 Simulation analysis

In this section we compare the security and performance of LEAVE and Stinger while varying the parameters just discussed.

[1]To implement false positives and negatives in the simulation, we have to set a probability of error each time the detection mechanism returns a result, which for our simulations occurs every 300 ms, or 200 times per minute. For a false positive rate of 10% per minute of interaction, for example, we trigger a false positive with probability $\frac{0.1}{200}$ each time the detection mechanism operates.

5.4.1 Security and performance metrics

We compute three metrics:

1. average time good devices are vulnerable,
2. average percentage of good neighbors that are ignored,
3. average number of messages received per device.

The first two metrics describe the security properties of the protocols, while the third is used to compute overhead.

Most envisioned attacks on vehicular networks are time-critical – they spread misinformation that causes a car to quickly make an incorrect decision. Hence, bad devices must be detected and removed very quickly. We measured the average time good devices are vulnerable to attack by a bad node. Here, vulnerability is defined as being within transmission range of an unblocked bad device.

The second security metric we measure is the average percentage of good neighbors that are ignored due to the exclusion protocol. Each ignored neighbor reduces the number of devices that can transmit safety information, as well as participate in the Stinger or LEAVE exclusion mechanisms. This metric is usually more important for Stinger, since good devices forgo participation in order to remove bad devices. However, it is still possible for good nodes to be excluded in LEAVE (due to accidental or malicious voting against good devices).

The final metric offers a useful comparison of the work required for each strategy, since each message requires a cryptographic verification operation.

5.4.2 Detection mechanism parameters

The detection mechanism's capabilities can directly impact the effectiveness of the exclusion mechanism. To demonstrate this, we computed the security metrics while varying the detection mechanism's characteristics under the assumption that only one device is compromised (see Figure 5.5).

Figure 5.5 (top left) shows how the time devices are vulnerable varies depending on the maximum distance for which bad devices can be detected. Devices can transmit up to 300 meters. Ideally, we would like the detection mechanism to trigger as soon as a good device comes within this transmission range. In this case, devices are vulnerable for 2.9 seconds before ignoring bad devices using LEAVE. Unsurprisingly, Stinger is faster (0.6 seconds with no rebroadcasts).

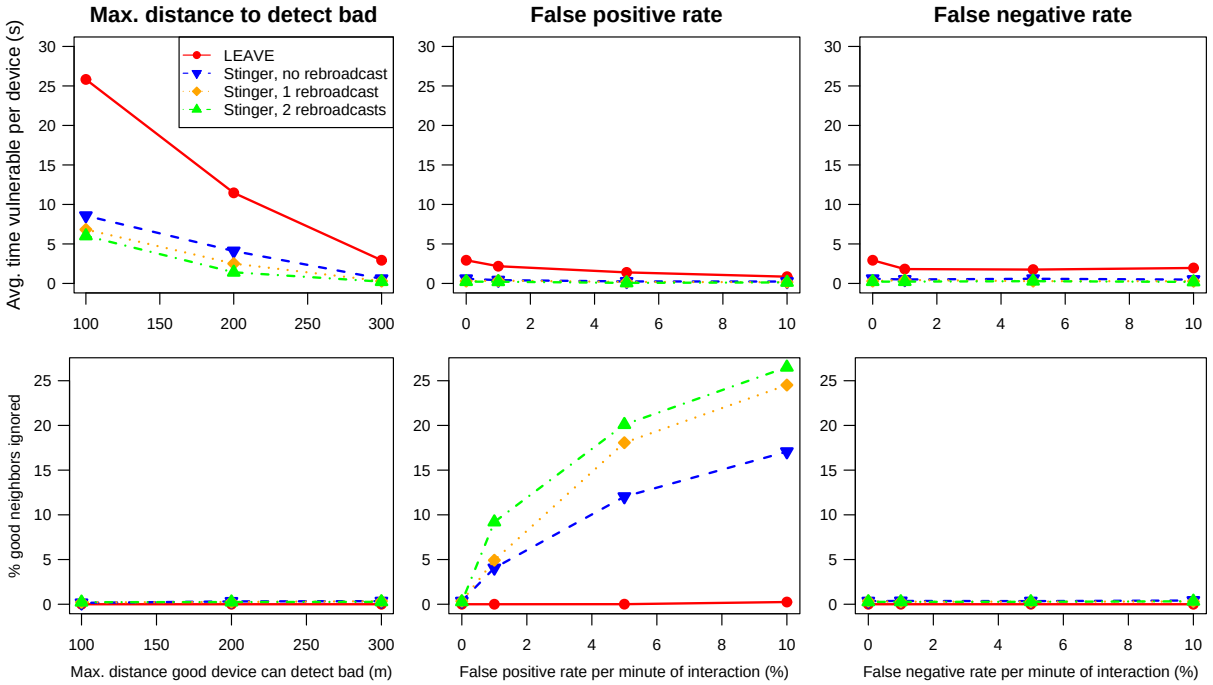


Figure 5.5: Security and performance costs introduced by imperfections in the detection mechanism.

As we reduce the maximum range for which misbehavior can be detected, the time exposed increases. For LEAVE, the vulnerable time increases fastest, to 11.4 seconds for 200 m and 25.8 seconds for 100 m. For Stinger without retransmission, the lag is 4.1 seconds for 200 m and 8.6 seconds for 100 m. Increasing the number of rebroadcasts further minimizes the time exposed at the expense of additional message overhead. Both LEAVE and Stinger do shorten the overall exposure time, which is 29.8 seconds. Notably, LEAVE barely helps when detection is not very good (25.3 seconds exposure for 100 m maximum detection range).

Varying the maximum detection range has no impact on the proportion of good neighbors ignored (Figure 5.5 (bottom left)). This is not surprising, given that reducing the detection range only enables more bad devices to go undetected, which does not trigger false accusations against good nodes. For Stinger, just 0.3% of a node's good neighbors are ignored on average. This proportion is so small because the devices issuing stings are likely to change neighbors frequently. Increasing the false positive rate, by contrast, does cause good devices to ignore each other more often. Figure 5.5 (bottom center) shows that the percentage of good neighbors ignored increases significantly as false positives are more likely. Notably, false positive rates of up to 10% do not cause problems for LEAVE, so long as false positives are not correlated.

Recall from Figure 5.5 (top left) that the time vulnerable to attack decreases as stings are rebroadcast. In Figure 5.5 (bottom center), the ordering is reversed. Rebroadcasting

stings causes more good devices to be ignored. With a 5% chance of false positive per minute of interaction, 12% of a device's good neighbors are ignored when stings are not retransmitted, 18% are ignored with one retransmission and 20% ignored with two retransmissions. Hence, there is a direct trade-off between speed of excluding bad devices and the number of good neighbors ignored.

The most noteworthy observation from the two graphs in Figure 5.5 (right) is that increasing the false negative rate to 10% has almost no impact. While not shown in the graph, the same holds as the share of attacker-controlled devices increases to 10%.

5.4.3 Adversary strategies

We now vary adversarial capabilities and strategies (Figure 5.6). The left-hand side graphs measure the effects of false information dissemination (i.e., a strategy that does not attempt to abuse the exclusion mechanism), while the right-hand side measures exclusion mechanism abuse where the goal is to cause as much damage using the exclusion mechanism as possible. In both cases, we set the maximum distance for detecting bad devices to 200 m with no chance of false positives or negatives.

We simulated situations where the adversary controls from 1 device up to 15, which is 10% of all devices. This does not mean that an attacker has compromised 10% of an entire country's vehicles, which is unrealistic for all but the strongest adversaries. Rather, compromising 10% of the vehicles in a localized region (in our simulations, a 5.8 km^2 area) is quite reasonable for relatively capable adversaries.

As the proportion of attacker-controlled devices increases, the time each good device is vulnerable increases (Figure 5.6 (top)). Consistent with Figure 5.5 (top left), LEAVE keeps devices vulnerable for longer than Stinger does. Note from Figure 5.6 (top right) that the vulnerable time actually decreases when the attacker is actively abusing Stinger, since maliciously transmitting stings instructs good devices to ignore the bad one. False information dissemination is preferred if the attacker's aim is to remain undetected.

By contrast, Figure 5.6 (middle) and (bottom) explain why an attacker might prefer abusing the exclusion mechanism. When using a false information dissemination strategy, the proportion of good neighbors ignored remains small: 2.8% of good neighbors are ignored using Stinger without rebroadcasts when 10% of devices are controlled by the adversary (Figure 5.6 (middle left)). Under an exclusion mechanism abuse strategy, the proportion of ignored good neighbors jumps to 21.2% (Figure 5.6 (middle right)). This proportion is still much smaller than the worst-case scenario. With 10% attacker-controlled devices, there could be more bad devices present than a good device has neighbors. Hence, we might expect all devices to be ignored, but the simulations do not bear this out.

Why not? In the worst case for collateral damage caused by active Stinger abuse, an attacker continually broadcasting stings against every neighbor can trick every good device

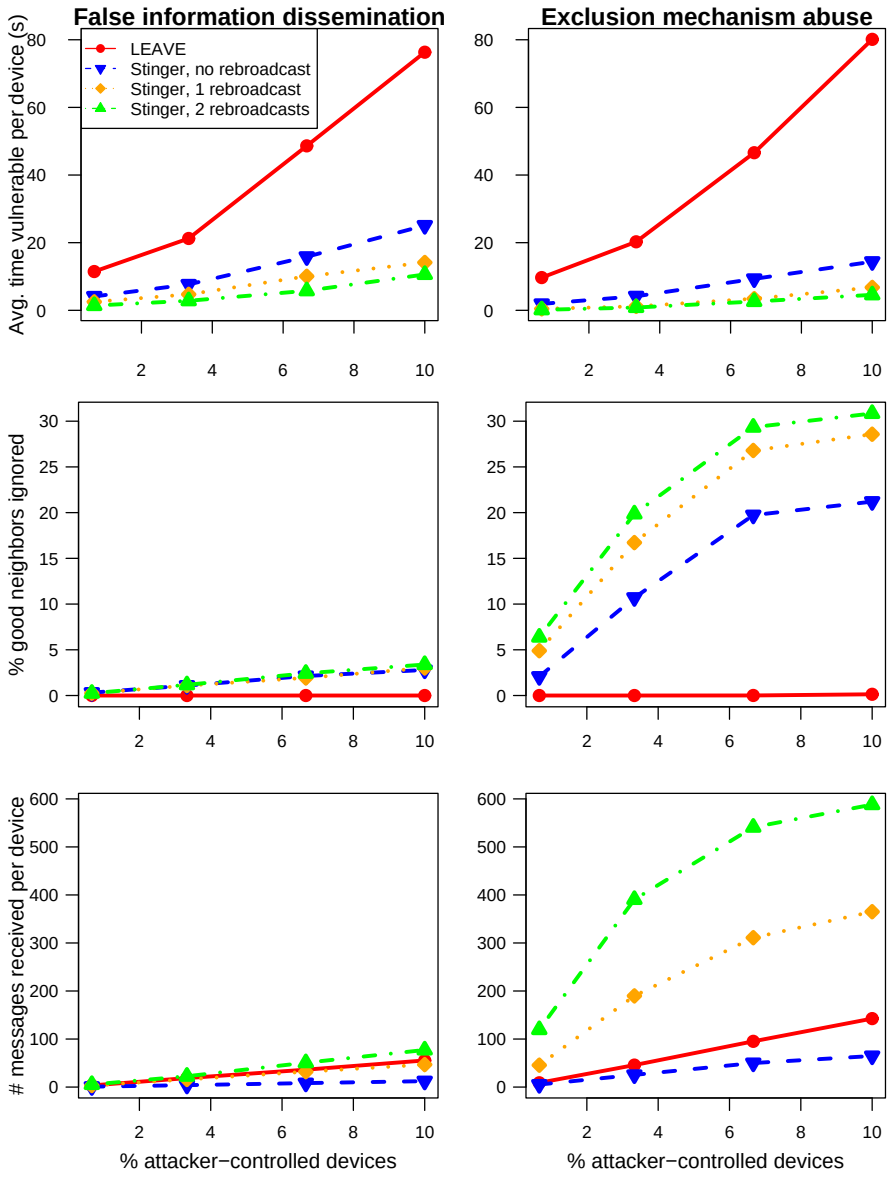


Figure 5.6: Security and performance costs as the attacker controls more devices. The left-hand side graphs measure the false information dissemination strategy, while the right-hand side graphs measure exclusion mechanism abuse.

to ignore one honest neighbor. Conceivably, this one ignored honest neighbor could remain in communication range permanently. This is hard to achieve in practice since cars move and naturally change neighbors. Furthermore, it is very difficult for an attacker to locally interact with every other car to determine a legitimate neighbor to remove. Therefore, it is not surprising that the proportion of ignored good neighbors plateaus as the attacker controls more devices, as indicated in Figure 5.6 (middle right).

As mentioned in Section 5.2.1, the worst-case scenario for abusing LEAVE is a roaming pack of attackers. So long as the number of attackers exceed the voting threshold, they can quickly eject any device at will. We did not simulate this situation because the out-

come is clear: all good devices are removed. Instead, we simulate the situation where attacker-controlled devices vote against their neighbors continuously, but in an uncoordinated fashion. As can be seen from the simulations, uncoordinated malicious voting is completely ineffective.

Figure 5.6 (bottom) compares the number of messages received when using LEAVE and Stinger for a range of rebroadcasts. So long as the adversary is not attempting to abuse the eviction mechanisms, Stinger requires fewer messages than LEAVE whenever stings are only rebroadcast one time, or not at all (Figure 5.6 (bottom left)). Beyond that, LEAVE is more efficient in terms of message overhead.

When abusing the exclusion mechanism, however, the impact on overhead changes dramatically (Figure 5.6 (bottom right)). Stinger without rebroadcasting remains the most efficient strategy, but adding rebroadcasting leads to a huge increase in overhead. With 10% of the devices under adversary control, Stinger without rebroadcasting requires each device to receive 65 messages, compared to 311 with a single retransmission and 588 with two retransmissions. This dramatic difference provides further evidence that using Stinger without retransmissions is the best approach.

5.4.4 Traffic conditions

The analysis so far has considered a single, typical traffic scenario. But what happens when traffic conditions change? We varied both the density of traffic and the average speed of vehicles.

Figure 5.7 plots the results with 10 bad devices present and a 200 m maximum distance for detecting bad devices. Increasing the density of traffic has a negative effect on LEAVE and a slightly positive effect on Stinger. Surprisingly, the average time devices are vulnerable increases as more cars are present when using LEAVE (Figure 5.7 (top left)). Under Stinger, by contrast, the time vulnerable decreases slightly as the density increases.

The number of messages that must be processed also increases significantly under LEAVE, while remaining steady when using Stinger (Figure 5.7 (bottom left)). This is because the number of other cars within communication range of each vehicle increases under higher densities. More neighboring vehicles means more broadcast warning messages from each nearby device. Since only one vehicle (or a few in the event of a collision) issues a sting for bad devices, there is no increase in overhead with increased traffic.

The right-hand side graphs in Figure 5.7 measure the effects of increasing the speed of vehicles, and therefore, reducing the average connection time between vehicles. From these graphs, it appears that speed does not have a significant impact on the security or performance of Stinger and LEAVE.

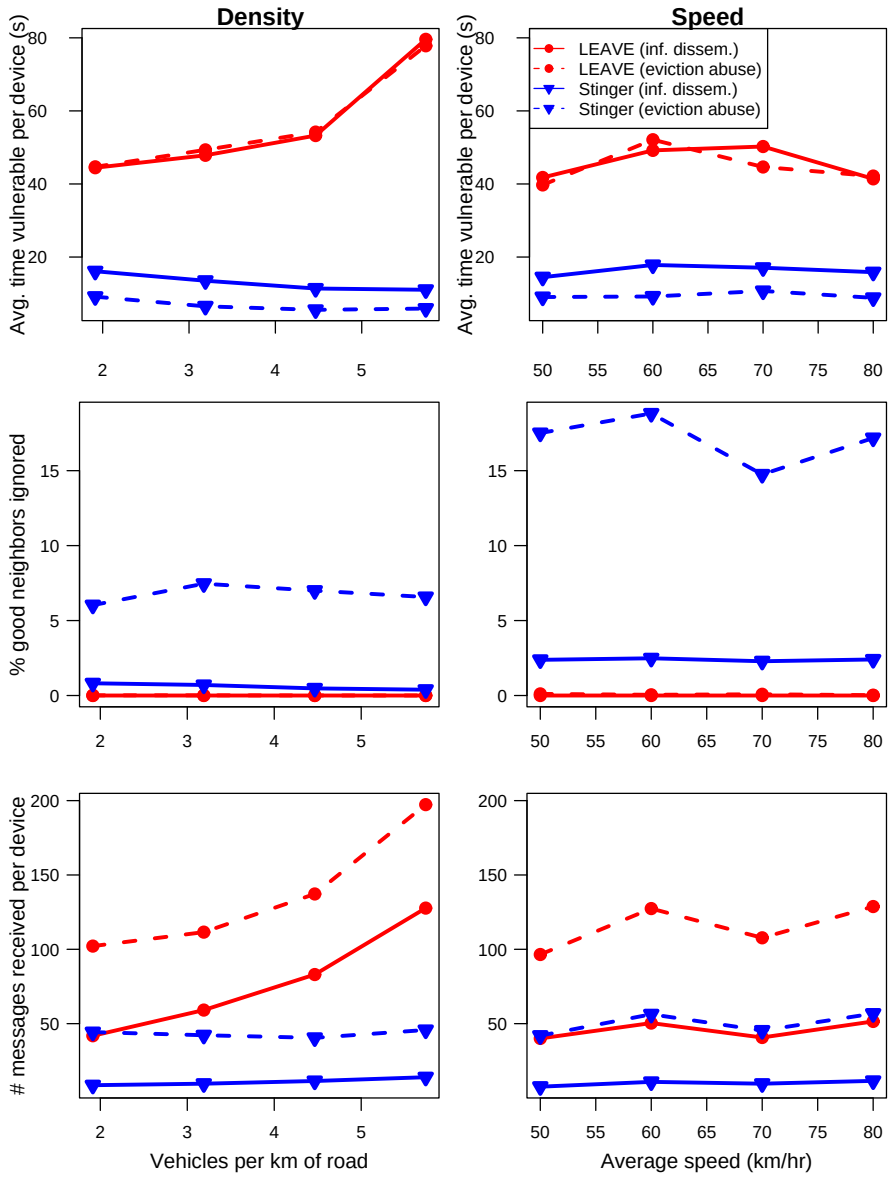


Figure 5.7: Security and performance costs while varying traffic conditions.

5.4.5 Verdict

The analysis in this section has identified a number of trade-offs between LEAVE and Stinger for different scenarios. To summarize:

- LEAVE is more resilient to non-zero false positive rates.
- Fewer good nodes are ignored using LEAVE than Stinger when an attacker abuses the exclusion mechanism.
- Stinger excludes bad devices faster, leaving a shorter vulnerability window.
- Stinger handles increasing proportion of attackers better than LEAVE whenever the attacker's goal is to avoid detection.

- Stinger scales better as the density of vehicles increases.

Stinger performs better than LEAVE for most circumstances. In the situations where LEAVE outperforms Stinger, Stinger's performance remains relatively satisfactory. For example, while ignoring 20% of good neighbors when 10% of the cars are actively abusing Stinger is undesirable, safety messages should still be propagated via the remaining 80% of recognized neighbors. By contrast, some circumstances where Stinger outperformed LEAVE are critical. Notably, Stinger is significantly faster than LEAVE when removing bad devices under every condition we tested. Speed of removal is critical to limit the transmission of misinformation. We therefore conclude that Stinger is better suited than LEAVE for excluding errant devices in vehicular networks provided that false positives can be minimized.

5.5 Related work

The literature on vehicular networks already contains methods for detecting bad devices. For example, Leinmüller *et al.* propose threshold-based tests to verify positioning information in vehicular networks [97]. In [67], a general framework for detecting malicious data detection compared received data to a vehicular network model.

Techniques for removing bad devices from a network often fall under the broad category of *revocation*. Revocation has been considered mostly in the context of the wired Internet and the design of Public Key Infrastructure (PKI) services [76], although it is also important for WSNs (as outlined in Chapters 2–4 of this thesis). Nevertheless, the design of mechanisms to disseminate the revocation information across systems similar to vehicular networks has not been considered in the wired Internet context (for a survey and discussion of trade-offs see [165, 172]). Due to network volatility and scale, the overhead of querying a server to obtain timely revocation status, assuming the server is reachable, could be impractically high. For the same reasons, schemes that distribute the load of a server to a set of participating clients [166] by redundantly forwarding revocation information would not be practical for deployment within the vehicular network, but only meaningful behind the fixed infrastructure.

Most existing works on vehicular network security [132, 169, 135] have proposed the use of a PKI and digital signatures but do not provide any mechanisms for certificate revocation, even though it is a required component of any PKI-based solution. In the context of vehicular networks, the IEEE 1609.2 Draft Standard [79] does refer to certificate revocation. It has proposed the distribution of CRLs and short-lived certificates, but does not elaborate how to achieve this. The paper proposing LEAVE [136] also described a more comprehensive revocation strategy that leverages the infrastructure to efficiently

distribute revocation lists from CAs. In this context, LEAVE is a fast, temporary exclusion mechanism which triggers a slower, permanent revocation by the CA. Stinger can also be used for this purpose.

In this chapter, we have distributed the task of temporarily excluding bad devices to untrusted vehicles to improve timeliness, while keeping the CA's substantial responsibilities centralized. Others have distributed the CA's responsibilities in different contexts. Zhou and Haas investigated CAs for use in mobile ad-hoc networks, distributing their functionality across a number of servers [173]. However, this scheme does not consider the problem of revocation, especially in a highly mobile environment like a vehicular network. Splitting up CA responsibilities over impromptu coalitions of devices (e.g., [41, 87]) is similar in motivation to the voting structure of LEAVE. However, threshold cryptography is of limited use whenever the voting coalitions are as dynamic as the short-lived neighbors in a vehicular network.

5.6 Conclusion

In this chapter, we compared two decision mechanisms, LEAVE and Stinger, for excluding misbehaving or faulty devices from vehicular networks. LEAVE corresponds to the blackballing mechanism for WSNs, while Stinger is adapted from the suicide mechanism proposed in Chapter 4. We applied them in the context of vehicular networks where fast exclusion is both critical and hard to achieve, given the ephemeral properties of the environment. Based on a detailed simulation analysis, we found that both protocols have unique advantages and disadvantages. In particular, Stinger is faster than LEAVE and scales better with increasing traffic density, but LEAVE is more resilient to false positives and to higher percentages of attackers abusing the exclusion mechanism. On balance, we conclude that Stinger is better suited than LEAVE to exclude errant devices in vehicular networks.

Chapter 6

Phishing attacks on the Internet

In the previous chapters we have identified attacks exploiting cooperation and designed collaborative decision mechanisms for decentralized wireless networks. Given the circumstances of these new applications, we simulated theoretical attacks and proposed the adoption of suitable countermeasures. But it is also important to study attacks already taking place, along with the defensive countermeasures being deployed. In the following chapters, we empirically analyze phishing attacks on the Internet. We find cooperation to be critically important in a number of real-world contexts, from evasive attacker strategies to information sharing between defenders.

Phishing is the criminal activity of enticing people into visiting websites[1] that impersonate the real thing, to dupe them into revealing passwords and other credentials to carry out fraud. Although a wide range of companies are attacked in this way, from businesses as diverse as online auctions (eBay), payments (PayPal), share dealers (E*Trade), social-networking (MySpace), gambling (PartyPoker) and online retailers (Amazon), the vast majority of attacks are against financial institutions. Hence for simplicity, within this thesis we use the term 'banks' for the firms under attack.

The academic work on phishing has been diverse, with a useful starting point being Jakobsson and Myers' book [81]. Researchers have tried to understand the psychology of the process [50], how to block the email containing the initial enticement [111], how server operators might automatically detect fraudulent sites [164], and whether there are patterns to their occurrence [140]. There have been many proposals for browser mechanisms to detect phishing websites [130, 171] and for schemes to prevent users from disclosing their secrets to them [138]. Others disseminate information about the trustworthiness of websites through central repositories (blacklists) or social networks [32], although it seems that users often ignore any cues that tell them that websites are likely to be malicious [142, 167].

In this thesis we consider phishing from a completely different angle. The banks are

[1] Throughout this thesis we use the terms 'site' and 'website' interchangeably.

dealing with the fake websites through 'take-down' procedures, so that there is nothing there for a misled visitor to see. Our aim has been to determine how effective this defensive strategy is, and whether it is likely to be sufficient on its own to prevent phishing from being profitable. Moreover, the take-down defense requires cooperation between many independent actors with conflicting motivations, from the banks impersonated to the ISPs inadvertently hosting the phishing websites. This chapter describes how phishing attack and defense works, along with the methodology used for collecting data on phishing websites.

6.1 The mechanics of phishing attacks

To carry out phishing scams, attackers transmit large numbers of spam emails which include links (URLs) to websites they control. The spam emails must resemble legitimate email, so that unsuspecting users might consider them genuine. The spam must also contain an appropriate message so that users choose to act upon it, be it an impending account suspension, a payment for a marketing survey, or a report of a transaction that the user knows to be fake and must therefore be canceled [50].

The user connects to a spoof website by clicking on a link in the email. Their web browser may access the website directly or be redirected from an initial site (perhaps via legitimate redirector systems at, for example, Google[2]) to the actual phishing pages. At this stage browsers may apply their own heuristics and consult their own blacklists to determine if the website should be blocked as clearly illegitimate. Provided the browser does not interfere, the user is then presented with an accurate imitation of the legitimate company's pages (often including all the links to warnings about fraud), and thus reassured fills in personal details. While a handful of attackers validate these details immediately, it is more common to accept any response.

The compromised details obtained are usually emailed to a webmail address, but are sometimes stored in plain text files at the spoof website, awaiting direct collection by the fraudster. Once they have received the compromised details, they discard obvious fakes and sell on the remaining details to cashiers who empty the bank accounts [154]. Money is often transferred using 'mules', who are recruited via further spam email seeking 'financial consultants' to accept and relay payments for a commission.

There are several reasons why a phishing attacker might impersonate many banks at the same time. First, it is very difficult to distinguish between the number of attackers and the number of instances of attack. This is problematic for banks since it creates the illusion that a law enforcement strategy is intractable. Second, victims of phishing attacks are

[2]In February 2007 Google started to detect usage of their redirectors and provide a warning message [39], so it is likely that other redirectors will now be used in preference.

restricted to only those banks being impersonated. A phisherman can cast a wider net by creating false web pages for lots of banks. Finally, each bank only attempts to remove the pages where it is the target. Consequently, an attacker can achieve a steadier income since banks are likely to independently respond at different times.

6.1.1 Ordinary phishing attacks

Phishing attackers typically impersonate a single bank per hosted website. Most commonly, the phishing pages are placed on a compromised machine – either a residential machine or a server in a data center. The hijacked machine could have come under the attacker's control either through a security vulnerability (typically unpatched applications within a semi-abandoned 'blog' or message board), or because the user is running some malware, delivered by email or downloaded during a visit to a malicious website.

The spoof website is sometimes hosted on 'free' webspace, where anyone can register and upload pages. Here, a typical URL would be `http://www.bankname.freehostsite.com/login` where the `bankname` is chosen to match or closely resemble the domain name of the financial institution being attacked.

Changing the hostname is not always possible for compromised machines, and attackers may have restricted permissions, so they add their own web pages within an existing structure, leading to URLs of the typical form `http://www.example.com/~user/www.bankname.com/` where, once again, the `bankname` is present to lend specious legitimacy should the user check which website they are visiting, yet fail to appreciate the way in which URLs are really structured.

Sometimes the '`example.com`' part of the hostname makes it unlikely that the URL will appear convincing. In this case, the URL may instead use the IP address of the compromised machine. To further allay suspicion, the fraudsters sometimes go to the effort of registering their own domain name. The domain is either pointed at free webspace or a compromised machine where the fraudster has obtained sufficient control of the web server configuration. The domain names are usually chosen to be a variation on `bankname.com` such as `bankname-usa.com`, or they might use the bank's name as a subdomain of some plausible, but superficially innocuous domain, such as `bankname.xtrasecuresite.com`. A half-way house to an actual domain name is the use of systems that provide domain names for dynamic IP address users, which results in domains such as `bankname.dyndns.org`.

6.1.2 Rock-phish attacks

We have just described the way in which typical phishing websites are operated with web pages added to existing structures and the occasional use of misleading domain names. However, the 'rock-phish' gang operate (as of 2007) in a rather different manner. While

some security engineers had recognized rock-phish attacks to be a dominant force in phishing as early as 2006 [106], no published description of their operation was available as we began investigating phishing attacks. Consequently, the following description of how the rock-phish gang operates (as well as the evidence of the group's status as a gang) is a notable contribution of this thesis.

Phishing attacks are inherently repetitive. This is especially true for rock-phish attacks. We developed our model of rock-phish behavior through trial and error, testing new attack instances to confirm expected behavior or refine our characterization.

After compromising a machine the rock-phish gang cause it to run a proxy system that relays requests to a back-end server. This server is loaded with a large number (up to 20 at a time) of fake bank websites, all of which are available from any of the rock-phish machines. The gang then purchase a number of domain names with short, generally meaningless, names such as `lof80.info`. The email spam contains a long URL such as: `http://www.volksbank.de.networld.id3614061.lof80.info/vr` where the first part of the URL is intended to make the website appear genuine and a mechanism such as 'wildcard DNS' can be used to resolve all such variants to a particular IP address. The IP address belongs to a machine acting as an HTTP proxy[3], relaying web traffic to and from a hidden 'mothership' machine.

Transmitting unique URLs trips up spam filters looking for repeated links, fools phishing-feed collators into recording duplicate entries, and misleads blacklist systems that search for exact matches. Since the numeric values in the URLs are sent to the DNS server (which the gang also hosts) it is clearly possible for the gang to track and customize responses. However, which bank site is reached depends solely upon the `url-path` (after the first `/`). Hence, a canonical URL such as `http://www.lof80.info/` is sufficient to fetch a top level web page, and its HTML fingerprint (an MD5 hash of the page's content) is sufficient to identify the domain and associated IP address as rock-phish.

Because the gang use proxies, the real servers – that hold all the web pages and collate the stolen information – can be located almost anywhere. The number and location of these servers might be found by inspecting the proxies and determining who they were communicating with, but we did not attempt to do this. However, we did observe some small variations between what the proxies returned both in the range of pages and the minutiae of their headers, making it clear that the gang was operating more than one server but failing to completely synchronize them.

The gang's methods have evolved over time. Originally they placed all their websites into a `/rock` directory (hence their name), but later this morphed into `/r1` and now the directory name has been dispensed with (although we found that `/r1/vr/` still works as a synonym for `/vr`). The gang's evolution has been tracked well enough, and their methods differ so much from other phishing websites, that it is useful to measure their activities separately.

[3]Throughout the thesis we sometimes refer to these proxy machines as simply rock-phish IP addresses.

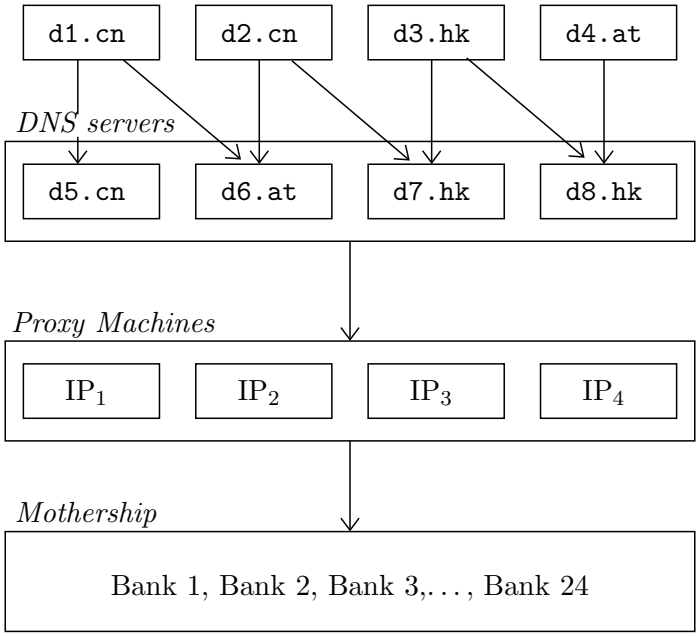


Figure 6.1: Cooperative architecture of rock-phish attacks.

In particular, their email spam, which in Spring 2007 has a characteristic section of random text followed by a GIF image containing the actual message, is estimated to account for between one third and one half of all phishing email. The rock-phish gang is believed to be extremely successful; it is claimed that they have stolen in excess of $100m by the end of 2006 [106].

To clean up an ordinary phishing site, it is sufficient to remove either the hosting website or the domain (if the domain is only used for phishing). However, the rock-phish gang shares resources as shown by the architecture sketched in Figure 6.1. It is designed so that the many domains and machines under the gang's control cooperate to maximize resilience. Whenever a domain name is registered, the domain owner must provide the name of a DNS server authoritative to resolve the domain name to an IP address. For example, the domain `tylerwmoore.com` points to the DNS servers `ns1.hostgo.com` and `ns2.hostgo.com` (servers run by a web hosting company), which return the web server's IP address `128.232.1.3`. Rather than point to legitimate DNS servers, each domain name used by the rock-phish gang points to one of several gang-controlled DNS servers. For example, in Figure 6.1, domain `d1.cn` uses `d5.cn` and `d6.at` as DNS servers. The gang's DNS servers are programmed to return the IP address of a proxy machine chosen from a pool of (recently) live proxies. Thus, whenever one of the proxy machines is removed, the website automatically switches to working machines that are still hosting a copy of the proxy code. Note that the rock-phish gang simultaneously exploits domains obtained from several registrars and machines hosted by many ISPs. This provides added protection from the occasional vigilant registrar or ISP. In Section 7.1 we provide evidence of coordinated replenishment of proxy machines and domains.

This switching behavior provides strong evidence that rock-phish websites collude, and are therefore truly a 'gang'. To verify this collusion, we selected a random rock-phish domain and examined each of the IP addresses associated with the domain. We tallied each domain that also used one of these IP addresses and recursively checked these domain's associated IP addresses. In this manner we identified every IP address associated with rock-phish domains starting from just one address.

The gang's architecture allows impersonation of many banks with each domain. Such overt cooperation creates additional risks, however. Notably, collusion should increase the domain's value as a take-down target. All banks whose websites are present on the rock-phish servers ought to be motivated to remove a site, not just one bank as for regular phishing websites. In Section 8.2 we study the extent of cooperation that takes place between organizations affected by rock-phish attacks.

6.1.3 'Fast-flux' phishing attacks

While collecting data in early 2007, the rock-phish gang introduced a new system dubbed 'fast-flux' by the anti-phishing community, with trials in February and wider deployment from March onwards.[4] The mechanism is similar to the one used by the rock-phish gang, except that the domain name is resolved to many IP addresses in parallel (typically 5 or 10 addresses are returned simultaneously) and the IP addresses used are rapidly changed (sometimes every 20 minutes).

Switching IP addresses so quickly of course 'eats up' hundreds of IP addresses per week, but the agility makes it almost entirely impractical to clean up the hosting machines. For these attacks the only workable strategy is to have the domain name suspended. The gang is likely to have large numbers of compromised machines available, since if they are not used to serve up phishing websites they are available for sending email spam. For further obfuscation, the gang changed from using the `url-path` to select the target bank to using the `Host:` header from the HTTP connection. This makes it somewhat more complex for ISPs and registrars to understand the nature of the websites and to what extent they can be considered to be 'live'. Interested readers can find more details about fast-flux in [75] (which gives many details about one of the networks that we also encountered).

[4] We were able to identify several machines that were used for both the original rock-phish scheme and for the new fast-flux architecture, so we are confident the same gang is involved. Further, although as of August 2007 there are three fairly distinct pools of fast-flux machines being used for phishing, there are a handful of overlaps which indicate to us that one gang is operating at least two of them.

6.2 Defending against phishing attacks

Banks mitigate the effects of phishing by continuously searching for fraudulent websites and initiating 'take-down' procedures, which remove the illegitimate pages from websites or suspend abusive domain names. Although a small number of banks deal with phishing website take-down exclusively 'in-house', the majority hire specialist companies to remove either all of the sites they care about, or sometimes just the 'hard' cases. The take-down companies that tackle phishing websites are usually one arm of more generic 'brand-protection' companies that deal with counterfeiting, and other intellectual property issues. We now describe three phases of phishing defense: detecting new websites, deciding if they are phish, and removing the sites.

6.2.1 Detection mechanisms

For timely removal of the website, it is essential that banks rapidly recognize URLs advertised in phishing emails. Some of these URLs are reported directly to the bank by customers who were not misled, and some are identified because the emails are undeliverable and delivery failure reports are received by the bank. URLs identified in these two ways are of direct interest only to the particular bank, but there are also other less specific sources of information.

Phishing emails are sent out indiscriminately, so many are detected by random members of the public who take the initiative to report them. Several websites exist for making reports, operated by organizations such as CastleCops [35] and the Anti-Phishing Working Group (APWG) [17]. In addition, many of the browser 'toolbars' that block access to phishing websites incorporate reporting mechanisms which deliver URLs to a central clearing house [110, 121]. Also, almost all of the URLs in circulation are detected by the anti-spam firms whose job is to keep unsolicited email from reaching their clients. These firms collate the URLs and submit them to groups such as the APWG.

Each take-down company collects multiple feeds of URL data. The company may also create its own feed by examining its own incoming spam, or by setting up special relationships with other companies. These proprietary feeds are used internally for the company's own take-down activities, but may also be supplied to client banks who wish to do their own take-downs, or simply be informed about the current levels of attack against their brands. They are also supplied to ISPs and domain name registrars who wish to proactively police their part of the Internet. In Chapter 8 we show that take-down companies are not sharing feeds, which significantly prolongs the lifetimes of phishing websites.

6.2.2 Decision mechanisms

The next step is to verify whether websites identified by the feeds are in fact phishing. Banks and take-down companies employ trained specialists to examine the websites picked up by the feeds. These employees can unilaterally decide whether each URL is phishing, so their evaluation is normally very fast.

However, there is an alternative approach to verifying potential phishing URLs. 'PhishTank' [134], one of the primary phishing-report collators, has explicitly adopted an open system powered by end-user participation. Users can contribute in two ways. First, they submit reports of suspected phishing websites. Second, they examine suspected websites and *vote* on whether they believe them to be phishing. PhishTank relies on the so-called 'wisdom of crowds' [152] to pick out incorrect reports (perhaps pointing to a legitimate bank) and confirm malicious websites. Each report is only confirmed (and subsequently disseminated to anti-phishing mechanisms) following the vote of a number of registered users. The tally of as-yet undecided votes is not revealed to users until after casting a vote. This helps prevent information cascades where early opinions influence later ones [8]. In Chapter 9 we compare the effectiveness of PhishTank's voting-based verification to a take-down company's unilateral approach.

6.2.3 Punishment mechanisms

Once a reported phish has been vetted, the URL is blacklisted to block further email spam and to assist anti-phishing browser toolbars in assessing the website's (in)validity. Meanwhile, the defenders send a take-down request to the operator of the free webspace, or in the case of a compromised machine or rock-phish proxy, to the relevant ISP who temporarily removes it from the Internet or otherwise ensures that the offending web pages are disabled. Where a domain name has been registered by a phishing attacker, as happens in rock-phish attacks, the defenders ask the domain name registrar to suspend the offending domain. However, not all ISPs and registrars are equally cooperative and knowing that a phishing website exists does not automatically cause its removal. Some ISPs take down phishing websites immediately, while others act very slowly. Responsiveness varies by company and by country, as well as with the competence (and language skills) of the organization requesting the removal. In Chapter 7 we study the effectiveness of the take-down strategy in removing phishing websites.

A longer-term punishment strategy is to go after the criminals behind phishing attacks. Unfortunately, this almost never happens in practice due to the same challenges faced in prosecuting other types of electronic crime. However, it is further exacerbated by the difficulty of connecting different phishing attack instances to the same criminal. This makes the financial losses too small for law enforcement agencies to justify pursuit.

6.3 Data collection and methodology

The average duration that phishing sites are accessible is an important measure of the state of phishing attack and defense. We next describe methodologies for quantifying phishing-site lifetimes and for determining the rate at which phishing victims are misled. We also describe how we gathered data on end-user verification of URLs in PhishTank.

6.3.1 Phishing-website availability

We have obtained a number of disparate feeds of phishing-website URLs. These include two volunteer organizations, PhishTank [134] which specializes in the URLs of phishing websites, and 'Artists Against 419' [20] which mainly deals with sites designed to facilitate auction scams or complex advanced fee fraud conspiracies. We take a feed from a brand owner, which consists almost exclusively of URLs for websites attacking their company. Finally, we receive feeds from two brand protection companies who offer specialist phishing website take-down services.

In all cases except PhishTank, the URLs are passed to us after they have been determined to be fraudulent websites. We deem the sites to have been 'up' at the time at which this determination was made or, if we are told when the site was initially reported, we use that (earlier) time instead. In practice, there is seldom much delay within the take-down companies, and they process URLs almost as fast as they are received.

In the case of PhishTank, we obtain the URL before the website has been validated as fraudulent – volunteers vote to determine this, sometimes triggering significant delays (see Chapter 9). For our purposes, we simply use the time of the first appearance on the PhishTank website as the earliest time that the site is known to have existed, and so our data is unaffected by how long voting decisions may take.

Unfortunately, the feeds do not provide an exact indication of when websites are removed. Additionally, some systems, notably PhishTank's, are regularly misled when phishing websites are not disabled, but replaced with generic advertising web pages. We therefore constructed our own testing system which, of necessity, became rather complex.

This system fetches reports of confirmed phishing websites from the feeds and records exactly when they first learned of the site. In order to track the existence of the website independently of whether its host name can be resolved, further records are constructed by replacing the host name part of the URL with the IP address it resolves to and the reverse DNS lookup of that IP address.

Often multiple URLs refer to the same website, either because they turn up in different feeds or because we have canonicalized two URLs into the same basic format: we remove specious parameters, fix insignificant case changes, and convert non-standard representations of IP addresses from hex, octal etc. into a standard dotted quad format. We treat

such multiple URLs as equivalent – and go further by removing the last component of the path as well. This solves two problems. First, it avoids double-counting equivalent URLs where one ends in a / and the other has an 'index.html' appended. Second, it overcomes the propensity of some feeds to include not only the initial webpage mentioned in the phishing email, but also the URLs of secondary pages that are encountered if credentials are filled in on the website. Wherever we have multiple URLs subsumed into one generic version, we determine the earliest of all the 'up' times and use that for our calculations.

We test all of the websites in our database on a continuous basis, around 15 to 20 times per day, to determine if they were still accessible. The web page data fetched (along with its HTTP headers) is fingerprinted so that significant changes (anything apart from date-stamps, session identifiers, etc.) can be detected. Just prior to fetching the page, the host name is once again resolved (having ensured that there was no cached data in the DNS server). If the host has moved to a new IP address, then further records for its IP address reverse DNS lookup are added to the database. Websites returning '404' errors are removed from the database, but timeouts and other temporary failures are retried for at least 48 hours.[5]

This testing regime enables us to determine precisely (with an accuracy of about one hour) when a phishing website is removed or changed, while remaining tolerant of temporary outages. Where multiple database entries pointed at the same web page, the fingerprinting enabled us to detect this and remove the duplicates. Also, for known malicious websites with identical fingerprints (and, in particular, the rock-phish attacks described in Section 6.1.2), we immediately categorized the sites as malicious, without waiting to discover whether the PhishTank volunteers had correctly done so.

We deem a phishing website to be 'up' so long as the content remains unchanged. Any significant change (beyond session identifiers, cookies, etc.) is treated as a removal. In practice, the phishing attackers create their fraudulent sites from 'kits' and do not change the pages once they are installed on a compromised machine. Indeed, they seldom change the content from one week to the next, as they host the pages on a series of different machines. The only manual step we apply is to ensure that the website has not been taken down and replaced with an innocuous page before we first monitor it. We measure the website lifetime from the earliest time we knew it was 'up' until the time of the last access to the fraudulent content.

Our data collection and analysis evolved significantly during the course of 2007. Initially, we obtained a single feed of suspicious URLs from PhishTank. In Chapter 7 we use data collected from PhishTank during a period of eight weeks between February and April 2007. During the summer, we obtained additional feeds from another volunteer organization, a

[5] At present, we are excluding all sites that involve non-standard forms of redirection to reach the final phishing webpage. This avoids considerable complexity (some phishers even use Macromedia flash files to redirect traffic), at the expense of completeness.

brand owner and two take-down companies. These additional feeds revealed substantially more phishing websites.

The data we discuss in Chapter 8 is taken from our record of the attacks that commenced in the time period Oct 1 to Dec 31 2007. To mitigate edge effects we monitored lifetimes for a further 25 days. Phishing attacks are continually evolving, but most of the attacks were relatively stable during this period – the exception being the fast-flux attacks which evolved from attacking one or two banks to attacking four or five at once. This makes them more like standard phishing websites at the beginning of the period and more like rock-phish at the end, which obscures many of the issues of cooperation we wished to examine. Hence, we completely exclude fast-flux attacks from the analysis within Chapter 8.

6.3.2 Visitor statistics

We also wished to gain a better understanding of the distribution of user responses to phishing attacks. We managed to gather some limited data from February to May 2007 on how many visitors a typical website received, as well as how many filled in the web form and provided any data.

In a small number of cases (approximately two dozen that we have detected) the website recorded details of victims into text files that were stored on the site itself in such a way that we could retrieve them. Inspection of these files showed how many responses were received and whether or not they were likely to be valid. Some of the entries were clearly testing (random sequences of characters), or consisted of profanities directed at the recipient of the data. The remainder of the responses were counted as valid, although it is understood that some banks deliberately provide data on dummy accounts for their own tracing purposes, so our counts may to some minor extent overestimate the number of people actually compromised.

In other cases we have collected publicly available web page usage statistics collated by the websites where the phishing pages reside. Webalizer [163] is a particularly popular package, which is often set up by default in a world-readable state on the type of web servers that seem to be regularly compromised. Indeed, it may be unpatched Webalizer vulnerabilities that permitted access in the first place. These statistical reports provide daily updates as to which URLs are visited, and these can be used to determine the total number of visitors and how many reached the 'thank you' page that is generally provided once personal data has been uploaded. By assuming that similar proportions of these 'hits' are valid occurrences of visitors compromising their identity information, it is possible to form a view as to the effectiveness of the phishing exercise and the distribution of visitors day by day. As new reports are obtained from PhishTank, we have automatically queried websites to determine whether Webalizer is running; if so, we returned daily to collect new reports. In all, we discovered over 2 500 websites using Webalizer in this manner.

6.3.3 User participation in PhishTank

The final set of data we analyze comes from PhishTank. Recall that PhishTank relies on web users to submit reports of suspected phishing websites and vote on whether they believe suspected websites are phishing. Consistent with its open policy, PhishTank publishes a record of all completed votes. This includes the identifiers of the user who submitted the report, the result of the vote (is or is-not a phish), the users who voted, and the percentage of votes cast for and against categorizing the website as a phish. However, the records do not specify how each user voted.

We examined reports from 200 908 phishing URLs submitted between February and September 2007. In all, 881 511 votes were cast during this period. 3 798 users participated by submitting reports and/or voting. In Chapter 9 we use this data on user participation to test the accuracy of PhishTank's decisions, its susceptibility to manipulation by attackers, and its completeness and timeliness compared to feeds from a take-down company.

Chapter 7

Cooperative phishing attack and defense

In this chapter we study the ongoing battle between attackers launching phishing websites and the banks removing them. We analyze empirical data on phishing-website-removal times and the number of visitors that the websites attract. We find evidence of a coordinated attacker for a major subset of phishing websites (operated by the rock-phish gang), whose architectural innovations have extended their average lifetime.

We also find evidence for a lack of cooperation between defenders. We show that several registrars are initially slow to remove rock-phish domains. Once they understand the nature of the attack, the gang moves on to a previously unaffected registrar, who repeats the slow response.

The website-removal times have a good fit to a lognormal distribution, which we argue reflects the inherently distributed responsibility for removing malicious content from the Internet. This highly skewed distribution of website lifetimes is the consequence of many uncoordinated defenders staving off many attackers. Within the general pattern, however, there is ample evidence that some service providers are faster than others at removing sites, and that some brands can get fraudulent sites removed more quickly. Finally, we provide a ballpark estimate of the total loss being suffered by the banking sector from the phishing websites we observed.

7.1 Rock-phish attacks and evidence of cooperation

We analyzed rock-phish websites during a period of eight weeks between February and April 2007. During this time, we collected 18 680 PhishTank reports which we categorized as rock-phish (52.6% of all PhishTank reports for the time period). While these reports are intended to be unique, we identified many duplicates due to the use of unique URLs

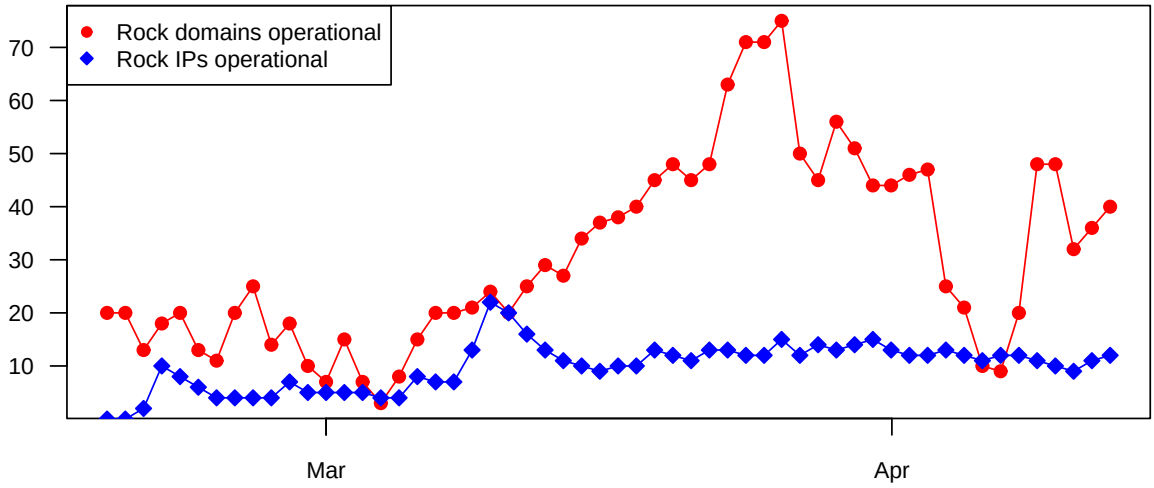


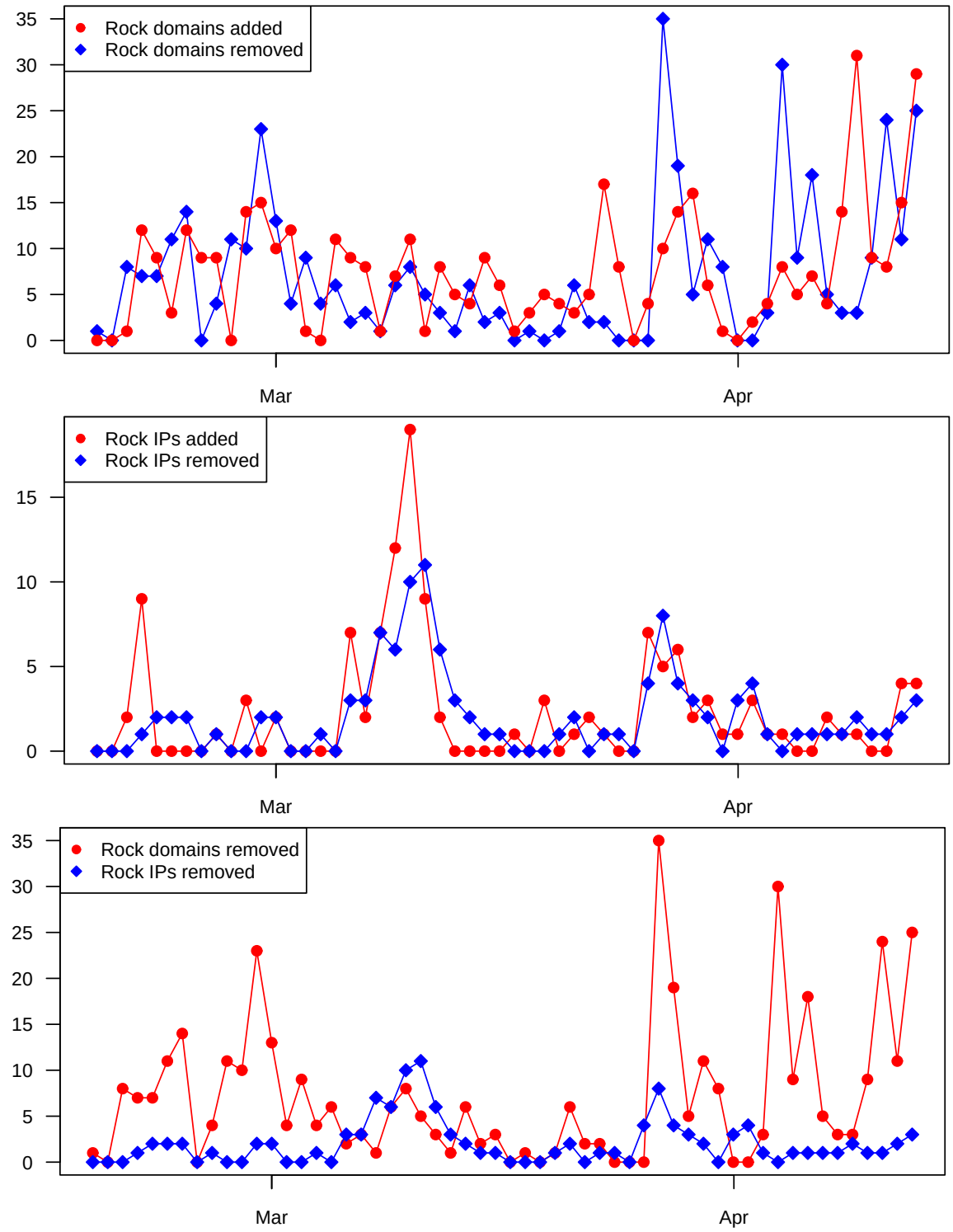
Figure 7.1: Rock-phish site activity per day.

as described in Section 6.1.2. This yielded a significant saving in effort, since just 421 canonical rock-phish domains were observed. Rock-phish sites used 125 IP addresses that were found to be operational for any duration. In all, the rock-phish sites impersonated 24 different banks.

Meanwhile, fast-flux sites triggered 1 803 PhishTank reports during the collection period. These reports pare down to 72 unique domains which resolve to 4 287 IP addresses. Observed fast-flux sites have targeted 28 banks.

Rock-phish sites continue to work for a particular domain that is mentioned in a spam email, provided that they can be resolved to at least one working IP address. Figure 7.1 tracks the average number of operational rock-phish domains and IP addresses on a daily basis. Proxies and domains were removed constantly, but they were replenished frequently enough to keep a number of sites working every day. Only once, right at the start of our data collection period, did the sites fail to work entirely, because the IP addresses being used for DNS resolution all failed. Otherwise, between 1 and 75 domains and between 2 and 22 IP addresses were always available.

Notably, the number of operational domains steadily increased during the month of March, before falling steadily in late March and early April. This is primarily due to a large number of `.hk` domains bought from a single registrar, which was slow to remove the offending domains. But why would the rock-phish gang continue to buy new domains when their earlier ones still worked? One reason is that the domains may lose effectiveness over time as they are blocked by spam filters. Indeed, comparing the number of domains added per day to the number removed (see Figure 7.2 (top)) reveals only a weak correlation between domain addition following removal. This suggests that the rock-phish gang is motivated to purchase new domains even when registrars are slow to take action.



	Correlation coefficient r	r^2
Rock domains added–Rock domains removed	0.340	0.116
Rock IPs added–Rock IPs removed	0.740	0.547
Rock IPs removed–Rock domains removed	0.142	0.0200

Figure 7.2: (Top) new and removed rock-phish domains per day; (Middle) new and removed rock-phish IPs per day; (Bottom) rock-phish domain and IP removal per day. Also included is a table of the respective correlation coefficients.

The story is rather different for the proxy machines to which rock-phish domains resolve. Figure 7.2 (middle) plots the day-by-day addition and removal of compromised machines used. Here the correlation is strong: as soon as machines are removed, new ones replace them. The correlation coefficient of 0.740 implies that 55% of the total variance is explained by the correlation between adding and removing machines. Perhaps the rock-phish gang uses automated IP replacement; automating domain acquisition, by contrast, is more difficult and costly. It is not surprising, then, that the data suggest manual selection prevails when adding domains.

Finally, we can infer whether coordination between rock-phish domain and machine removal takes place by comparing daily take-down rates for both (Figure 7.2 (bottom)). There is almost no correlation between the number of domains removed on a given day and the number of machines removed. This suggests that very little cooperation between registrars and ISPs takes place. Furthermore, the lack of correlation implies that either the banks and other removal entities are not communicating convincingly to both ISPs and registrars, or that they do not fully understand the rock-phish gang's use of domains and compromised machines.

7.2 Who is winning the arms race?

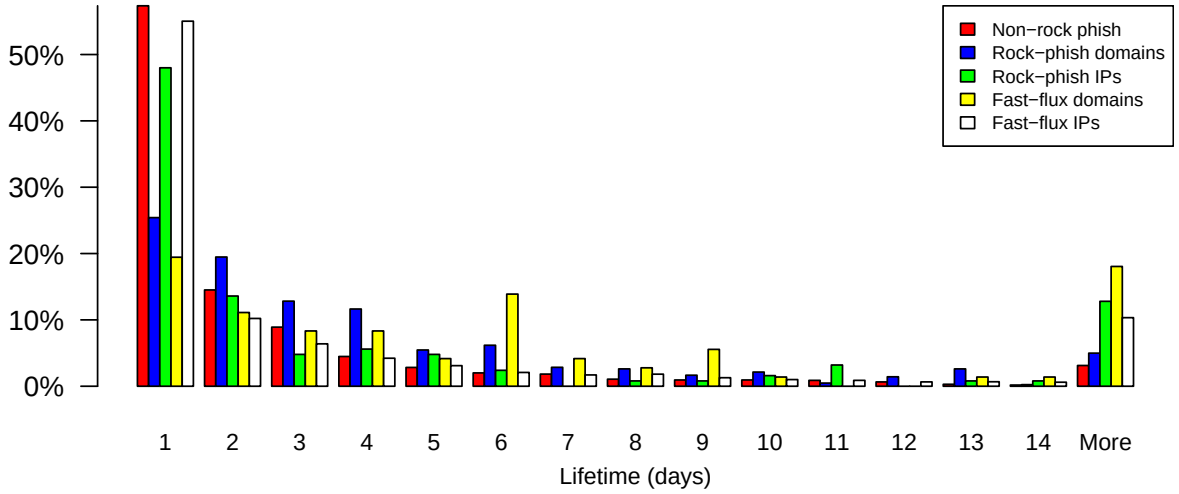
Banks invest significant resources in removing phishing sites. In this section we present data on the duration of phishing sites and on user response to these sites to determine the effectiveness of the take-down strategy.

In addition to the collection of rock-phish sites, we also examined reports of ordinary phishing sites. From 15 030 reports gathered over the same 8-week period from February to April 2007, we identified 1 695 unique non-rock-phish sites that were alive upon initial inspection.[1]

7.2.1 Phishing-site lifetimes

The site lifetimes for each type of phishing attack are given in the table in Figure 7.3. The mean lifetime of a normal phishing site is 61.7 hours, while for rock-phish domains the mean lifetime is 94.7 hours. Notably, for all phishing types, the median take-down time is much less than the average time. The reason why can be seen in the histogram of phishing-site lifetimes in Figure 7.3. Each bin represents one day, and the histogram covers two weeks, which is long enough for most websites (sites lasting longer are indicated by the 'More' column). 57% of non-rock-phish sites are removed within 24 hours of reporting,

[1]Many sites had already been removed by the time they have been verified and promulgated by PhishTank. Because we cannot evaluate whether dead-on-arrival sites are in fact a phishing site or simply a malformed URL, we exclude them from our lifetime analysis in this chapter.



	Sites	Mean lifetime (hours)	Median lifetime (hours)
Non-rock	1 695	61.7	19.5
Rock domains	421	94.7	55.1
Rock IPs	125	171.8	25.5
Fast-flux domains	57	196.2	111.0
Fast-flux IPs	4 287	138.6	18.0

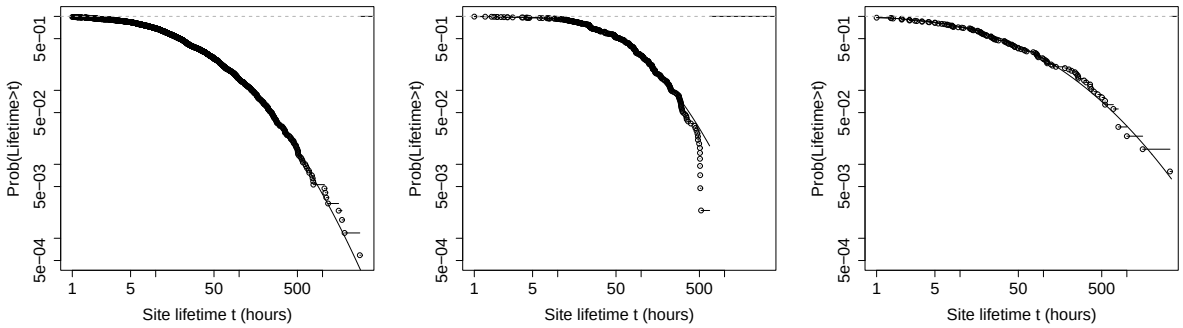
Figure 7.3: Histogram of phishing-site lifetimes with table of sample size and mean and median lifetimes.

while the remainder do not survive much longer. Only 28% of non-rock-phish sites last more than 2 days, though notably the tail carries on for several weeks. For instance, the longest-lived ordinary phishing site from our sample lasted over seventeen weeks! Chapter 8 identifies one of the reasons why some websites can remain for so long.

For rock-phish sites, the distribution is slightly different. While still skewed toward shorter times, the distribution has a heavier tail: a small but substantial number of rock-phish domains remain operational for longer periods. 25% are removed on the first day, 19% on the second, and 56% remain for 3 days or longer.

Rock-phish domains and proxies take significantly longer to be removed than ordinary phishing websites – 95 hours for domains and 172 hours for proxies on average. Recall that rock-phish spam always uses a domain name in the linked URL. This allows the gang to cycle through IP addresses as they fail. However, the data suggests that they do not have to switch all that often. While many are removed within one day, some remain for months before being removed.

One explanation for the longer lifetimes of rock-phish proxies and domains is that their attack method is not widely understood, leading to sluggish responses. Splitting up the components of the phishing attack (domains, compromised machines and hosting servers) obfuscates the phishing behavior so that each individual decision maker (the domain



	Lognormal				Kolmogorov-Smirnov	
	μ	Std err.	σ	Std err.	D	p-value
Non-rock	3.011	0.03562	1.467	0.02518	0.03348	0.3781
Rock domains	3.922	0.05966	1.224	0.04219	0.06289	0.4374
Rock IPs	3.434	0.1689	1.888	0.1194	0.09078	0.6750

Figure 7.4: Cumulative probability distributions with lognormal curve fit: non-rock-phish lifetimes with $\mu = 3.01$,$\sigma = 1.47$ fit (Left); rock-phish domain lifetimes with $\mu = 3.92$, $\sigma = 1.22$ fit (Center); rock-phish IP lifetimes with $\mu = 3.43$, $\sigma = 0.169$ fit (Right).

registrar, ISP system administrator) cannot recognize the nature of the attack as easily when an impersonated domain name is used (e.g., `barclaysbankk.com`), or HTML for a bank site is found in a hidden subdirectory on a hijacked machine.

Fast-flux sites exhibit markedly different behavior. Domains last much longer: over eight days on average, and there are far fewer than used for rock-phish. The fast-flux systems were also used to host a number of other dubious domains, mainly websites devoted to the recruitment of 'mules' who launder the proceeds from phishing. The 15 mule-recruitment domains we tracked lasted an average of 463 hours (median 135 hours), indicating that their removal was not a priority. Interestingly, the average lifetime of fast-flux IP addresses (138.6 hours) is a bit less than the lifetimes of IPs used for rock-phish attacks (171.8 hours). We speculate that the phishers are using machines at random and relying upon the domains resolving to multiple IP addresses to provide resilience, rather than actively selecting a handful of hosts that they believe are more likely to remain available.

The skewed distribution of site lifetimes shows that while most sites are removed promptly, a substantial number remain for a very long time. These long-lived sites cause the average lifetime to be much longer than the median lifetime. We have managed to fit some of the take-down data to match the lognormal probability distribution. To do so, we first estimated the parameters μ and σ which specify the distribution using maximum-likelihood estimation. To test the fit, we computed the Kolmogorov-Smirnov test [105] 1000 times to compute the average maximum difference D between the model and data.

The lognormal distribution turns out to be a good fit for the distribution of ordinary phishing sites as well as the rock-phish domains and IP address lifetimes. However, it

is not as good of a fit for fast-flux sites. (Fast-flux IP addresses are typically ignored by take-down procedures, while the lifetime of fast-flux domains is consistent with a lognormal distribution but there is too little data to confirm it.) The table in Figure 7.4 gives the relevant attributes for each fitted distribution, and the plot shows the lognormal cumulative probability distributions and the observed data points. Note that both axes are logarithmic in scale to demonstrate the goodness-of-fit in the tail of the distribution. It is significant that the take-down times for these three different categories of phishing attack can be modeled by the same family of long-tailed distribution, particularly since the actors responsible for the take-down are different (domain registrars, ISPs and system administrators).

Lognormal distributions arise whenever outcomes depend on realizing a number of independent, randomly-distributed preconditions. For example, software failure rates have been shown to follow a lognormal distribution [122]. This is because bugs occur in code locations surrounded by many conditional statements. Similarly, in order to successfully remove a phishing site, a number of conditions must be met: the site must be detected, the site may or may not be located at an ISP used to removing phishing sites, the bank may or may not have a working relationship with the police in the jurisdiction where the site is located, and so on.

7.2.2 User responses to phishing

Having established how long phishing sites remain operational, we now estimate user-response rates to phishing sites. We analyzed the site usage statistics from 144 phishing sites, from which we obtained daily snapshots of hit rates broken down according to URLs. From this list of popular URLs, we identified the phishing entry and completion pages and cross-referenced its PhishTank report to establish the earliest report date. Note that these were all ordinary phishing sites; the rock-phish websites do not leave logging data visible.

Webalizer also provides a rank ordering of entry pages. An entry page is the first page viewed by a site visitor. By tracking entry pages, we can readily distinguish between hits to the phishing page and the rest of the site. Each time we discovered a live site publishing Webalizer reports, we automatically returned daily to obtain updated reports until the site was taken offline. Thus, we ended up with a time sequence of reports used to estimate the distribution of victim responses for the days surrounding the phishing report.[2]

For most phishing scams, when someone enters their details on the site, they are taken to a fake confirmation page. We picked out these confirmation pages and noted the number

[2]Our system was not alone in visiting these websites to determine if they were still operational. We took steps to exclude these automated monitors from our datasets.

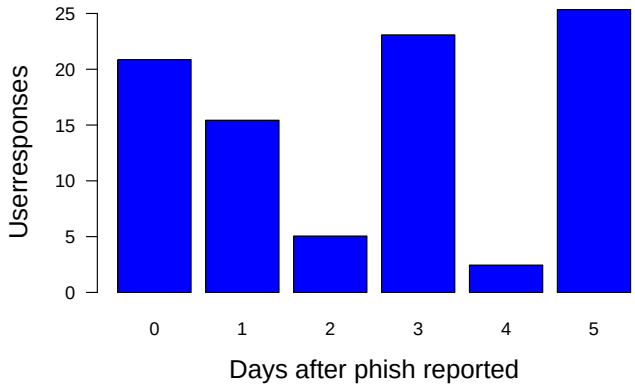


Figure 7.5: User responses to phishing sites over time. Data includes specious responses.

of hits they received, which was small compared with the number of hits on the initial page. Regrettably, Webalizer does not record the number of unique visits for all URLs, so we could seldom obtain the number of unique visits to the more popular entry pages. Instead, we estimated the number of unique visits to each site's confirmation page by taking the number of hits, and assuming the same fraction of visits to hits that we saw for the entry page.

Unfortunately, from the point of view of collecting good data, in many cases the site statistics presented difficulties: we could only obtain one reading before the site was removed, it could be unclear which were the confirmation pages, or the Webalizer values were not fetched until several days after the site went live. For these reasons, we obtained usable day-by-day statistics from just twenty sites. An average of these results is given in Figure 7.5.

We estimate that 21 unique users reach the confirmation page on the same day that the phish is reported. On the next day, another 15 responses are expected. A wide fluctuation in the average daily responses then occurs, from 5 responses on the second day after reporting to 25 responses on the third. This is doubtless due to the wide variation in the overall number of responses each site receives.

Somewhat surprisingly, for many sites the user responses sustain a fairly high level until the site is removed. We cannot say whether this is caused by ongoing spamming activity, or by users catching up with email backlogs in their inboxes. This ongoing activity was demonstrated to an extreme by the usage statistics for a PayPal phishing site loaded onto a web page for the Niger Water Basin Authority. This site remained alive into March 2007 and received a steady stream of phishing responses over a month and a half, so the failure to take it down more quickly caused ongoing problems. Thus it does appear that take-down, even when slow, always has some positive effects.

We also observed noticeable variation in the number of responses received. One site (excluded from the average presented in Figure 7.5 because of missing data) drew over

500 responses in one day. Hence a small number of sites may draw significantly larger numbers, so the data presented here should be viewed as a conservative estimate.

But how accurate is the confirmation rate as a measure of successful attack? Just because the confirmation page is visited, this does not necessarily mean that every hit corresponds to a theft of personal details. To arrive at a more accurate success rate, we have also gathered 414 user responses with personal information published on phishing sites in what the attacker believed to be an obscure location. We examined each response by hand to determine whether the responses appeared plausible. Many responses were obviously fake, with names and addresses like 'Die Spammer' and '123 Do you think I am Stupid Street'. In fact, the responses were evenly split: 214 responses were obviously fake, while 200 appeared real. Hence, albeit from a small sample, we can estimate that half the responses to a phishing site represent actual theft of details.

So how does this user-response data relate to the phishing-site lifetimes we described in Section 7.2.1? Of the sites we sampled, we might expect around 17 victims per site if they are removed within one day of reporting, and rising by 8.5 victims for each successive day. This is a substantial number, and it is unclear whether the banks can act sufficiently quickly to reduce it by very much.

7.2.3 Estimating the cost of phishing attacks

We can now use our empirical data to estimate the cost imposed by phishing attacks. We must of course qualify our calculations by noting that we are using a number of rather fuzzy estimates, so that substantial refinement will be possible in the future as better figures come to light.

We first consider the cost imposed by ordinary (i.e., not rock-phish or fast-flux) phishing sites. We collected data for eight weeks and confirmed 1 438 banking phishing sites (we exclude eBay phishing scams for the purpose of this calculation). Extrapolating, we might expect 9 347 sites per year. These particular sites remain operational for around 61 hours on average, which yields approximately 30 victims based on the analysis in Section 7.2.2. Gartner has estimated the cost of identity theft to be \$572 per victim [64].[3] Hence, the estimated annual loss due to ordinary phishing sites is $9\,347 * 30 = 280\,410$ victims $*$ \$572 = \$160.4m. Gartner estimates that 3.5 million Americans give away their details annually, which leads to an estimated loss of \$2bn.

We cannot reliably provide an estimate for the costs of rock-phish and fast-flux phishing scams since we do not have similar response data. However, given that the rock-phish gang sends a large proportion of all spam [106], which drives visitor numbers, it is fair to assume that they steal at least as much money as ordinary phishers. Thus, we estimate,

[3]Gartner also gives a value of \$1 244 per victim, but reports that over half of this is subsequently recovered.

at an absolute minimum, that at least $320m is lost annually due to phishing scams. The disparity with Gartner's total of $2bn is doubtless due to the extremely rough approximations used, both by ourselves and Gartner. Incompleteness in PhishTank's feed may also account for the difference, as explained below.

Concurrent to our own investigations, other researchers have used a completely different technique to estimate the number of phishing victims worldwide and managed to arrive at a remarkably similar figure to ours. Florêncio and Herley got half a million volunteers to run special software that identified when they likely fell victim to a phishing attack [58]. They constructed a whitelist of known websites where users enter passwords. A central server was notified anytime a user entered the same password into a whitelisted website and an unknown site. If several users of a whitelisted website entered their passwords for the same unknown website, then the responses were marked as phishing.

The vast majority of Internet users do not fall victim to phishing attacks despite their prevalence. Consequently, the authors' technique required a very large number of participating users in order to pick up on several users visiting the same phishing website. During a three-week period when approximately 436 000 users were actively using their software, the central server recorded 101 instances of password reuse. Using this data, the authors estimated the proportion of the Internet-using population annually being phished as follows:

$$\frac{101 \text{ victims} \times 365 \text{ days per year}}{436\,000 \text{ users} \times 21 \text{ day sample}} = 0.4\%$$

Given the authors' estimate of 500 million Internet users, this translates to around 2 million phishing victims per year. Note that this technique does not distinguish between types of phishing attacks, so the estimate encompasses both ordinary and rock-phish attacks.

2 million victims is about one order of magnitude greater than the 280 410 victims we calculated. However, our estimate significantly underestimates the number of phishing websites present because it only includes sites reported by PhishTank. We have subsequently added more feeds from additional sources, and we show in Chapter 8 that each new feed identifies many websites missed by the others.

Unfortunately, the data in Chapters 7 and 8 are not directly comparable. Chapter 8 examines only those phishing pages targeting the clients of two major take-down companies during October to December 2007. Even so, we can obtain a useful measure of the proportion of phishing websites missed by PhishTank. Approximately 34.9% of the phishing URLs examined in Chapter 8 are identified by PhishTank, with the rest coming from the two take-down companies. Therefore, a more accurate measure of phishing victims is $\frac{280\,410}{0.349} = 803\,000$ victims per year. This revised estimate gets us to nearly half the number estimated by Florêncio and Herley. Since our estimate does not consider rock-phish attacks (which sends out large volumes of rock-phish spam), the estimates are in fact quite similar.

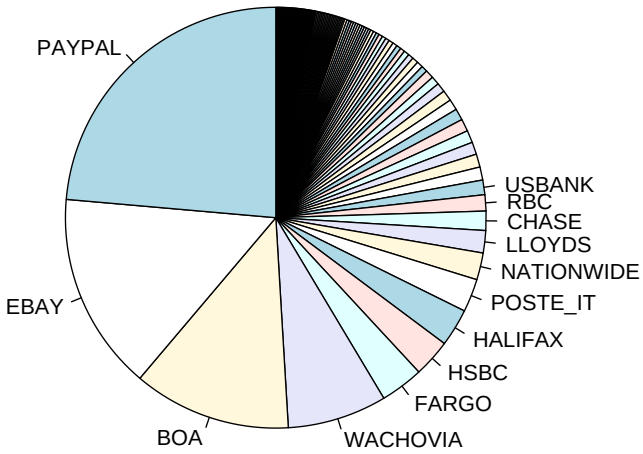


Figure 7.6: Proportion of ordinary phishing sites impersonating each bank.

7.3 Factors causing variation in take-down speed

Many factors can affect the lifetime of a phishing website. We now test some of the more plausible explanations, from the bank being targeted to the day the phishing website is launched. We explore an important additional factor, awareness of phishing websites' existence, in the next chapter.

7.3.1 Comparing bank performance

122 banks were targeted according to our sample of ordinary phishing sites. However, some banks were targeted a lot more than others: PayPal was impersonated by 399 of the 1 695 sites, while 52 banks were only spoofed once. A pie chart showing the proportion of targeted banks is given in Figure 7.6.

While banks cannot control the number of fake sites that appear, they can certainly help determine how long they stick around. Here there is also significant disparity. Figure 7.7 presents a rank-ordering of the average site lifetimes for banks impersonated more than five times during the sample period. Egg, TCF Bank, eGold and Bank of America are slowest at taking down sites (over 4 days), while Capital One, NatWest and Flagstar Bank are quickest (around 12 hours). Note that the results should be treated with caution because the differences, at least in part, result from different choices by the attackers as to where sites are hosted. Furthermore, a few long-lived sites can drastically alter the average lifetimes when banks are rarely impersonated.

7.3.2 Comparing free-hosting performance

We identified a number of providers of 'free' webspace that regularly host phishing websites. We tracked five organizations' take-down performance for phishing sites launched

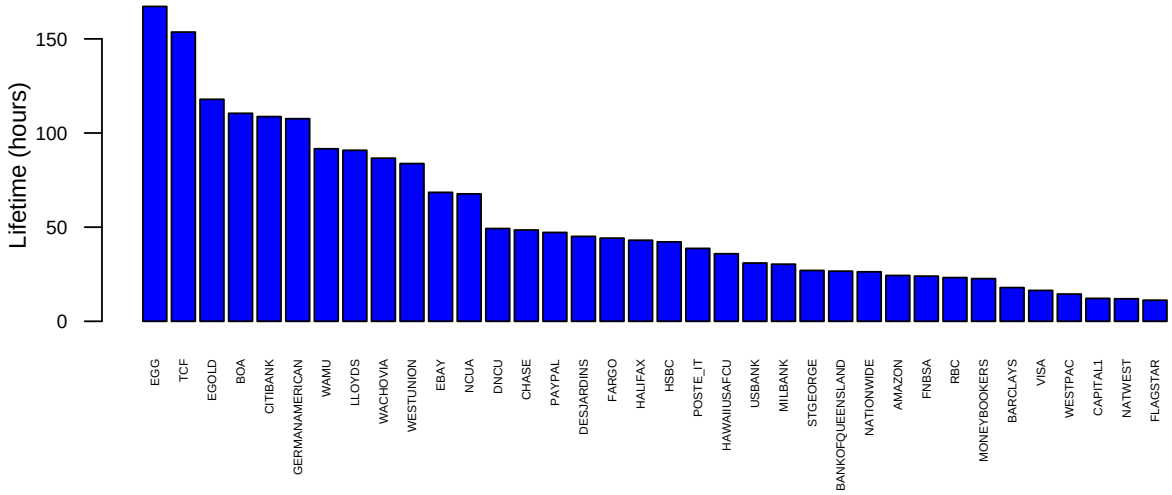


Figure 7.7: Phishing-site lifetimes per bank (only banks with five or more sites are presented).

between February 17 and June 30, 2007 (a longer period than the other datasets we report upon in this chapter). The results are given in the following table:

	Sites	Mean lifetime (hours)	Median lifetime (hours)
`yahoo.com`	174	23.8	6.9
`doramail`	155	32.8	18.1
`pochta.ru`	1 253	33.8	16.8
`alice.it`	159	52.4	18.8
`by.ru`	254	53.1	38.2

As is apparent, the take-down times differ between the organizations, with Yahoo! being the fastest. Yahoo's already impressive take-down performance is understated, since approximately half of the sites had already been removed before appearing in PhishTank and are consequently ignored by our calculations.

However, it is a little more complex than the table indicates. The vast majority of phishing sites hosted on `doramail`, `pochta.ru`, `alice.it` and `by.ru` impersonated eBay, along with a few PayPal and Posteitaliane fakes. By contrast, the sites on Yahoo's free 'GeoCities' webspace impersonated a wide range of different institutions, so it is not possible to determine cause and effect with complete confidence. There may be some delays not only at the hosting provider but also within eBay (and these sites accounted for over a third of all eBay sites). However, it is noteworthy that in all cases the average lifetime of free-hosting sites is shorter than for regular phishing sites. This is likely to be due to differences in service obligations: 'free' webspace can be pulled at the first sign of foul play while regular ISPs must be sensitive to inconveniencing a paying customer with a potential disruption.

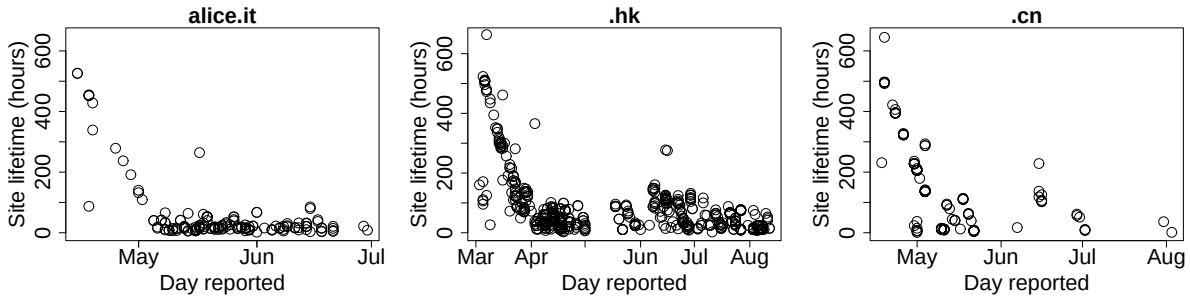


Figure 7.8: Scatter plot of phishing-site lifetimes over time.

7.3.3 The 'clued-up' effect on take-down speed

We also investigated whether the take-down performance of the providers and registrars changed over time. Figure 7.8 presents scatter plots of phishing-site lifetimes based on the date reported. Phishing sites started appearing on the free host `alice.it` in April 2007. Yet nothing was done to remove any of these sites until early May. This effect can be seen in its scatter plot (Figure 7.8 (left)), where the April site lifetimes decrease linearly. Once the host became 'clued up' to the existence of phishing sites, it started removing sites much more promptly. However, we did not observe a similar effect for the other free-hosting firms. Most likely, they had already been targeted prior to our data collection period, so we could not witness a similar effect.

We did, however, see the same effect for domain name registrars removing rock-phish domains. Both `.hk` (Hong Kong) domains (Figure 7.8-center) and `.cn` (China) domains (Figure 7.8 (right)) lasted much longer in their first month of use when compared to later months. These plots support the often-espoused notion that attackers benefit by continuously seeking out new targets, and suggest that some of the relative success of the rock-phish gang may come from their rate of innovation rather than innate technical ability. Such a conclusion is consistent with Ohm's warnings against the myth of the 'Superuser' [127] – people ascribe extraordinary capabilities to hackers when the reality is that successful attackers exploit basic failings which are missed due to a lack of effective measurement.

7.3.4 Do weekends affect take-down speed?

Defenders working for targets of phishing attacks often speculate that attackers deliberately wait to advertise phishing sites until just before the weekend to maximize uptime, since many system administrators will be away. Upon examining the data, we find that sites launched before the weekend are no more likely to last longer.

We first examine whether sites reported near the weekend stay around longer than those reported earlier in the week. The left graph in Figure 7.9 shows the average duration of

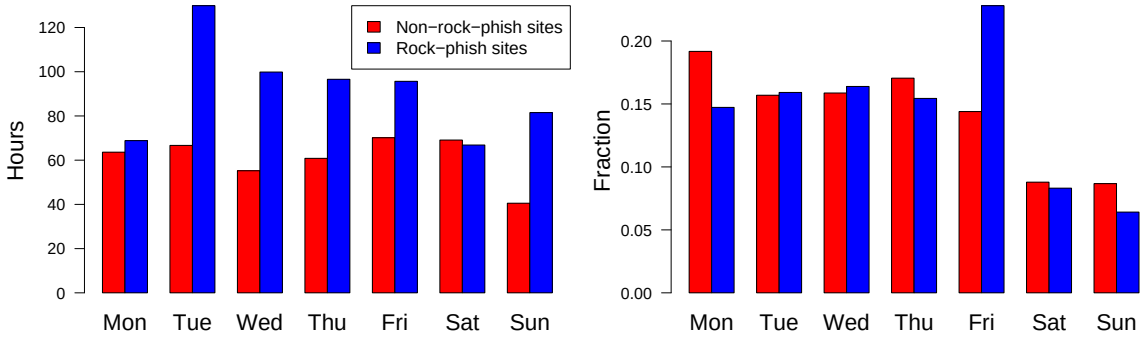


Figure 7.9: Phishing-site lifetimes (Left) and report totals (Right), collated by the weekday first reported.

phishing sites based upon the day of the week the site was first reported. Rock-phish sites reported on Tuesday last longest, while those reported on Monday and Saturday are removed quickest. It is unclear whether there is any significance to these differences. Non-rock-phish sites launched on Saturday last around one day longer than those reported on Sunday, so it seems as if reports from both Saturday and Sunday are actioned at much the same time.

The next question we address is whether some days are more popular for launching phishing sites than others. The right graph in Figure 7.9 measures the fraction of sites reported on each day of the week. The most striking conclusion to be drawn from this graph is that the weekend is the least popular time for both rock-phish and ordinary phishermen to set up sites. More accurately, fewer *reports* of new phishing sites are created over the weekend. It is impossible to tell whether there are fewer sites appearing, or fewer people looking for them, on Saturday and Sunday.

7.4 Discussion

7.4.1 DNS trade-offs

When phishing first became widespread attackers often used domain names which were minor variations on the real site's identity. This is now rather less common. One of the reasons for this is that it gives the defenders the option of either getting the site removed or having the domain name suspended. The latter approach is simpler since it requires cooperation by relatively 'clued-up' registrars who are already experienced in dealing with the branding implications of too-similar domain names, rather than seeking help from ISPs who might not be familiar with phishing attacks.

The rock-phish gang use nondescript domain names to avoid this issue of branding, leaving the registrar with the dilemma of whether to break a contract on the word of a third party

who claims that the domain name is being used for phishing. That registrars are now prepared to suspend the names is apparent from our data – though it is interesting to note that at present no systematic attempt is being made to suspend the names that are being used for the DNS servers associated with the rock-phish domains. This is despite these names being created solely for the purpose of providing an indirection for the DNS servers used to resolve the rock-phish URLs. The argument that these too are entirely fraudulent is not yet won – though as can be seen from Figure 7.1, when the rock-phish DNS system is disrupted the effect can be dramatic. Of course, once these name service domains are regularly suspended the gang could switch to absolute IP addresses for locating their DNS servers, thereby continuing operations with less flexibility.

The final trade-off of note that relates to DNS is the caching mentioned in Section 6.1.2. Setting a high value for 'time-to-live' ensures that domain names may be resolved, particularly at larger ISPs, for some time after the domain is suspended by a registrar. However, lower values offer more agility as compromised machines are reclaimed by their owners.

7.4.2 Countermeasures

If take-down strategies do not entirely mitigate phishing attacks, what else can be done?

One important advance would be to reduce the information asymmetry for the defenders. Phishers obfuscate their behavior so that sites appear independent, which causes many to view phishing as an intractable problem. It is in the interest of security vendors to accept inflated statistics to make the problem seem more important. Indeed, such inflation has occurred frequently, from PhishTank boasting about the large number of sites it identifies [128] to APACS, the UK payment association, asserting a 726% increase in phishing attacks between 2005 and 2006 (with merely a 44% rise in losses) [18]. If there appears to be thousands of small-scale phishing attackers, then law enforcement is unlikely to investigate. Whereas, their interest would peak if there were only a handful of criminals to catch. Hence, improving the measurement systems, and better identifying patterns of similar behavior, can help defenders focus their response upon a smaller number of unique phishing gangs.

Other entirely obvious countermeasures include reducing the availability of compromised machines, rate-limiting domain registration, dissuading users from visiting the sites, and reducing the damage that disclosing private information can do. Unfortunately, these strategies are either infeasible or are being attempted with limited impact so far. What does work, to some extent, is when banks being attacked improve their back-office controls. The incentives to go phishing are much reduced if miscreants cannot use the account numbers and passwords they steal to transfer money out of accounts. Similarly, the situation could be improved if phishers were stopped from moving money out of the banking system in ways that the transfers could not be clawed back [9].

7.5 Conclusion

In this chapter we have empirically measured phishing-site lifetimes and user response rates to better understand the impact take-down strategies have. While take-down certainly hastens the fraudsters' movement from one compromised website to another, many users continue to fall victim. Furthermore, the data reveals that sophisticated attackers can extend site lifetimes. Indeed, the rock-phish gang has already demonstrated techniques for adapting to regular removal. They have invented (or stumbled upon) a relatively successful strategy, and with 'fast-flux' are experimenting with another, but it is far from clear that all defenders currently understand those mechanism, or how best to disrupt them.

Removing phishing websites is often perceived as a Sisyphean task, but our analysis shows that even when done slowly, it does reduce the damage caused. We have also demonstrated wide disparities in reaction time between comparable organizations. We have shown that these disparities extend across borders, that some banks work faster than others and that some web-hosting companies do a better job at removing sites. Improving the transparency of attacker strategy and defender performance is key to reducing the success of phishing scams.

Chapter 8

Non-cooperation when countering phishing

Many security problems are dealt with by several distinct groups of people, who take individual action against a common threat. It is often the case that cooperation could improve their overall effectiveness, but at the same time, some of the groups may see cooperation as unfair because they would contribute more than others; or they may fear that their ability to sell specialist services is diminished; or the effect may be that the incentives for them to invest in improved techniques is much reduced. The balance between the positives and the negatives, and hence who has an incentive to change tactics, helps determine whether cooperation occurs naturally, or whether it happens only if imposed by an outside force.

In this chapter we examine the phishing website take-down industry, where there is some measure of cooperation already. However, several key players are not cooperating to the full extent. We experimentally measure the impact of a lack of cooperation in sharing information about attacks, and estimate the dollar cost this incurs.

As described in Section 6.3.1, during the course of our research into phishing activities, we have obtained proprietary feeds from a number of take-down companies. We have combined this data with some other feeds, from a brand owner, from the Anti-Phishing Working Group (APWG) and from some public domain sources. The average website lifetimes that we measured in Chapter 7 were somewhat higher than the take-down companies had expected. Initially we attributed the disparity to selective memory – the companies remembered the run-of-the-mill work they did, without realizing quite how many sites hung around for many days. However, upon further consideration, we realized that some of the effect might be ascribed to the companies simply being unaware of the websites whereas we, with our multiple feeds, simply knew about more sites than any one company ever did. It is this conjecture, that take-down times are unnecessarily long because of lack of information sharing, that this chapter examines – and shows to be correct.

The data we analyze in this chapter is taken from our record of the attacks that commenced in the time period from October 1 to December 31, 2007. It incorporates feeds from two take-down companies and PhishTank. In Section 8.1 we compare the feeds of two take-down companies and unequivocally show that take-down times for phishing websites would be shorter if the two companies were to share the information in their feeds with each other. In Section 8.2 we consider websites that are operated by the rock-phish gang. Taking down rock-phish websites is inherently cooperative because the process is already a 'sum of efforts'. Nevertheless, we show that once again there is a gain to be made from sharing feeds. In Section 8.3 we consider the incentives which prevent information sharing by take-down companies, while drawing parallels with other realms of computer security within which sharing is, or is not, the norm. We then make a clear recommendation for change within the anti-phishing community. In Section 8.4 we survey related work on the pros and cons of information sharing and finally in Section 8.5 we draw overall conclusions.

8.1 Consequences of non-cooperation

We start by considering the removal of ordinary phishing websites. We will consider rock-phish attacks in the next section; in this section we consider the take-down of sites attacking a single bank.

8.1.1 Motivating example

We begin by examining the phishing websites of one of take-down company A's clients, a bank called $A1$, which has hired A to remove phishing websites on their behalf.[1] While many of the websites impersonating $A1$ are identified by A, A is not always aware of every site impersonating $A1$. Figure 8.1 presents Venn diagrams [159] showing the sites impersonating $A1$. The left circle represents A's view of the sites impersonating $A1$, while the right circle represents the view from our own aggregated feed of phishing websites.

In A's view, 760 websites impersonated $A1$ during the sample period. In fact, the information from other sources reveals that at least 1 424 websites did so. What is the effect of A's incomplete view? We can approximate the adverse effect by examining the lifetimes of the phishing sites for each category.

The sites A knows about exclusively are removed within 15 hours on average. Phishing-website lifetimes are highly skewed (as demonstrated in Chapter 7), so a few long-lived websites can greatly impact the average lifetime. Hence, the median is a more robust

[1] Unfortunately, we cannot disclose the names of the take-down companies or the identity of any of their clients.

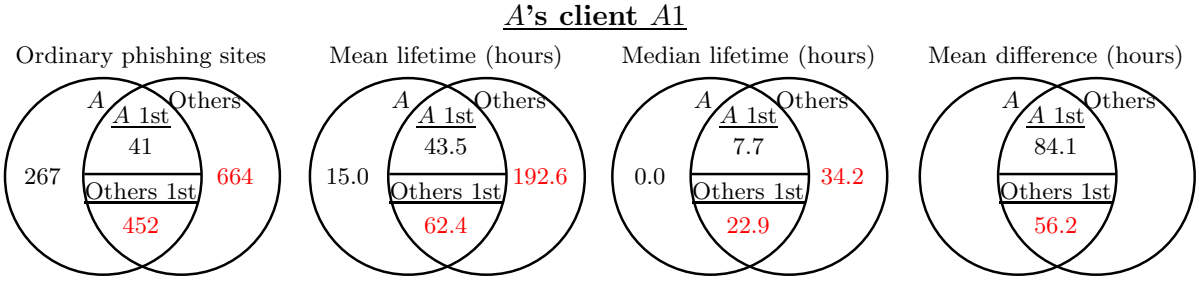


Figure 8.1: Comparing take-down company A's awareness of phishing websites impersonating its client $A1$ to an amalgamated view constructed from all of our sources of data.

measure of typical behavior. The median lifetime for sites known exclusively to A is 0 hours – in other words, half the sites are already dead by the time we have started to monitor the URLs in the feed delivered to us from A. By contrast, sites unknown to A have a much longer lifetime: 193 hours on average (34 hour median), over a week longer than sites known exclusively to A.

It is reasonable to ask why phishing websites unknown to A are removed at all. There are several plausible explanations. First, $A1$ may also attempt to remove phishing sites directly without involving the take-down company. Second, $A1$ may have hired more than one take-down company – we have entirely excluded from our analysis clients we were told were employing multiple take-down companies, but the companies may have been unaware of all of their clients' arrangements. Another explanation is that the owner of the machine where the website was hosted may have become aware of the site themselves, either through examining logs, or because of other abusive activity, from sending out spam to hosting IRC 'bots'; or they may have received reports of other phishing websites hosted on the same machine. Finally, many motivated third parties (such as the Castle Cops [35] team) work on a voluntary basis to persuade ISPs to remove phishing websites.

In addition to the websites impersonating $A1$ that A was unaware of, many websites were identified by A some time after other sources knew of their existence. Of the 760 websites A did know about, 452 were identified by the other sources first, at an average of 56 hours (19 hrs median) before A learned of them. The impact of such delays can be seen in the lifetime figures: these websites remain for 62 hours on average, 47 hours longer than sites known only to A.

Clearly, the bank $A1$ stands to gain through reduced phishing-site lifetimes if A improves its knowledge by obtaining feeds from other take-down companies. A would also benefit, not only from the increased revenue opportunities of having more work to do, but also from being able to market a better overall service. However, we have only examined one client bank so far, so we now conduct a more comprehensive analysis, examining all of A's clients, along with the clients of a second take-down company, B.

8.1.2 Comparing coverage of two take-down companies

Having just demonstrated that an incomplete view of phishing websites has harmed one bank and one take-down company, we now show that the effect applies more broadly. We study the phishing feeds provided to us by two take-down companies, called A and B. We cannot simply compare the complete coverage of each feed to ascertain whether their feeds are comprehensive. This is because disparities between the feeds inevitably result from take-down companies being more concerned about obtaining comprehensive lists of websites that impersonate their client banks than in overall completeness. Instead, we examine the extent to which their feeds omit websites impersonating their respective clients.

Table 8.1 breaks down website lifetimes for the six most frequently attacked client banks for each take-down company. We refer to company A's clients as $A1$ (this is the same $A1$ as in Section 8.1.1 above), $A2$, $A3$, $A4$, $A5$ and $A6$, and similarly we refer to company B's most frequently attacked clients as $B1$, $B2$, $B3$, $B4$, $B5$ and $B6$.

We study A's performance first. Notably, the website lifetimes of every client $A2$–$A6$ is consistent with the intuition established for $A1$ in the previous section: sites missing from A's feed take much longer to be removed, while sites appearing in A's feed only after being discovered elsewhere are also longer-lived. We can also see that A's feed is somewhat incomplete when compared with our amalgamated view. Combining all of A's 53 clients that were attacked during the sample period, 2 219 phishing websites, or 31% of the total, were completely missed by A. Another 2 225 websites, or 31%, were identified by other sources before A identified them. Hence, A's clients stand to gain substantially if the company can arrange to obtain more feeds.

In particular, there exists a large gap of over three and a half days between the lifetimes of phishing websites known exclusively to A and those known only to others. This stark difference can help us understand why there are conflicting views of take-down performance, and it offers an explanation why take-down companies often boast of the speed with which they remove targets when the actual times measured by outside observers can be worse.

Having established that A stands to gain from greater sharing of phishing-website feeds, we now study whether B also needs to obtain more information. Studying additional companies is important because it helps determine whether the case for reciprocation can be made. In other words, do A and B both stand to gain from sharing their feeds with each other?

Alas, the case for B is less clear-cut than for A. B's feed appears to be substantially more comprehensive than A's. The benefit B might obtain from outside sources is smaller. For clients $B2$ and $B6$, for example, there is no unique contribution from other sources, just a handful of sites discovered by others before B.

	A only			Others only			*A* first			Others first		
		Lifetime			Lifetime			Lifetime			Lifetime	
	#	mean	med.	#	mean	med.	#	mean	med.	#	mean	med.
A's top 6 clients:												
*A*1	267	15.0	0.0	664	192.6	34.2	41	43.5	7.7	452	62.4	22.9
*A*2	552	14.1	0.0	527	49.2	15.9	120	35.1	19.0	463	46.7	24.9
*A*3	537	10.5	0.0	200	74.3	20.5	95	27.5	12.3	545	50.3	20.4
*A*4	183	8.0	0.0	310	114.4	0.0	25	339.5	133.8	290	16.8	0.0
*A*5	203	10.6	0.0	121	37.5	0.0	55	5.8	0.0	169	15.6	0.0
*A*6	43	18.4	0.0	213	68.2	24.1	21	79.2	41.8	148	102.1	25.0
A's 53 clients combined:												
A∗	2 267	13.1	0.0	2 219	112.2	20.1	395	38.8	13.7	2 225	51.1	18.2
	B only			Others only			*B* first			Others first		
		Lifetime			Lifetime			Lifetime			Lifetime	
	#	mean	med.	#	mean	med.	#	mean	med.	#	mean	med.
B's top 6 clients:												
*B*1	295	21.6	6.1	52	72.3	0.0	158	40.4	21.5	20	43.7	26.4
*B*2	165	3.1	0.0	0	0.0	0.0	26	16.0	9.5	1	0.0	0.0
*B*3	97	46.1	0.0	25	23.8	0.0	41	82.1	28.8	15	48.1	45.4
*B*4	93	0.0	0.0	5	0.0	0.0	34	0.0	0.0	5	0.0	0.0
*B*5	77	29.3	5.1	1	7.8	7.8	22	15.0	6.2	1	18.5	18.5
*B*6	60	12.0	4.4	0	0.0	0.0	19	21.0	15.6	2	49.4	49.4
B's 66 clients combined:												
B∗	1 540	16.4	0.0	115	42.9	0.0	443	26.3	9.7	80	31.4	17.3

Table 8.1: Phishing-website lifetimes (in hours) for the clients of take-down companies A and B, broken down according to whether the websites are identified by their respective take-down provider or by outside sources.

It is the case that B's most attacked client $B1$ has figures that are consistent with the results for A: the 52 websites discovered exclusively by others remain for two days longer than those identified only by B, and the 20 sites picked up by others before B remain for about one day longer. However, the lifetimes for the remainder of B's top six clients do not nicely match up to the expected outcomes as A's top clients did. This could be due to the small sample size of the other contributions, the scarcity of which would reflect well on B, or it could merely arise because B receives much the same feeds as we do. Nonetheless, the overall contribution from others, while small, is not trivial. Of the 2 178 websites impersonating B's clients, 115, or 5%, were identified by others and missed by B. An additional 80 websites, or 4%, were picked up by others first. When we combine

	A first			Others first		
		Difference			Difference	
	#	mean	med.	#	mean	med.
A's top 6 clients:						
*A*1	41	84.1	0.0	452	56.2	18.7
*A*2	120	30.8	16.4	463	44.1	18.4
*A*3	95	77.2	21.0	545	25.0	11.5
*A*4	25	0.0	0.0	290	18.4	6.4
*A*5	55	16.4	1.5	169	31.0	9.7
*A*6	21	55.4	20.8	148	32.4	15.4
A's 53 clients combined:						
A∗	395	41.3	14.3	2 225	40.9	15.5
	B first			Others first		
		Difference			Difference	
	#	mean	med.	#	mean	med.
B's top 6 clients:						
*B*1	158	16.8	2.9	20	28.7	8.7
*B*2	26	21.6	21.6	1	0.0	0.0
*B*3	41	5.6	5.6	15	7.5	3.2
*B*4	34	0.0	0.0	5	0.9	0.9
*B*5	22	2.6	2.6	1	43.4	43.4
*B*6	19	9.2	5.3	2	0.0	0.0
B's 66 clients combined:						
B∗	443	15.7	6.3	80	14.3	2.4

Table 8.2: Time differences (in hours) in reporting phishing websites when both A and B identified the website.

the results from all 66 of B's clients attacked during the sample period, the outcome once again becomes consistent with that found for A's clients. We conclude that although the effect is smaller than for A, almost all of B's clients still stand to gain something from a shared data feed.

Table 8.2 lists the average time lag in reporting between phishing websites that are detected by the relevant take-down company and by someone else as well. Interestingly, the time gaps for A are larger than for B, both when A is faster and also when it is slower than others in detecting the same websites. Whenever A is slower to identify phishing websites, it is 41 hours slower than the first reports on average. This corresponds to the average 37-hour gap between sites identified by others first and sites only found by A. Similarly for B, the average 15-hour difference whenever its reports are slower matches

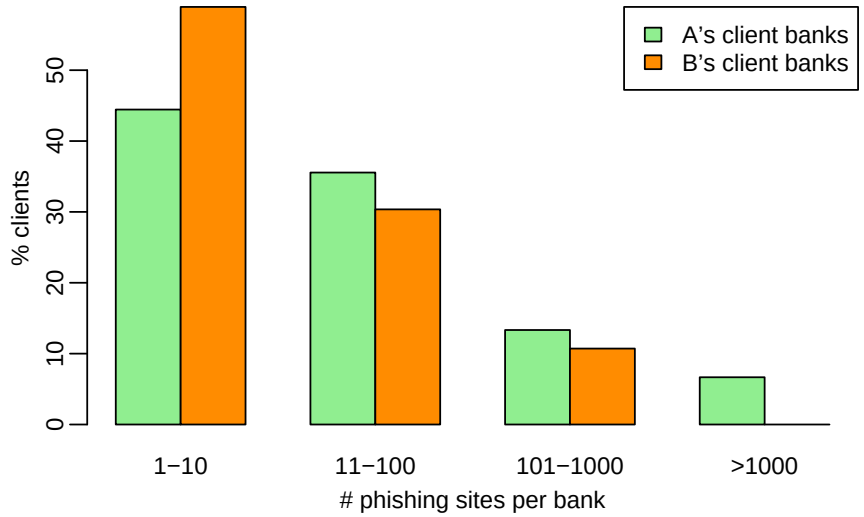


Figure 8.2: Distribution of client size (measured by the number of phishing sites observed) for take-down companies A and B.

the 11-hour gap between lifetimes when B finds sites alone and when other feeds pick up the websites first. Hence, the data in this table reinforce the connection between delays in reporting and longer phishing-website lifetimes.

8.1.3 Bigger targets harmed more by non-cooperation

Table 8.1 also reveals substantial differences between the composition of A's and B's clients. Slightly more of B's clients were attacked during the period of our study (66 to A's 53), yet B's clients are impersonated much less frequently (2 178 to A's 7 106). Many of B's clients are smaller banks and credit unions, whereas A's client base includes several large national banks, which attract more criminal interest.

Figure 8.2 shows the distribution of clients according to the number of observed phishing websites per client. (Note the logarithmic x-axis.) Nearly 60% of B's clients were impersonated fewer than 10 times during the sample period, compared to around 40% for A. Notably, around 5% of A's clients were impersonated more than 1 000 times, while none of B's were. These highly targeted clients account for much of the difference in the number of websites removed.

Given such a wide disparity, it is worth examining whether the number of phishing websites per client affects how likely outside sources are to contribute. Figure 8.3 plots the proportion of client phishing sites detected by A (left) and B (right) compared to other sources as the number of impersonating websites varies. For example, around 75% of phishing sites were only identified by A for A's clients targeted fewer than 10 times. This proportion steadily decreases, finally to around 30%, for clients impersonated more than

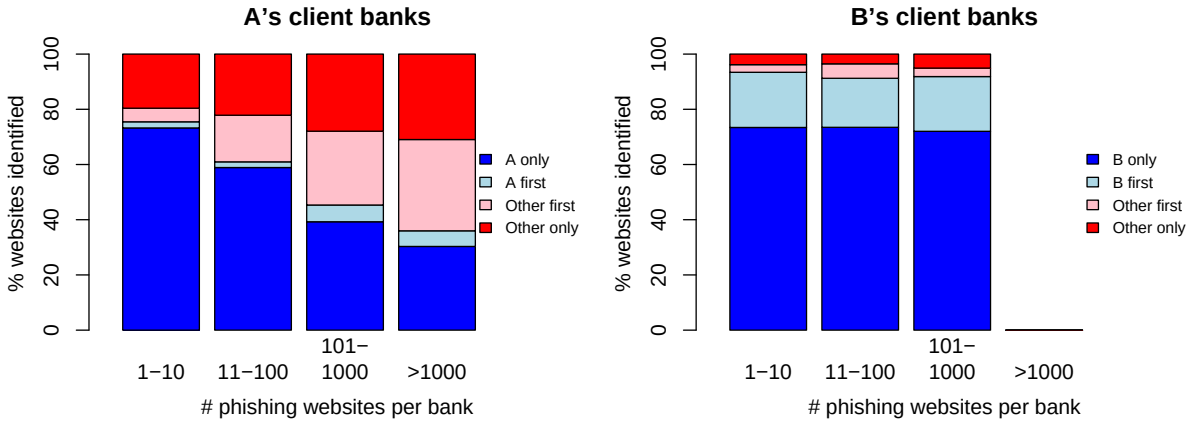


Figure 8.3: Proportion of client phishing sites identified by take-down companies and other sources (Company A – left, Company B – right). For both companies, as the number of observed phishing sites per client increases, outside sources identify a larger proportion of sites.

1 000 times. For these highly targeted clients, other feeds uniquely identified around 30% of websites, followed by an additional 35% of websites picked up by others before A.

B's clients follow a similar pattern, albeit on a reduced scale. B misses only 4% of the websites impersonating clients targeted fewer than 10 times, but misses 11% of websites for clients impersonated between 100 and 1 000 times. B does not have any clients impersonated over 1 000 times, but the trend is for a higher proportion of sites to be missed in such a case. So we conclude that B's feed, while consistently more complete and faster than A's, would benefit less from outside help than A simply because it has more rarely attacked clients and fewer highly targeted ones.

There are several reasonable explanations for the effect we have just observed. Banks and credit unions send the details of websites they learn about to the take-down companies for removal. Since only the company they have hired can be expected to take any action, they are unlikely to inform anyone else, which explains why the sites are missing from other sources. In addition, when only a handful of phishing sites are created, the bank may be able to identify all of them, so the more general phishing-site detection mechanisms used by the take-down companies are not as helpful. When there are many sites, by contrast, it is also likely that there has been a stronger spam effort. In these circumstances, a bank-provided list is unlikely to be sufficient or timely. The take-down companies do use spam traps and other proprietary methods of identifying phishing websites. However, these techniques are unlikely to be comprehensive, and they are likely to miss more sites whenever many are being created.

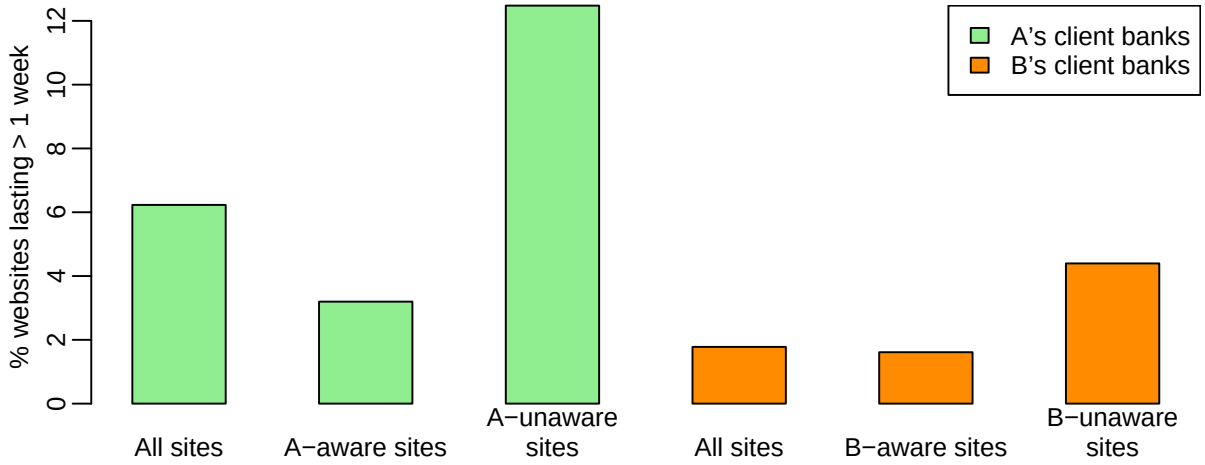


Figure 8.4: Proportion of websites lasting more than one week depending on who knows about them.

8.1.4 The effect of information sharing on long-lived sites

In Chapter 7, we found that the distribution of phishing-website lifetimes corresponds to a lognormal distribution, which is very skewed. This means that most websites are removed quickly, but there is a 'long tail' of websites that remain for much longer, even for many weeks. There are several explanations for the existence of long-lived phishing websites. One is that the sites are hosted in places with unresponsive owners and ISPs. Another is that the take-down company or bank is unaware of the website entirely. Examining the feeds for both A and B, we find evidence that not knowing of a website increases the chances that it will remain for a longer period.

We studied the proportion of websites impersonating A's and B's client banks that remain alive for more than one week. Our findings are presented in Figure 8.4. Overall, 6.2% of websites impersonating A's clients remain for at least one week. However, just 3.2% of the websites that A was aware of remained for that long. Strikingly, 12.5% of websites missed by A but identified by others remain for more than a week. This higher proportion once again suggests that knowing about a website has a great impact on whether it will be removed.

If the other sources had shared their feeds with A, then A would have tried to take them down, and so only 3% might be expected to remain after a week. Thus $2\,206 \times (12-3)\% = 199$ websites impersonating A's clients might be removed quickly instead of hanging on for much longer.

Similar results can be seen for B, albeit on a smaller scale. Overall, 1.8% of websites impersonating its client banks remain for more than one week, but if B knows about them the proportion is 1.6%; whereas 4.4% last a week or more if B is ignorant of their existence.

	Total exposure	Exposure due to not sharing
A's client banks	\$117m	\$62m
B's client banks	\$20m	\$799k

Table 8.3: Banks' estimated financial exposure due to not sharing phishing feeds.

8.1.5 What is the cost of non-cooperation?

In Section 7.2.3 we obtained a rough estimate for the cost of phishing. Our method estimated the number of victims each phishing website snagged by examining usage statistics for a number of the phishing websites we were tracking. We then combined this measure with our knowledge of phishing-website lifetimes and an estimate from Gartner that \$572 is lost for each phishing victim on average [64]. We apply our method here to quantify the financial exposure to banks caused by not sharing information about phishing websites.[2]

Two formulas are required for estimating the financial exposure to banks caused by phishing. The first formula estimates the number of victims for each phishing site, using website lifetimes and our earlier estimates of victims over time:

$$\frac{\#\text{victims}}{\text{site}} = \text{mean lifetime} \times \frac{8.5 \text{ victims}}{24 \text{ hrs}} + 8.5 \text{ victims before detection.}$$

The second formula computes the estimated exposure by multiplying the victim rate by the number of sites and the cost per victim according to Gartner:

$$\text{Estimated loss} = \frac{\#\text{victims}}{\text{site}} \times \#\text{sites} \times \$572.$$

For example, we can compute the total financial exposure of A's banks, given that the average lifetime of all 7 106 websites impersonating A's banks is 57.4 hours.

$$\text{Estimated loss}(A) = (57.4 \text{ hrs} \times \frac{8.5 \text{ victims}}{24 \text{ hrs}} + 8.5 \text{ victims}) \times 7\,106 \text{ sites} \times \$572 = \$117\text{m}.$$

We can also estimate what portion of the \$117 million is caused by not sharing feeds. To do this, we must first determine the estimated lifetime of sites whenever A knows about them, and then calculate the difference in time for the sites missed or identified later by A. To compute the lifetime of sites A knows about, we subtract the average difference for sites identified by others first, arriving at an average of 13.9 hours.

If the 2 219 websites missed by A had instead been identified by A, we would expect their lifetimes to shorten from 112.2 hours to 13.9 hours. This represents a financial exposure of $(112.2 - 13.9)$ hrs $\times \frac{8.5 \text{ victims}}{24 \text{ hrs}} \times 2\,219$ sites $\times \$572 = \$44\text{m}$. But that is not all. We

[2]Note that our earlier \$160m figure for the losses due to phishing is an underestimate because it only considered phishing sites from one community source. The data analyzed in this chapter is gathered from additional sources, each of which contributes websites missed by the others.

also have to account for the difference in the 2 225 websites identified by A 40.9 hours more slowly than other sources on average. This translates to a financial exposure of 40.9 hrs $\times \frac{8.5 \text{ victims}}{24 \text{ hrs}} \times 2\,225$ sites $\times$ \$572 = \$18m. Therefore, the financial exposure to A's client banks caused by not sharing feeds is \$62 million for one quarter (our sample period), or \$248 million at an annualized rate. Table 8.3 presents the complete figures, along with the lower results for B's banks.

It is impossible for us to extrapolate from these figures to the entire anti-phishing industry. A and B are both significant players within this industry, but other companies also have substantial numbers of clients, so the overall exposure caused by not sharing is in fact much higher. Note that these figures do not incorporate the lack of cooperation when tackling rock-phish attacks, which are reckoned to account for nearly half of all phishing. We discuss rock-phish attacks in the next section.

8.2 Cooperation and rock-phish attacks

Rock-phish attacks represent a common threat to the banking industry – up to 25 banks are impersonated simultaneously within each domain, and all currently impersonated banks may be reached from any live domain. Hence, there is automatically implicit cooperation in the removal of rock-phish domains because whoever gets the domain suspended stops all of the attacks on other banks simultaneously. Nonetheless, we have found no evidence that any explicit cooperation is occurring at present. While many banks and take-down companies do not fully understand the nature of rock-phish attacks, the take-down companies A and B certainly do, as their clients have been targeted for some time. They track rock-phish websites, even when their clients are not currently attacked. However, they wait until their own clients are targeted (i.e., when they are paid for their efforts) before starting to actively remove rock-phish domains.

We studied A's and B's clients that were targeted by the rock-phish gang during the sample period. Five of A's clients and two of B's were attacked. Figure 8.5 shows the days when new rock-phish domains appear and their clients are one of the targets. B's clients were attacked throughout October until mid-November, while A's clients were attacked briefly in early October and again beginning in late October through early December. For most of December, neither A's nor B's clients were attacked. Figure 8.5 also plots (as the blue line) the average lifetimes of rock-phish domains depending on the day on which the domains were launched. It is immediately apparent from this plot that activity by A and B significantly shortens the lifetime of rock-phish domains. When A and B are not actively removing rock-phish domains, as happened during December, the domains remain up for much longer.

Table 8.4 provides further insight into the impact that A and B have on defending against rock-phish attacks. We observed 2 458 rock-phish domains over 92 days. On average,

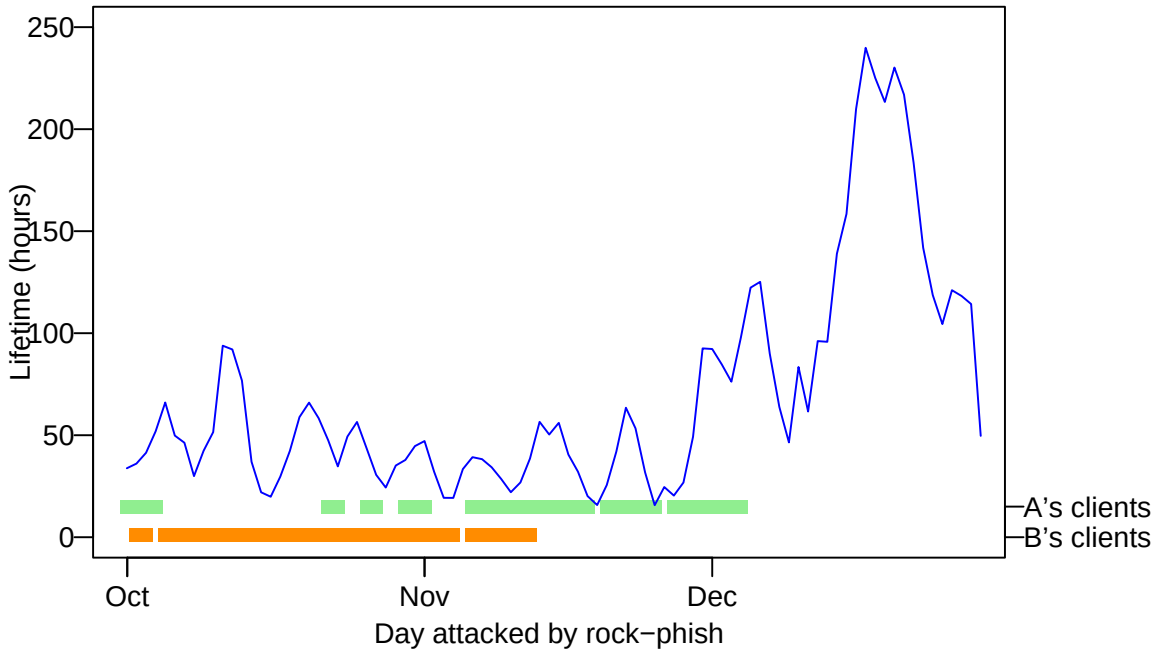


Figure 8.5: Timeline for rock-phish attacks.

	Rock-phish sites		Lifetime (hrs)	
	#	days	mean	med.
All rock-phish	2 458	92	73.1	34.0
Neither A nor B attacked	739	30	141.2	76.4
Only A attacked	590	22	43.0	21.3
Only B attacked	553	24	47.8	33.5
A and B attacked	576	16	40.8	27.6

Table 8.4: Variation in rock-phish site lifetimes whether A's and B's clients are targeted.

these domains are removed after 73 hours, about three days. However, rock-phish domains launched on days when neither A's nor B's clients are attacked last 141 hours, nearly twice as long as average. On days when only A's clients are attacked, domains are removed within 43 hours – which is considerably faster. The story is similar for days when only B's clients are attacked (48 hours), and domains are removed fastest (41 hours) on days when clients of both A and B are attacked.

We also tested the feeds' coverage of rock-phish domains. Figure 8.6 shows Venn diagrams for A, B and others. Despite considerable domain reuse when targeting several banks simultaneously, there remain significant gaps in coverage. A knows about 1 818 domains, but is unaware of 628 domains, 26% of the total. Similarly, B is aware of 2 072 domains but missed 374, 15% of the total. If A and B exchanged rock-phish feeds both could defend their clients more effectively. The only potential impediment to sharing is that take-down revenue would be more evenly spread whenever both have clients under attack.

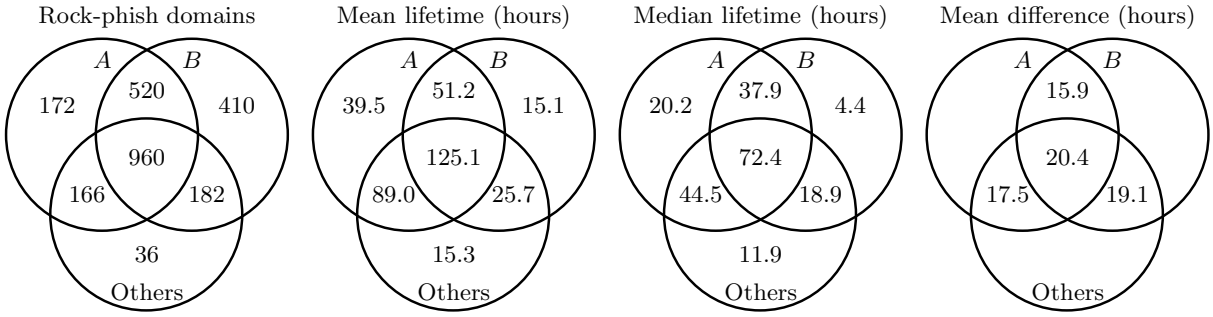


Figure 8.6: Comparing awareness of rock-phish domains for take-down companies A and B.

There may be a disincentive to share for a company that thought it had a more extensive list of domains than its competitors.

While A and B would benefit from exchanging rock-phish feeds, other companies and banks with less complete feeds stand to gain even more. One explanation of the much longer-lived rock-phish domains in December (see Figure 8.5) is that the other firms being targeted do not know about the domains A and B have discovered. Sharing their feeds with less-informed banks and take-down companies might greatly strengthen efforts to tackle rock-phish.

The middle two Venn diagrams in Figure 8.6 give the domain lifetimes based upon when different groups became aware. Surprisingly, it appears that the more feeds a domain appears in, the higher the average lifetime. Domains only appearing in the feeds from A, B and elsewhere last for 39.5, 15.1, and 15.3 hours, respectively. The figures increase if the domain is picked up by one of A, B and others, and increase even more to 125.1 hours on average (72.4 hours median) when the domain is identified by all three sources. This result runs counter to intuition – we might expect that more widely-known domains would be more likely to be removed, since at any given time at least one motivated defender has identified the domain and should be trying to get it suspended.

So what might explain this effect? The first explanation is that our monitoring is likely to become aware of domains earlier if several organizations are picking them up, and some detect the domain's usage faster than others. The Venn diagram in Figure 8.6 (right) examines the overlap and shows the difference between the first organization to identify a site and the last. Whenever two groups identify the domain, the lag between the first discovery and the second ranges from 15.9 to 19.1 hours. Whenever a domain is rediscovered three times, the difference between the first and last identification is slightly higher, averaging 20.4 hours. This difference accounts for a portion of the disparity between lifetimes.

Another reason why rediscovered websites last longer on average is that there may be a selection bias. Domains that remain around long enough to be rediscovered several times

are more likely to be difficult to remove, whereas domains that can be removed shortly after the first organization finds it may be removed before the other detectors get a chance to notice them. It is difficult to determine the magnitude of this bias.

It should be carefully noted that any selection bias that is present would also influence our earlier results for the lifetimes of ordinary phishing websites described in Section 8.1 above. However, because this tends to inflate the lifetimes we measure for multiply detected websites, it serves to strengthen our finding that websites appearing in only one feed have longer lifetimes.

8.3 Can sharing work in the anti-phishing industry?

On the face of it, the anti-phishing industry cooperates quite a bit. There is the APWG, which conducts regular meetings where members share tips on the latest attacker innovations. It also collects and disseminates a feed of phishing websites. But who exactly contributes to this feed? It seems to primarily come from third parties who have access to spam data, or who build browser toolbars, but are not directly interested in removing sites. Once aggregated, this feed is then distributed to APWG members who are in the business of removing phishing websites.

There are a number of other organizations which solely *produce* lists of phishing URLs and many of these are happy to share their data with other organizations. There are numerous organizations that mainly *consume* data, generating very little of their own. However, the take-down companies find themselves in a more curious position. They happily accept phishing feeds from any organization willing to share. But there is seldom reciprocity, or a willingness to share with anyone who asks. Take-down companies often market themselves as having a unique and valuable insight into phishing, which other companies do not possess. Some companies are proactive in selling their feed, or at least those URLs which target a given client. Hence, their own contributions serve as a differentiator and these URLs are not shared, as evidenced by the substantial number of websites uniquely identified by both A and B in earlier sections of this chapter.

Given that the feeds could better inform take-down companies and banks, we now weigh the merits and drawbacks of companies sharing their phishing feeds. We envision a third party collating the feed and distributing it to the industry, including the competitors of those who supplied the data. This third party could be the APWG, expanding their existing feed to include contributions from all possible producers and specifically the take-down companies.

8.3.1 Incentive analysis

Banks targeted by phishing attacks would benefit most from sharing feeds. Any reduction in the lifetimes of phishing websites would be welcome. From the discussion in Section 8.1.3, larger banks that are targeted more often stand to gain the most.

Take-down companies would also benefit, through improved service for their clients and increased revenue per client as more websites are identified. Inevitably, the benefit to a particular company can vary greatly. This depends upon client composition (see Section 8.1.3), along with differences between the speed and comprehensiveness of detection. It is true that our analysis has shown that company B stands to gain less from sharing than company A, due to both B's more comprehensive feed and its large base of small clients. However, B would still be able to offer marginal improvements to its clients, and it seems entirely likely that other take-down companies would benefit too.

Of course, there are disadvantages to sharing as well – which helps to explain why it is not happening already! Phishing feeds do have some inherent value. Some take-down companies emphasize the selling of feeds to clients, so they may take a dim view of 'giving them away for free'. For most companies, though, reduced sales from feeds should be dwarfed by the prospect of increasing the number of sites that are known about and can therefore be taken down and charged for. It would require some changes in marketing stance: take-down companies compete on a number of factors, including price, customer service, speed of removal, and feed completeness. Sharing feeds would eliminate one competitive aspect, and well-established companies with comprehensive feeds (such as B) may view their feed as giving them a major advantage over less-informed firms when selling services to a client. In such a case, it would be in their interest to refuse sharing with weaker competitors.

Were widespread sharing of phishing feeds to happen, it would have significant competitive implications. Most importantly, sharing would substantially lower barriers to entry for prospective take-down firms. This would be good for competition within the take-down industry, and consequently helpful to banks, take-down companies' primary clients. By contrast, reduced barriers to entry would be viewed negatively by established take-down companies.

Widespread sharing could allow some companies to free-ride by taking the feed without investing any effort in finding new sites to contribute. While this is a significant concern, the existence of thriving community feeds like PhishTank and the APWG list suggests that a significant proportion of sites are already detected completely independently of the take-down companies. Furthermore, take-down companies should remain incentivized to continue operating their own detection systems because they wish to identify phishing sites that impersonate their own clients; passing along irrelevant sites detected at the same time is unlikely to substantially increase their costs.

Another form of free-riding is possible as banks face common threats like rock-phish attacks. Section 8.2 revealed diminishing lifetimes for rock-phish domains as more defenders get involved. This is hardly surprising. Defending against rock-phish attacks can be considered a 'sum-of-efforts' problem, where total protection depends on the aggregate contributions from each defender. Game-theoretic analysis of similar circumstances [157] highlighted the potential for less efficient players to free-ride off the efforts of the higher-motivated. Take-down companies are highly motivated to remove phishing sites that target their clients, since they are compensated for removing them. However, banks who do not outsource website removal may be tempted to free-ride off the efforts of the more motivated take-down companies so long as the take-down companies mitigate the threat from rock-phish more than the expense of removing sites directly.

8.3.2 Other information-sharing examples

At this point it is helpful to consider other threats to information security where sharing has or has not happened. We first consider the anti-virus industry. In its early days, anti-virus companies did not share virus definitions; instead, they differentiated themselves by the comprehensiveness of their lists. Trade magazines published head-to-head comparisons of competing products, testing whether 'Dr. Solomon' caught more viruses than 'Norton'. However, this produced significant biases in the results depending upon who supplied the virus samples that were tested, and this led to some sharing of data. Over time the industry stopped hoarding viruses, and instead began sharing virus samples with their competitors.

Today, whenever a virus is identified, it is first published to a common list so that each company can develop its own detection 'signature' as quickly as possible. Consumers benefit from more comprehensive virus detection, and the companies compete on other factors (such as price or levels of support). In fact, it is now viewed as extremely bad manners to refuse to share a virus sample, as evidenced by the industry's recent uproar over a newcomer's reticence to pass on information about a mobile phone virus [98]. To keep potentially harmful information from reaching outsiders (and forestall free-riding), the group remains quite exclusive and shares only between established members who have demonstrated value to the group. Such clubbiness is occasionally railed against by newcomers, such as when the organizers of a malware repository publicly pleaded with the industry to share [90].

A second lesson about sharing can be drawn from the world of vulnerability disclosure. Some security researchers advocate full and immediate disclosure: publishing details (including exploit code) on mailing lists such as Bugtraq [143]. While undoubtedly prompting the vendors to publish a patch, full and immediate disclosure has the unfortunate side effect of leaving consumers immediately vulnerable. A more balanced alternative is 'responsible disclosure' as pioneered by CERT/CC in the US. CERT/CC notifies vendors to

give them time to develop a patch before disclosing the vulnerability publicly. When the vulnerability is finally disclosed, no exploit code is provided.

Empirical analysis comparing the patch-development times for vulnerabilities reported to Bugtraq and to CERT/CC revealed that CERT/CC's policy of responsible disclosure led to faster patch-development times than Bugtraq's full disclosure policy [19]. This is because CERT/CC has developed a more constructive relationship with software vendors, working with them to fix vulnerabilities.

Some firms, led by iDefense and Tipping Point, have gone a step further by actively buying vulnerabilities. Their business model is to provide vulnerability data simultaneously to their customers and to the vendor of the affected product, so that their customers can update their firewalls before anyone else. However, the incentives in this model have been shown by Kannan and Telang to be suboptimal: users who do not participate in the closed circle of subscribers cannot protect their systems in time [84].

8.3.3 Recommendation

The anti-phishing industry has an important choice to make: whether to increase sharing, following the anti-virus industry's example, or to continue leveraging their feeds as a competitive edge, as is currently the case with some vulnerability brokers. In 2006, the anti-phishing industry appeared to be at the same point as the early days of the anti-virus industry, arguing over the completeness and accuracy of each other's anti-phishing toolbars [112].[3] Today, the APWG feed shows that some cooperation is occurring, and it has proved to be an effective way of getting third parties to contribute the URLs they learn about.

We believe that the evidence is strongly in favor of choosing to evolve beyond the current arrangements, much as the anti-virus industry did, and start viewing all phishing feeds as public goods rather than keeping some of the information private. Stopping short of a fully public arrangement, by instituting a 'sharing club' might appeal to take-down companies by addressing issues of market entrance – but as we have already noted, this reduces competition, and so the banks may pay more than otherwise. Additionally – since feeds are also used by the anti-phishing toolbars – it may not be in the wider consumer interest either. Anti-virus companies exchange virus samples, but each verifies the sample's legitimacy and develops its own signatures. Similarly, take-down companies add value by cleaning up phishing feeds and standing behind their assessments. Conceivably, sharing could happen by exchanging raw feeds, and each take-down company could continue charging for processed feeds to clients who want them.

[3]Meanwhile, academic research by Zhang *et al.* has contradicted the industry's sponsored research, showing none of the toolbars to be satisfactory [171].

Whatever the minutiae of the change, in our view, sharing feeds is a winning proposition for most take-down companies: better coverage can lead to increased revenue and improved customer service. For the banks the issue is a 'no brainer': sharing feeds means that phishing websites that attack their brands are removed more quickly. Only the few companies which specialize in producing feeds, and do little take-down of their own, can have any reasonable objection to this change of approach.

It is our recommendation that the take-down companies start sharing feeds immediately. Significantly, the banks, who pay the take-down companies for their services, can use their financial clout to encourage this change to happen. We contrast this relatively small number of clients, each with significant purchasing power, with the much broader spectrum of mainly individual customers to which the anti-virus industry sells. We would suggest that this concentration of purchasing power could enable comparatively rapid change in the anti-phishing industry. Should this change not occur, then a regulator might intervene. Regulations mandating sharing, while in our view economically justifiable in light of the data analysis presented in this chapter, are unlikely to be practical or timely enough to be effective.

8.4 Related work

Weaver and Collins examined two phishing feeds [162] and found that they did not share URLs. They computed the overlap between the feeds and applied capture-recapture analysis to estimate the number of overall phishing attacks. Gordon and Ford discussed early forms of sharing in the anti-virus industry and contrasted it with sharing when disclosing vulnerabilities [68].

Section 1.1 discusses several research threads relevant to this chapter. We reiterate some of the key ideas here and relate them to cooperation in the anti-phishing industry.

Information sharing has been recognized as an important way to improve information security. Worried about protecting critical infrastructures owned by private industry, the US government has encouraged data exchange via closed industry groups known as Information Sharing and Analysis Centers (ISACs). However, making information sharing occur in practice has often proven difficult. The development of ISACs has yielded mixed results. While some industries responded quickly, others took several years to comply. Many firms have expressed concerns over sharing security information with competitors and with the government [43]. Another worry about information sharing explored in the academic literature is that firms might free-ride off the security expenditures of other firms by only 'consuming' shared security information (e.g., phishing feeds) and never providing any data of their own [69].

There can also be positive economic incentives for sharing security information. Gal-Or and Ghose developed a model where sharing can trigger additional security invest-

ment. [60] In many circumstances, the providers of security services stand to gain by sharing information, which can drive up demand. Where there is a lack of industry awareness of threats, sharing information can certainly foster broader investment. This tendency to simultaneously share information and spend more on security has a greater effect on fiercely competitive industries, such as take-down companies, where product substitutability is high. Gal-Or and Ghose also found that formal sharing organizations are more effective (in terms of information sharing and investment spurred) when members join sequentially. By joining first, market-leading firms bootstrap the alliance and demonstrate their commitment to share information, which encourages others to subsequently join. So if the more established take-down companies take the initiative and share feeds, others are likely to follow.

Often, overall security levels depend on the efforts of many interdependent principals. Varian developed a stylized game-theoretic model to explore dependability issues in information systems – where performance depends on the minimum effort, the best effort, or the sum-of-efforts [157]. Program correctness can depend on minimum effort (the most careless programmer introducing a vulnerability) while software validation and vulnerability testing may depend on the total of everyone's efforts. Similarly, defense against common threats like rock-phish attacks rely on the sum of efforts from all banks targeted. Further, constructing the most complete phishing feeds requires aggregating everyone's contribution. Varian's analysis predicts that the principals who stand to gain most will carry out the bulk of the effort, leaving others to free-ride. Since take-down companies are compensated for taking action they will tend to find themselves in the former category, contributing when it helps their clients – especially if those clients are insisting upon the best possible service.

8.5 Conclusion

This chapter has studied feeds of phishing-website URLs obtained from two take-down companies hired by the banks to remove phishing websites. While some banks share feeds, take-down companies currently fail to share with each other. This chapter empirically demonstrated the adverse impact of such non-cooperation. Having examined data for the bank clients of the two take-down companies, we found that websites had consistently longer lifetimes when the take-down company was either completely unaware they existed, or when they belatedly learned of them. This effect was most apparent for banks that were frequently attacked, whereas it was less obvious, but still non-trivial, for small credit unions that might only be attacked on a handful of occasions. We also showed that websites were far more likely to remain active for more than a week if the responsible take-down company was unaware of their existence.

We calculated how much shorter website lifetimes would be if the take-down companies

were to share information with each other. There is a direct link between longer take-down times and the funds put at risk by the compromise of visitor credentials. So we also translated these lifetimes from hours into dollars, finding that for these two companies alone – on some fairly rough estimates – around $250 million a year might be made safe.

We also examined take-down times for rock-phish domains and found that lifetimes were higher when no client of the two take-down companies was being attacked. We also demonstrated that once again each company was only seeing a part of the overall picture, and hence that lifetimes might be reduced by sharing information.

We considered the reasons why the take-down companies might not wish to share information, and concluded that in almost every case they would benefit, to a greater or lesser extent, from data sharing. We have therefore recommended that the industry change its practices as soon as possible. We noted in passing that the banks uniformly benefited from universal sharing and – since they were paying the bills – they were in a strong position to force change upon the industry.

The lessons we have learned, and the conclusions we have drawn for fostering cooperation, go much wider than phishing. Many pressing threats to information security, from botnets to malware, can now only be countered by piecing together disparate, incomplete data sources. Defenders should instead arrange to work more closely together in tackling common threats.

Chapter 9

Evaluating phishing-decision mechanisms

PhishTank is part of a growing trend to utilize web-based participation when implementing security mechanisms, from aggregating spam to tracking malware. Anyone may submit URLs of suspected phishing websites, and may vote on the accuracy of other submissions. In this chapter, we study participation in PhishTank in order to better understand the effectiveness of crowd-based security generally. The chapter also provides an empirical complement to the comparisons by analysis and simulation of the voting and suicide decision mechanisms for ad-hoc networks presented in Chapters 4 and 5. In this respect, voting in PhishTank corresponds to the blackballing and LEAVE protocols, while the unilateral decisions by take-down companies corresponds to suicide and Stinger.

We find that PhishTank is dominated by its most active users. Participation follows a power-law distribution, leaving it particularly susceptible to manipulation. We compare PhishTank with a proprietary source of reports, finding PhishTank to be slightly less complete and significantly slower in reaching decisions. We also evaluate the accuracy of PhishTank's decisions and discuss cases where incorrect information has propagated. We find that users who participate less often are far more likely to make mistakes, and furthermore that users who commit many errors tend to have voted on the same URLs. Finally, we explain how the structure of participation in PhishTank leaves it susceptible to large-scale voting fraud which could undermine its credibility. We also discuss general lessons for leveraging the 'wisdom of crowds' in taking security decisions by mass participation.

9.1 Data collection and analysis

9.1.1 Phishing-website reporting and evaluation

We examined reports from 200 908 phishing URLs submitted to PhishTank between February and September 2007. Voting was suspended for 24 254 of these because the websites in question went offline before a conclusive vote was reached. In these cases, we could only determine who submitted the record and not who voted on it. We gathered completed votes for the remaining 176 366 submissions. 3 786 users participated by submitting reports and/or voting.

In all, 881 511 votes were cast, implying an average of 53 submissions and 232 votes per user. However, such averages are very misleading. Small numbers of users are responsible for the majority of submissions and votes. The top two submitters, adding 93 588 and 31 910 phishing records respectively, are actually two anti-phishing organizations that have contributed their own, unverified, feeds of suspect websites. However, neither verifies many submissions. The top verifiers have voted over 100 000 times, while most users only vote a few times.

Many of the leading verifiers have been invited to serve as one of 25 PhishTank moderators. Moderators are granted additional responsibilities such as cleaning up malformed URLs from submissions.[1] Collectively, moderators cast 652 625 votes, or 74% of the total. So while the moderators are doing the majority of the work, a significant contribution is made by the large number of normal users.

9.1.2 Duplicate submissions in PhishTank

PhishTank asks its users to vote on every unique URL that is submitted. Unfortunately, this imposes a very large and unnecessary burden on its volunteers due to the use of wildcard DNS by the rock-phish gang.

Transmitting unique URLs trips up spam filters looking for repeated links, and also fools collators like PhishTank into recording duplicate entries. Consequently, voting on rock-phish attacks becomes very repetitive. We observed 3 260 unique rock-phish and fast-flux domains[2] in PhishTank. These domains appeared in 120 662 submissions, 60% of the overall total. Furthermore, 893 users voted a total of 550 851 times on these domains! This is a dreadfully inefficient allocation of user resources, which could instead be directed to speeding up verification times, for example.

[1] Moderators also, on some rare occasions, use their powers to preemptively remove obviously incorrect submissions.

[2] For clarity of presentation, we refer to rock-phish and fast-flux domains as simply 'rock-phish' throughout this chapter.

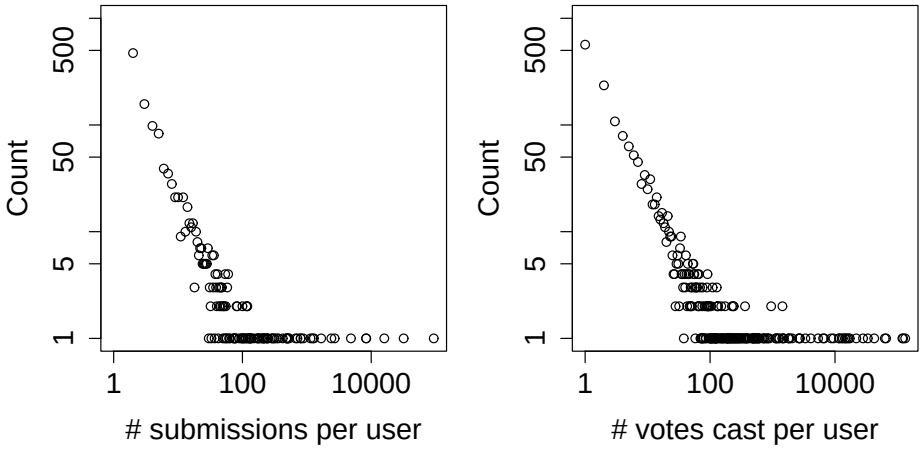


Figure 9.1: Density of user submissions (Left) and votes (Right).

Further duplication exists in the remaining 80 246 submissions. Using the techniques for identifying duplicate URLs described in Section 6.3.1, we arrive at 75 501 unique URLs.

9.2 Power-law distribution of user participation rates

The wide range of user participation is captured in Figure 9.1. Noting the log-log axes, these plots show that most users submit and vote only a handful of times, while also indicating that a few users participate many times more.

In fact, the distribution of user submissions and votes in PhishTank are each characterized by a power law. Power-law distributions appear in many real-world contexts, from the distribution of city populations to the number of academic citations to BGP routing topologies (see [123] for a survey). More precisely, the probability density function of a power law corresponds to $p(x) \propto x^{-\alpha}$, where α is a positive constant greater than one. Power-law distributions have highly skewed populations with 'long tails', that is, a limited number of large values appear several orders of magnitude beyond the much smaller median value. Power-law distributions are even more skewed than lognormal distributions.

The intuitive argument put forth in favor of the robustness of 'crowd-sourced' applications like PhishTank's phish verification mechanism is that the opinions of many users can outweigh the occasional mistake, or even the views of a malicious user. However, if the rate of participation follows a power-law distribution, then a single highly active user's actions can greatly impact a system's overall accuracy. This is why a power-law distribution invalidates the standard Byzantine Fault Tolerance view of reliability [93]: subverting even a single highly active participant could undermine the system. In Section 9.5, we study how the skewed structure of participation rates in PhishTank could cause trouble.

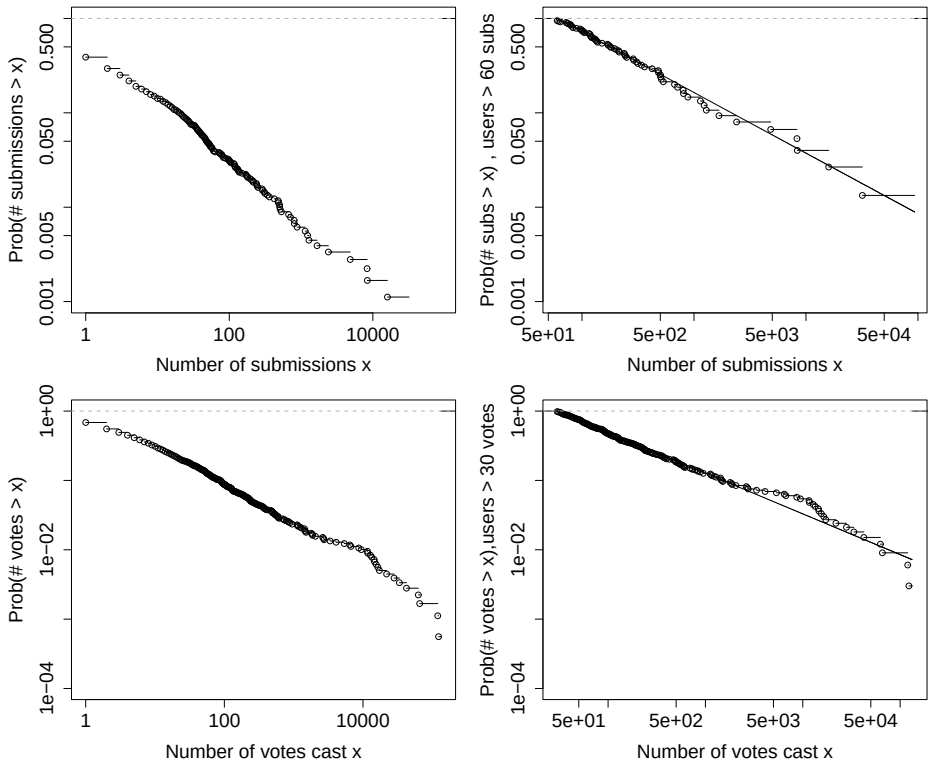


Figure 9.2: Complementary CDF of user submissions (Top Left) and votes (Bottom Left). Tail of submission CDF with power-law curve fit (Top Right), $\alpha = 1.642$ and the number of submissions per user at least 60. Tail of vote CDF with power-law curve fit (Bottom Right), $\alpha = 1.646$ and the number of votes per user at least 30.

Figure 9.2 (top left) plots the complementary cumulative distribution function (CDF) of user submissions. Both axes are logarithmic in scale. Figure 9.2 (bottom left) plots the CDF for the number of votes. Power-law distributions appear as a straight line on log-log axes, so visual inspection suggests that PhishTank data is likely to be distributed in this way. We have examined the tails of the voting and submission distributions to determine whether the data are consistent with a power-law tail.

The CDF for a power-law distribution is given by:

$$Pr(X > x) = \left(\frac{x}{x_{\min}}\right)^{-\alpha+1}$$

For the submission data, we tested the tail by considering only those users who submit at least $x_{\min} = 60$ times, while we set $x_{\min} = 30$ for the voting data. We estimated the best fit for α using maximum-likelihood estimation. We then evaluated the fit by computing the Kolmogorov-Smirnov test [105]. The results are given in the following table:

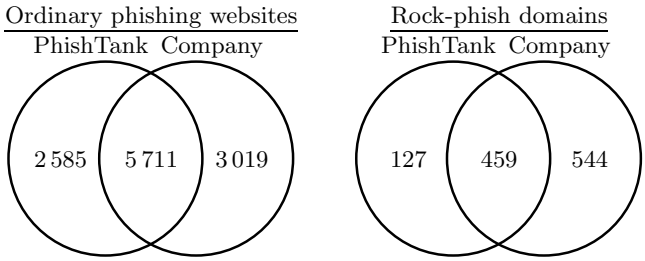


Figure 9.3: Venn diagram comparing coverage of phishing websites identified by PhishTank and a take-down company.

	Power-law distribution		Kolmogorov-Smirnov	
	α	$x_{\min}$	D	p-value
Submissions	1.642	60	0.0533	0.9833
Votes	1.646	30	0.0368	0.7608

Given the large p-values and small D values from the Kolmogorov-Smirnov test, we can say with high confidence that both the submission and voting data are consistent with a power-law distribution. Figure 9.2 (top and bottom right) presents the CDF for the tails of the submission and voting data, respectively, along with a line showing the power-law fit.

9.3 Comparing open and closed phishing feeds

As explained in earlier chapters, PhishTank is only one of several organizations tracking and classifying phishing websites. Other organizations do not follow PhishTank's open submission and verification policy. Instead, they gather their own proprietary lists of suspicious websites, and employees determine whether they are indeed phishing. We have obtained a feed from one such company. In this section, we examine the feeds of PhishTank and the take-down company to compare completeness and speed of verification.

9.3.1 Phishing-website identification

We compared the feeds during a 4-week period in July and August 2007 (see Figure 9.3 for Venn diagrams). We first examine ordinary phishing websites, excluding rock-phish URLs. PhishTank reported 10 924 phishing URLs, while the company identified 13 318. After removing duplicates, the numbers became much closer: 8 296 for PhishTank and 8 730 for the company. The two feeds shared 5 711 reports in common. This means that 3 019 reports were unique to the company's feed, while 2 585 reports only appeared in PhishTank. Hence, although neither feed is comprehensive, the company's feed contains a slightly wider selection of websites than PhishTank achieves.

For rock-phish URLs the difference is starker. PhishTank identified 586 rock-phish domains during the sample, while the company detected 1 003, nearly twice as many. Furthermore, the company picked up on 459, or 78%, of the rock-phish domains found in PhishTank, and detected 544 that PhishTank had missed.

By examining the overlap between the feeds, we can gain some insight into the company's sources. The overlap for all phishing reports corresponded to 9 380 submissions to PhishTank. 5 881 of these submissions, 63% of the total overlap, came from a user called *PhishReporter*, that we understand to be an anti-phishing report collation organization in its own right. This certainly implies that the company and PhishTank both receive a feed from *PhishReporter*. However, the remaining reports are more widely distributed, coming from 316 users. Unfortunately, we cannot say with any certainty whether these reports were also given to the company or if they were independently rediscovered.

It is noteworthy that both feeds include many phishing websites which do not appear within the other. This overlap is consistent with the comparison of the feeds from two take-down companies in Chapter 8.

9.3.2 Phishing-website verification

Given that prompt identification and removal of phishing websites is a priority, a feed's relevance depends upon the speed with which websites are reported and subsequently verified. Requiring several users to vote introduces significant delays. On average, PhishTank submissions take approximately 46 hours to be verified. A few instances take a very long time to be verified, which skews the average. The median, by contrast, is around 15 hours.

We also found that, on average, unanimous votes were verified slightly quicker than votes where there was disagreement, but that conflicting votes had a much shorter median (7 hours). URLs confirmed to be phishing were verified a few hours faster than those determined not to be a phishing website. The precise values are given in the following table:

Verification time	All entries	Conflict	Unanimous	Is-phish	Not-phish
Mean (hours)	45.6	49.7	45.8	46.1	39.5
Median (hours)	14.9	6.6	27.8	14.1	20.6

We also compared the submission and verification times for both feeds during the four-week sample. On average, PhishTank saw submissions first, by around 11 minutes, but the company verified submissions after an average delay of just 8 seconds.[3] However,

[3]We suspect that verification of any particular URL is in the hands of an individual on-duty employee, who often submits and verifies in a single operation.

PhishTank's voting-based system did not verify the URLs (and therefore did not disseminate them) until 16 hours later. For the rock-phish URLs, we compared the earliest instance of each domain, finding that overlapping domains appeared in PhishTank's feed 12 hours *after* they appeared in the company's feed, and were not verified for another 12 hours. The time differences between feeds are summarized in the following table:

ΔPhishTank	Ordinary phishing URLs		Rock-phish domains	
− Company	Submission	Verification	Submission	Verification
Mean (hrs)	−0.188	15.9	12.4	24.7
Median (hrs)	−0.0481	10.9	9.37	20.8

To sum up, voting-based verification introduces a substantial delay when compared to unilateral verification.

9.4 Testing the accuracy of PhishTank's decisions

Having compared the breadth and timeliness of PhishTank's reports to the closed source, we now examine the correctness of its users' contributions. Unfortunately, since the closed phishing feed does not provide a record of invalid submissions, we cannot compare its accuracy to PhishTank's. We first describe common causes of inaccuracy and discuss their prevalence. We then demonstrate that inexperienced users are far more likely to make mistakes than experienced ones. Finally, we show that users with bad voting records 'cluster' by often voting for the same phishing URLs.

9.4.1 Miscategorization in PhishTank

The vast majority of user submissions to PhishTank are indeed phishing URLs. Of 176 654 verified submissions, just 5 295, or 3%, are voted down as invalid. Most of these invalid submissions appear to be honest mistakes. Users who do not understand the definition of phishing submit URLs from their spam, while others add URLs for other types of malicious websites, such as those involved in advanced fee fraud (419 scams). However, a number of carefully crafted phishing websites have also been miscategorized and 'foreign-language' websites are sometimes classified incorrectly. Most commonly, an obscure credit union or bank that uses a different domain name for its online banking may be marked as a phish.

Yet there is even dissent among moderators as to what exactly constitutes a phish: 1.2% of their submissions are voted down as invalid. For example, some moderators take the view that so-called 'mule-recruitment' websites should be categorized as phishing because they are used to recruit the gullible to launder the proceeds of phishing crime. Other

mistakes may just be the result of fatigue, given that the moderators participate many thousands of times.

In addition to invalid submissions that are correctly voted down, submissions that are incorrectly classified present a significant worry. Identifying false positives and negatives is hard because PhishTank rewrites history without keeping any public record of changes. As soon as a submission has received enough votes to be verified, PhishTank publishes the decision. Sometimes, though, this decision is reversed if someone disputes the conclusion. In these cases, voting is restarted and the new decision eventually replaces the old one. Once we realized this was happening, we began rechecking all PhishTank records periodically for reversals. In all, we identified 42 reversals. We found 39 false positives – legitimate websites incorrectly classified as phishing – and 3 false negatives – phishing websites incorrectly classified as legitimate. 12 of these reversals were initially agreed upon unanimously!

We first discuss the false positives. 30 websites were legitimate banks, while the remaining 9 were other scams miscategorized as phishing. Sometimes these were legitimate companies using secondary domains or IP addresses in the URLs, which confused PhishTank's users for a time. However, several popular websites' primary domains were also voted as phish, including eBay (`ebay.com`, `ebay.de`), Fifth Third Bank (`53.com`) and National City (`nationalcity.com`). Minimizing these types of false positives is essential for PhishTank because even a small number of false categorizations could undermine its credibility.

Unsurprisingly, there are many more false positives than false negatives since the vast majority of submitted phishes are valid. However, we still observed 3 false negatives. Most noteworthy was incorrectly classifying as innocuous a URL for the rock-phish domain `eportid.ph`. Five other URLs for the same domain were submitted to PhishTank prior to the false negative, with each correctly identified as a phish. So in addition to the inefficiencies described in Section 9.1.2, requiring users to vote for the same rock-phish domain many times has enabled at least one rock-phish URL to earn PhishTank's (temporary) approval.

9.4.2 Does experience improve user accuracy?

Where do these mistakes come from? It is reasonable to expect occasional users to commit more errors than those who contribute more often. Indeed, we find strong evidence for this in the data. The left graph in Figure 9.4 plots the rates of inaccuracy for submissions and votes grouped by user participation rates. For instance, 44% of URLs from users who submit just once are voted down as invalid. This steadily improves (30% of submissions are invalid from users who submit between 2 and 10 URLs, 17% invalid for users with between 11 and 100 submissions), with the top submitters incorrect just 1.2% of the time.

A similar, albeit less drastic, difference can be observed for voting accuracy. Unfortu-

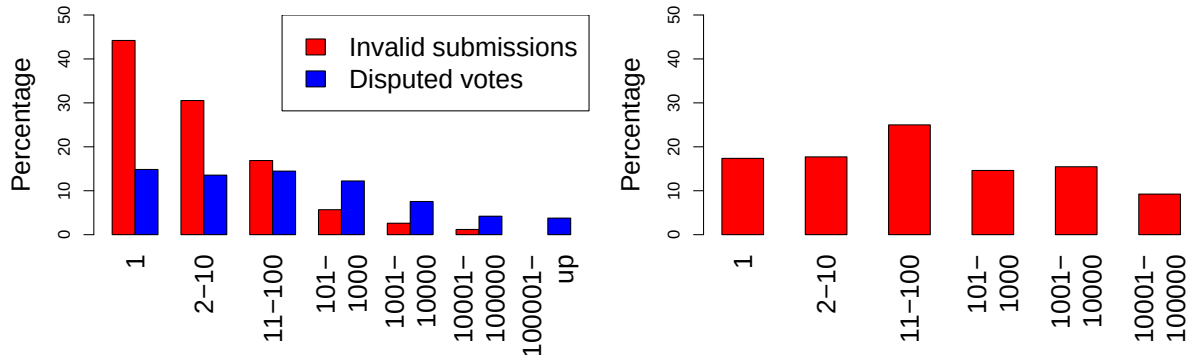


Figure 9.4: Inaccuracy of user submissions and votes according to the total number of submissions and votes per user, respectively (Left). Proportion of all invalid user submissions grouped by number of submissions (Right).

nately, we cannot determine with certainty whether a user has voted incorrectly (i.e., voted a submission as a phish when the majority said otherwise, or vice versa). This is because PhishTank does not publicly disclose this information. So we are left to devise a proxy for incorrectness using votes where there is disagreement (i.e., a mixture of yes/no votes). This is a reasonable approximation given that nearly all submissions (97%) are decided unanimously.

Users voting fewer than 100 times are likely to disagree with their peers 14% of the time. This improves steadily for more active users, with the most active voters in conflict just 3.7% of the time, in line with the overall average.

These results suggest that the views of inexperienced users should perhaps be assigned less weight when compared to highly experienced users.[4] However, we must note that simply ignoring low-contribution users would not eradicate invalid submissions and votes. Since most contributions come from experienced users, many of the errors can be traced to them as well. The right graph in Figure 9.4 groups user submissions together logarithmically, then plots the proportion of all invalid user submissions each group contributes. For instance, users submitting once contribute 17% of all invalid submissions. Users with fewer than 100 submissions collectively make 60% of the mistakes, despite submitting less than 7% of the phishing candidate URLs.

9.4.3 Do users with bad voting records vote together?

We now consider whether bad decisions reinforce themselves. More precisely, we ask whether users with bad voting records are likely to vote on the same phishing reports more often than randomly.

[4]Developers at PhishTank tell us that they have never treated users equally, but weigh their votes according to the user's accuracy over time.

We define a *high-conflict user* as one where a large fraction of votes f_{HC} cast are in conflict. We denote the set of all high-conflict users as HC, and the set of votes for user A as V_A.[5] T is the set of all phishing submissions, and $V_A \subset T$.

We denote high-conflict users as those where the majority of their votes are in conflict ($f_{HC} \geq 0.5$). Of 1 791 users who have voted, 186 are in high conflict. We now explore the relationship between these users.

We can empirically measure the observed overlap between high-conflict votes using the following formula:

$$\text{overlap}(HC) = \sum_{A \in HC} \sum_{B \in HC, B \neq A} |V_A \cap V_B|$$

If there is no relationship between the users, then we would expect their interactions to be random chance. Hence, we can develop a measure of the expected overlap in this case:

$$E(\text{overlap}) = \sum_{A \in HC} \sum_{B \in HC, B \neq A} \sum_{i=1}^{min(|V_A|,|V_B|)} i \times \frac{\binom{|V_A|}{i} \times \binom{|T|-|V_A|}{|V_B|-i}}{\binom{|T|}{|V_B|}}$$

If the overlap observed, overlap(HC), is greater than the overlap expected, E(overlap), then the high-conflict voters have tended to vote with each other more often than randomly. In our data, overlap(HC) = 254, while the expected overlap, E(overlap) = 0.225.[6] In other words, the rate of overlap in high-conflict voters is approximately one thousand times higher than would be the case if there was no connection between how high-conflict voters select their votes.

What are the implications? While it is possible that these high-conflict users are deliberately voting incorrectly together (or are the same person!), the more likely explanation is that users tend to make similar errors. When well-intentioned users vote incorrectly, they have apparently made the same mistakes.

[5] Technically, V_A represents the set of phishing submissions for which A has voted.

[6] We can simplify the combinations to yield a more computationally tractable formula:

$$\begin{aligned}
&E(\text{overlap}) \\
&= \sum_{A \in HC} \sum_{B \in HC, B \neq A} \sum_{i=1}^{min(|V_A|,|V_B|)} i \times \frac{\frac{|V_A|!}{i!(|V_A|-i)!} \times \frac{(|T|-|V_A|)!}{(|V_B|-i)!(|T|-|V_A|-|V_B|-i)!}}{\frac{|T|!}{|V_B|!(|T|-|V_B|)!}} \\
&= \sum_{A \in HC} \sum_{B \in HC, B \neq A} \sum_{i=1}^{min(|V_A|,|V_B|)} i \times \frac{|V_A|!}{i!(|V_A|-i)} \times \frac{|V_B|!}{(|V_B|-i)!} \times \frac{(|T|-|V_A|)!}{(|T|-|V_A|-|V_B|-i)!} \times \frac{(|T|-|V_B|)!}{|T|!} \\
&= \sum_{A \in HC} \sum_{B \in HC, B \neq A} \sum_{i=1}^{min(|V_A|,|V_B|)} i \times \frac{|V_A|!}{i!(|V_A|-i)} \times \frac{|V_B|!}{(|V_B|-i)!} \times \frac{(|T|-|V_A|)\cdots(|T|-|V_A|-|V_B|+i+1)}{|T|\cdots(|T|-|V_B|+1)}
\end{aligned}$$

9.5 Disrupting the PhishTank verification system

We now consider whether PhishTank's open submission and voting policies may be exploited by attackers. Recently, a number of anti-phishing websites were targeted by a denial-of-service attack, severely hindering their work in removing malicious sites [94]. Hence, there is already evidence that phishermen are motivated to disrupt the operations of groups like PhishTank. But even if enough bandwidth is provisioned to counter these attacks, PhishTank remains susceptible to vote rigging that could undermine its credibility. Any crowd-based decision mechanism is susceptible to manipulation. However, as we will see, certain characteristics of user participation make PhishTank particularly vulnerable.

9.5.1 Attacks and countermeasures

We anticipate four types of attacks on PhishTank:

1. Submitting invalid reports accusing legitimate websites.
2. Voting legitimate websites as phish.
3. Voting illegitimate websites as not-phish.
4. Coercing legitimate users to vote incorrectly.

We can envision two scenarios where an attacker tries to manipulate PhishTank. The *selfish attacker* seeks to protect her own phishing websites by voting down any accusatory report as invalid. Such an attacker shares no empathy with other phishing attackers. The selfish attacker attempts to avoid unwanted attention by just allowing her own websites through (attack type 3 above). The attacker's strong incentive to protect herself even when it causes harm to others is a novel property of PhishTank's voting system.

The *undermining attacker* does not bother with such subtleties. Instead, this attacker seeks to harm the credibility of PhishTank, which is best achieved by combining attacks 1 and 2: submitting URLs for legitimate websites and promptly voting them to be phish. This attacker may also increase the confusion by attempting to create false negatives, voting phishing websites as legitimate.

Coercing legitimate PhishTank users to vote incorrectly (type 4) is certainly feasible, since a public record of each vote is kept. However, coercion attacks are unlikely to be necessary given the ease of registering new user identities and the limited value of user reputation in deciding whether a submission is a phish.

Detecting and defending against these attacks while maintaining an open submission and verification policy is hard. Many of the straightforward countermeasures can be

sidestepped by a smart attacker. We consider a number of countermeasures in turn, demonstrating their inadequacy.

One simple countermeasure is to place an upper limit on the number of actions any user can take. This is unworkable for PhishTank due to its power-law distribution: some legitimate users participate many thousands of times. In any case, an enforced even distribution is easily defeated by a Sybil attack [49], where users register many identities. Given that many phishing attackers use botnets, even strict enforcement of 'one person, one vote' can probably be overcome.

The next obvious countermeasure is to impose voting requirements. For example, a user must have participated 'correctly' n times before weighing their opinion. This is ineffective for PhishTank, though the developers tell us that they do implement this countermeasure. First, since 97% of all submissions are valid, an attacker can quickly boost her reputation by voting for a phish slightly more than n times. Second, a savvy attacker can even minimize her implication of real phishing websites by only voting for rock-phish domains or duplicate URLs. Indeed the highly stylized format for rock-phish URLs makes it easy to automate correct voting at almost any desired scale.

Let us consider the complementary countermeasure. What about ignoring any user with more than n invalid submissions or incorrect votes? A malicious user is unlikely to force through all of his bad submissions and votes. Hence, a large number of deviating actions is a good proxy of misbehavior. Unfortunately, the power-law distribution of user participation causes another problem. Many heavily participating users who do a lot of good also make a lot of mistakes. For instance, the top submitter, *antiphishing*, is also the user with the highest number of invalid submissions, 578.

An improvement is to ban users who are wrong more than x% of the time. Nevertheless, attackers can simply pad their statistics by voting randomly, or by voting for duplicates and rock-phish URLs. Furthermore, many well-intentioned users might be excluded. Ignoring all users where more than 5% of their submissions are invalid would exclude 1 343 users, or 44% of all submitters. Ignoring them would also exclude 8 433 valid submissions, or 5% of all phishing URLs.

Moderators already participate in nearly every vote, so it would not be a stretch to insist that they were the submitter or voted with the majority. We do not know how often they vote incorrectly, but as discussed in Section 9.4.1, we know that even moderators make mistakes. One sign of fallibility is that just over 1% of moderator's submissions were voted down as invalid. Nonetheless, perhaps the best strategy for PhishTank is to use trusted moderators exclusively if they suspect that they are under attack. Given that the 25 moderators already cast 74% of PhishTank's votes, silencing the whole crowd to root out the attackers may sometimes be wise, even if it contradicts principles of open participation.

9.5.2 Lessons for secure crowd-sourcing

We can draw several general lessons about applying the open-participation model to security tools after examining the PhishTank data.

Lesson 1: The distribution of user participation matters. There is a natural tendency for highly skewed distributions, even power laws, in user participation rates. Power law-like distributions have also been observed in the interactions of online communities [170] and blogs [148]. While there may certainly be cases that are not as skewed as PhishTank, security engineers must check the distribution for wide variance when assessing the risk of leveraging user participation.

Skewed distributions can indeed create security problems. First, corruption of a few high-value participants can completely undermine the system. This is not a huge threat for PhishTank since attackers are probably too disorganized to buy off moderators. Nonetheless, the power-law distribution still means that the system could be in trouble if a highly active user stops participating.

Second, because good users can participate extensively, bad users can too. Simple rate-limiting countermeasures do not work here. Bad users may cause significant disruption under cover of a large body of innocuous behavior. Note that we do not take the view that all crowd-based security mechanisms should have balanced user participation. Enthusiastic users should be allowed to participate more, since their enthusiasm drives the success of crowd-based approaches. However, the distribution must be treated as a security consideration.

Lesson 2: Crowd-sourced decisions should be difficult to guess. Any decision that can be reliably guessed can be automated and exploited by an attacker. The underlying accuracy of PhishTank's raw data (97% phish) makes it easy for an attacker to improve her reputation by blindly voting all submissions as phish.

Lesson 3: Do not make users work harder than necessary. Requiring users to vote multiple times for duplicate URLs and rock-phish domains is not only an efficiency issue. It becomes a security liability since it allows an attacker to build up reputation without making a positive contribution.

9.6 Related work

In his book 'The Wisdom of Crowds', Surowiecki argued that under many circumstances the aggregation of a group's opinions can be more accurate than even the most expert individual [152]. He noted that web participation is particularly suited to crowd-based aggregation. Surowiecki listed a number of conditions where crowd-based intelligence may run into trouble: from overly homogeneous opinions to imitative users. We have

highlighted how crowds may be manipulated if the distribution of participation is highly skewed and the correct decision can be reliably guessed.

Recently, user participation has been incorporated into security mechanisms, primarily as a data source rather than performing assessment as is done by PhishTank. Microsoft Internet Explorer and Mozilla Firefox both ask users to report suspicious websites, which are then aggregated to populate blacklists. 'StopBadware' collects reports from users describing malware and disseminates them after administrators have examined the submissions [151]. Herdict is a software tool which collects data from client machines to track malware [78]. Vipul's Razor, an open-source spam-filtering algorithm used by Cloudmark, solicits user spam emails as input [160]. NetTrust is a software application that shares information about the trustworthiness of websites via social networks [32].

Researchers have observed skewed distribution of user activity on the web in contexts other than security. Shirky argued that the influence of blogs (measured by the number of inbound links) naturally exhibited power-law distributions and discussed concerns about the effects of such inequality [149]. Adar *et al.* studied the structure of links between blogs to develop a ranking mechanism [2], while Shi *et al.* found blogs exhibited near-power-law distributions [148] in the number of inbound links. Meanwhile, Zhang *et al.* found power-law distributions in the participation rates of users in online communities [170].

Concerns over the manipulability of user-contributed web content have been raised before, most notably in the case of Wikipedia [46]. The SETI@Home distributed computational project was reported to have experienced widespread cheating [82]. More generally, Albert *et al.* found that networks whose connections between nodes are distributed according to a power law are vulnerable to targeted removal [7].

Countermeasures to voting manipulation where some users' votes are weighed more heavily than others share similarities to research in trust management [26]. Researchers have devised many different metrics which differentiate between good users and bad [137, 99], often for use in reputation systems [83]. More sophisticated trust metrics like these might fare better than the simple countermeasures discussed in Section 9.5.1.

9.7 Conclusion

End-user participation is an increasingly popular resource for carrying out information security tasks. Having examined one such effort to gather and disseminate phishing information, we conclude that while such open approaches are promising, they are currently less effective overall than the more traditional closed methods. Compared to a data feed collected in a conventional manner, PhishTank is less complete and less timely. On the positive side, PhishTank's decisions appear mostly accurate: we identified only a few incorrect decisions, all of which were later reversed. However, we found that inexperienced

users make many mistakes and that users with bad voting records tend to commit the same errors. So the 'wisdom' of crowds sometimes shades into folly.

We also found that user participation varies greatly, raising concerns about the ongoing reliability of PhishTank's decisions due to the risk of manipulation by small numbers of people. We have described how PhishTank can be undermined by a phishing attacker bent on corrupting its classifications, and furthermore how the power-law distribution of user participation simultaneously makes attacks easier to carry out and harder to defend against.

Despite these problems, we do not advocate against leveraging user participation in the design of all security mechanisms. Rather, we believe that the circumstances must be more carefully examined for each application, and furthermore that threat models must address the potential for manipulation.

Chapter 10

Concluding remarks

Computing applications are becoming more decentralized. Consequently, cooperation is increasingly important for achieving many tasks. This matters especially for security. An adversary might compromise several devices in a network, and use them in a combined way that causes more damage than would be possible when abusing the devices individually. Cooperation among defenders may also be required, since no one may control all the components of a network yet the entire network must be protected. Cooperating in defense is further complicated by a lack of trust between components and the potential for conflicting interests. This thesis has examined cooperative attack and defense for a range of distributed network applications.

Chapter 3 presented two categories of attack on wireless sensor networks. In the first attack, an adversary compromises a number of devices to extract the keys loaded onto them. The adversary then selectively swaps the secrets between the devices in order to overwhelm the sensor network with illegitimate communication channels. The attack is noteworthy because of the scaling effect achieved through collusion: compromising 5% of the devices enables an attacker to control 50% of authenticated communication channels. In the second attack, revoked devices can rejoin the network by setting up path keys via colluding intermediaries. This attack can be mitigated by keeping a better record whenever path keys are established, but this comes at a price of increased complexity. The root cause of both attacks is the trade-off made by the designers of key-management schemes to assign a limited number of keys to devices. Such an approach favors minimizing the costs at deployment, while ignoring any increase in maintenance costs. Notably, many attacks made possible by the limited assignment of keys are cooperative in nature (e.g., key-harvesting attacks [54]).

When it comes to network defense, one of the most difficult tasks is agreeing upon a course of action between mutually untrusted parties. In order to revoke a device, the other devices detecting misbehavior must first agree to take action. This thesis has contributed a number of strategies for use in wireless sensor networks, which face the added difficulty of reaching agreement using as lightweight a mechanism as possible. First,

though, we fixed one voting-based strategy by making it resilient to the path-key attacks outlined in Chapter 3. Next, we proposed a reelection protocol in Chapter 4, where devices issue positive votes of support instead of negative votes for removal. We then adapted reelection to create an extremely lightweight 'buddy list' mechanism using hash chains. Buddy lists are also quite versatile, supporting diverse strategies suited to an individual participant's risk appetite. However, each of these voting-based protocols faces significant communication and storage costs, and none cope very well with node mobility and churn. So we developed an even simpler strategy: suicide. A node observing another node behaving badly simply broadcasts a signed message declaring both of them to be dead. Suicide has very low communication and storage overhead, and is well suited for applications where the individual network components value the health of the overall system.

Chapter 5 adapted the suicide and voting strategies for use in a slightly different application: vehicular networks. Researchers in academia and the auto industry are investigating ways to equip cars with wireless radios for sending safety messages in an ad-hoc manner. It is therefore important to develop techniques for ignoring the transmissions of errant devices. To this end, we compared a modified suicide protocol, called Stinger, to an existing blackballing scheme, called LEAVE. Extensive simulations identified a number of trade-offs between the strategies depending on traffic conditions and the accuracy of the detection mechanism. On the whole, we found Stinger to be better suited than LEAVE for the same reasons suicide was preferable to the voting-based alternatives: it is faster, more resource-efficient, and less prone to manipulation.

It is all very well to identify attacks exploiting cooperation and design collaborative decision mechanisms for networking applications, as we have done in Chapters 3–5. The circumstances of new applications demand simulating theoretical attacks and proposing the adoption of suitable countermeasures. But it is also important to study attacks already taking place, along with defensive countermeasures being deployed. This is what we have done in Chapters 6–9. By empirically analyzing phishing attacks on the Internet, we found that the same notions of cooperation in terms of attack and defense are true for attacks occurring today.

The Internet is comprised of many interdependent components, each managed by companies and organizations spanning the globe. Phishing attacks target many of these components, and to the extent that one adversary can launch many fake websites simultaneously, phishing attacks are cooperative. Chapter 6 described how the rock-phish gang cooperates to create a resilient architecture of many domains mapping to many proxies.

The defenders of phishing attacks – ISPs, registrars, banks and the removal companies they employ – stand to gain from cooperation. The localized view of single ISPs and registrars makes it hard to understand some of the more complex phishing scams. The banks are not always aware of the websites impersonating them, even though their competitors

sometimes are.

The primary defensive strategy employed is to identify phishing websites as they appear and initiate take-down procedures to remove the offending material. So we gathered extensive data on the lifetimes of phishing websites as a measure of the effectiveness of the take-down strategy. Chapter 7 found empirical evidence that rock-phish attackers cooperate across the components of attack. They successfully exploit the lack of cooperation between defenders (especially registrars and ISPs), which showed up in the substantially longer lifetimes of rock-phish domains and proxies when compared to ordinary phishing websites.

More generally, we found that the lifetimes of phishing websites correspond to a highly skewed lognormal distribution. This means that while many websites are removed quickly, others remain for much longer. Individual factors can certainly affect website lifetimes, from the competence of the hosting ISP to the vigilance of the bank being targeted. Overall, though, the lognormal distribution reflects the fundamentally uncoordinated defense that takes place when protecting the Internet, and the empirical evidence of its occurrence is a key contribution of this thesis.

Chapter 8 looked more closely at the feeds of phishing URLs maintained by different defenders. We observed that while there is some data sharing in the anti-phishing industry, it is distinctly lacking between competing take-down companies employed by the banks. We found that websites had consistently longer lifetimes when the responsible take-down company was either completely unaware the website existed, or when the company belatedly learned of them. We calculated how much shorter website lifetimes would be if the take-down companies were to share information with each other. Next, we translated the difference in lifetimes to dollars, finding that roughly $250 million annually is put at risk as a result of non-cooperation between the two take-down firms we observed.

Chapter 9 returned to the question of the suitability of voting as a decision mechanism. We examined user participation in PhishTank, which encourages anyone to vote on the accuracy of its phishing feed. While turning over decisions to the 'wisdom of crowds' is intuitively appealing, we found that it concentrates power with the most active participants. Such a power-law distribution of user activity exposes PhishTank to manipulation. We also found evidence that bad decisions reinforce themselves: users who vote 'incorrectly' tend to make the same mistakes. Finally, we compared PhishTank's voting-based system to the unilateral approach taken by the take-down companies. Consistent with the speed of Stinger compared to voting-based LEAVE identified in the vehicular network simulations of Chapter 5, we found verifications in PhishTank to be significantly slower than those performed by the take-down company.

10.1 Future research opportunities

This thesis has examined a number of strategies for attack and defense in different contexts. It has proposed several ways for devices in ad-hoc networks to decide when to remove errant devices, and identified performance and security trade-offs. However, more opportunities for comparison remain.

Not all strategies are suited to all applications. In fact, hybrid strategies may work better – assigning a portion of the devices to adopt suicide while the rest use blackballing, for example. In the case of vehicular networks, cars could be free to dynamically choose between LEAVE and Stinger as their eviction strategy depending upon traffic conditions. For many applications, system characteristics such as network structure, interaction length, and risk tolerance may vary greatly. Strategies must be tested for their ability to handle such dynamics.

One promising approach could draw inspiration from Axelrod's tournaments, which compared the suitability of different strategies for iterated prisoner's dilemma games [23]. Axelrod found that a simple tit-for-tat strategy worked best. Similarly, strategies for cooperation in computer networks should be rigorously compared in network simulations.

Future research could also model the strategies used in phishing attack and defense, which might help identify how best to allocate defensive resources. At present, defenders indiscriminately react to phishing websites as they are discovered. Investing more effort into back-end controls, or concentrating efforts on particular groups of attackers, could prove more effective.

Beyond phishing, the more general problem of electronic crime must be better understood. Attacks are increasingly commoditized, driven by criminal gangs but also accessible to individuals. Consequently, the proliferation of attacks has hindered distinguishing between the instances of attack and number of attackers. This thesis has already identified that one group, the rock-phish gang, accounts for half of all phishing attacks. Linking attacks together matters because defenders do not always see the relationship, and identifying them can focus countermeasures.

There is growing evidence of connections between different classes of attacks – phishing, malware, botnets, and denial-of-service attacks. Much could be learned by empirically examining these attacks and their role in the underground economy. Future research is needed to explore linkages between attacks, to estimate the value of illicit transactions, and to identify techniques for disrupting the community's efforts. Sound macroeconomic analysis of the underground economy could be achieved by studying the supply and demand effects of selling online credentials, for example.

Reducing the amount of wickedness traversing the Internet is a fundamental problem, and better measurement is likely an essential part of the solution. As identified by the empirical analysis in this thesis, many thousands of compromised web servers host fraudulent

phishing websites. Additionally, millions of corrupted end-user machines are enslaved as botnets sending spam, launching denial-of-service attacks and hosting dubious websites. Without consistent measurements of where this wickedness emanates from, ISPs have little incentive to take precautions or clean up infected machines.

This thesis has found great variation in the take-down speed for different banks and ISPs – many sites are removed within a few hours, but others remain for many weeks. Future research efforts could develop a global league table of ISP security, using data on phishing attacks, botnet populations and scanning activity. It is equally crucial to develop better measurements of defense – tracking the adoption of ISP security practices, for instance. Collaboration with statisticians is likely to be required since attack data is often messy: reliable metrics can be hindered by unbalanced sampling of attacks per target, and most measurements exhibit highly skewed distributions.

Objective measurements that identify the best and worst performers could raise overall security levels. They could quantify the variation in security across the Internet. Finally, a league table could serve as a basis for policy to reward compliant ISPs and punish irresponsible ones.

Many network security problems conflict with the standard (Byzantine fault tolerance) view of system reliability: attackers may collude, invalidating the random failure assumption; the targeted removal of a few critical devices can trigger cascading failure. Chapter 3 described an attack where a small number of colluding devices can overwhelm the majority of communications channels. The latter part of this thesis has identified cases where the structure of distributed networks has created long-tailed distributions. The average lifetimes of phishing sites follow a lognormal distribution, for example, and the rates of end-user participation on PhishTank follow a power-law distribution. Such skewed distributions make systems brittle: a few components can greatly impact a system's overall security.

One open research question is why these skewed distributions keep appearing. It seems that the lognormal distribution for phishing website lifetimes arises from dependencies between the many components responsible for removing malicious content from the Internet. Future research could develop a more formalized model to explain why the skewed distribution emerges. Such modeling could then be used to help select countermeasures which minimize lifetimes.

In any case, new methods are needed to adapt to the reality of unbalanced participation. Threat models must accommodate the compromise of highly influential components, and applications should be designed to be robust against exceptional failures.

List of acronyms

APWG	Anti-Phishing Working Group
BGP	Border Gateway Protocol
CA	Certification Authority
CDF	Cumulative Distribution Function
CERT/CC	Computer Emergency Response Team/Coordination Center
DDoS	Distributed Denial of Service
DNS	Domain Name Service
DNSSEC	Domain Name Service Security Extensions
DSRC	Dedicated Short Range Communications
HMAC	Hash Message Authentication Code
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISAC	Information Sharing and Analysis Center
ISP	Internet Service Provider
LEAVE	Local Eviction of Attackers by Voting Evaluators
MAC	Message Authentication Code
PIKE	Peer Intermediaries for Key Establishment
PKI	Public Key Infrastructure

SSH Secure Shell

TCP Transmission Control Protocol

URL Uniform Resource Locator

WSN Wireless Sensor Network

Bibliography

[1] A. Acquisti, A. Friedman, and R. Telang, 'Is there a cost to privacy breaches? An event study', in *5th Workshop on the Economics of Information Security (WEIS)*, 2006.

[2] E. Adar, L. Zhang, L. Adamic, and R. Lukose, 'Implicit structure and the dynamics of blogspace', in *Workshop on the Weblogging Ecosystem, 13th International World Wide Web Conference (WWW)*, 2004.

[3] Advanced Safety Vehicle Program, `http://www.ahsra.or.jp/demo2000/eng/demo_e/ahs_e7/iguchi/iguchi.html`.

[4] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou, 'Selfish behavior and stability of the Internet: a game-theoretic analysis of TCP', *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 117–130, 2002.

[5] G. Akerlof, 'The market for "lemons": quality uncertainty and the market mechanism', in *The Quarterly Journal of Economics*, vol. 84, no. 3, pp. 488–500, 1970.

[6] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, 'A survey on sensor networks', *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[7] R. Albert, H. Jeong, A. Barabási, 'Error and attack tolerance of complex networks', *Nature*, vol. 406, pp. 378–382, Jul. 2000.

[8] L. Anderson and C. Holt, 'Information cascades in the laboratory', *American Economic Review*, vol. 87, no. 5, pp. 847–862, 1995.

[9] R. Anderson, 'Closing the phishing hole: fraud, risk and non-banks', in *Federal Reserve Bank of Kansas City Payments Conference*, 2007.

[10] R. Anderson, 'The eternity service', in *First International Conference on the Theory and Applications of Cryptology (PRAGOCRYPT)*, 1996.

[11] R. Anderson, 'The initial costs and maintenance costs of protocols', in *13th International Workshop on Security Protocols*, 2005, Springer Lecture Notes in Computer Science (LNCS), vol. 4631, pp. 333–343, 2007.

[12] R. Anderson, 'Why information security is hard – an economic perspective', in *17th Annual Computer Security Applications Conference (ACSAC)*, 2001, pp. 358–365.

[13] R. Anderson, F. Bergadano, B. Crispo, J. H. Lee, C. Manifavas, and R. Needham, 'A new family of authentication protocols', *ACM SIGOPS Operating Systems Review*, vol. 32, no. 4, pp. 9–20, 1998.

[14] R. Anderson, H. Chan, and A. Perrig, 'Key infection: smart trust for smart dust', in *IEEE International Conference on Network Protocols (ICNP)*, 2004, pp. 206–215.

[15] R. Anderson and T. Moore, 'The economics of information security', *Science*, vol. 314, no. 5799, pp. 610–613, 2006.

[16] E. Anshelevich, A. Dasgupta, É. Tardos and T. Wexler, 'Near-optimal network design with selfish agents', in *35th ACM Symposium on Theory of Computing (STOC)*, 2003, pp. 511–520.

[17] Anti-Phishing Working Group, `http://www.antiphishing.org`.

[18] APACS, 'Card fraud losses continue to fall', 14 Mar. 2007, `http://www.apacs.org.uk/media_centre/press/07_14_03.html`.

[19] A. Arora, R. Krishnan, R. Telang and Y. Yang, 'An empirical analysis of vendor response to disclosure policy', in *4th Workshop on the Economics of Information Security (WEIS)*, 2005.

[20] Artists Against 419, `http://www.aa419.org`.

[21] J. Aspnes, K. Chang, and A. Yampolskiy, 'Inoculation strategies for victims of viruses and the sum-of-squares partition problem', in *16th ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 43–52.

[22] ASTM E2213-03, Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems – 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[23] R. Axelrod, *The Evolution of Cooperation*, New York: Basic Books, 1984.

[24] S. Baker, 'Gambling sites, this is a holdup', *Business Week*, 9 Aug. 2004, `http://www.businessweek.com/magazine/content/04_32/b3895106_mz063.htm`.

[25] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede, 'Low-cost elliptic curve cryptography for wireless sensor networks', in *3rd European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, Springer LNCS, vol. 4357, pp. 6–17, 2006.

[26] M. Blaze, J. Feigenbaum, and J. Lacy, 'Decentralized trust management', in *IEEE Symposium on Security and Privacy (S&P)*, 1996, pp. 164–173.

[27] Bluetooth SIG, 'Specification of the Bluetooth system', 2001.

[28] S. Brands and D. Chaum, 'Distance-bounding protocols (extended abstract)'. in *Advances in Cryptology (EUROCRYPT)*, Springer LNCS, vol. 765, pp. 344–359, 1993.

[29] S. Buchegger, J.-Y. Le Boudec, 'Performance analysis of the CONFIDANT protocol', in *3rd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002, pp. 26–236.

[30] L. Buttyán, and J. P. Hubaux, *Security and Cooperation in Wireless Networks*, Cambridge: Cambridge University Press, 2008.

[31] California State Senate, 'Assembly bill 700', 2002, `http://info.sen.ca.gov/pub/01-02/bill/asm/ab_0651-0700/ab_700_bill_20020929_chaptered.pdf`.

[32] L. Jean Camp, 'Reliable, usable signaling to defeat masquerade attacks', in *5th Workshop on the Economics of Information Security (WEIS)*, 2006.

[33] S. Čapkun, L. Buttyán, and J. P. Hubaux, 'SECTOR: secure tracking of node encounters in multi-hop wireless networks', in *1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2003, pp. 21–32.

[34] S. Čapkun and J. P. Hubaux, 'Secure positioning in wireless networks', *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 221–232, 2006.

[35] Castle Cops, 'Phishing incident reporting & termination', 2008, `http://www.castlecops.com/pirt`.

[36] H. Chan, V. D. Gligor, A. Perrig, and G. Muralidharan, 'On the distribution and revocation of cryptographic keys in sensor networks', *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 3, pp. 233–247, 2005.

[37] H. Chan and A. Perrig, 'PIKE: peer intermediaries for key establishment in sensor networks', in *24th IEEE Conference on Computer Communications (INFOCOM)*, 2005, pp. 524–535.

[38] H. Chan, A. Perrig, and D. X. Song, 'Random key predistribution schemes for sensor networks', in *IEEE Symposium on Security and Privacy (S&P)*, 2003, pp. 197–213.

[39] I. A. Chitu, 'Google redirect notice', 16 Feb. 2007, `http://googlesystem.blogspot.com/2007/02/google-redirect-notice.html`.

[40] Communications for eSafety, `http://www.comesafety.org`.

[41] C. Crépeau and C. Davis, 'A certificate revocation scheme for wireless ad hoc networks', in *1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2003, pp. 54–61.

[42] B. Crow, I. Widjaja, L. Kim, and P. Sakai, 'IEEE 802.11 wireless local area networks', *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116–126, 1997.

[43] R. Dacey, 'Information security: progress made, but challenges remain to protect federal systems and the nation's critical infrastructures', *GAO-03-564T*, pp. 1–75, US General Accounting Office (GAO), Apr 2003.

[44] G. Danezis and R. Anderson, 'The economics of resisting censorship', *IEEE Security & Privacy*, vol. 3, no. 1, pp. 45–50, 2005.

[45] C. Dellarocas, 'Analyzing the economic efficiency of eBay-like online reputation mechanisms', in *3rd ACM Conference on Electronic Commerce*, 2001, pp. 171–179.

[46] P. Denning, J. Horning, D. Parnas, and L. Weinstein, 'Wikipedia risks', *Communications of the ACM*, vol. 48, no. 12, p. 152, 2005.

[47] W. Diffie, 'The first ten years of public-key cryptography', *Proceedings of the IEEE*, vol. 76, no. 5, pp. 560–577, 1988.

[48] R. Di Pietro, L. V. Mancini, and A. Mei, 'Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks', *Wireless Networks*, vol. 12, no. 6, pp. 709–721, 2006.

[49] J. R. Douceur, 'The Sybil attack', in *1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Springer LNCS, vol. 2429, pp. 251–260, 2002.

[50] C. Drake, J. Oliver, and E. Koontz, 'Anatomy of a phishing email', in *1st Conference on Email and Anti-Spam (CEAS)*, 2004.

[51] W. Du, J. Deng, Y. Han, S. Chen and P. Varshney, 'A key management scheme for wireless sensor networks using deployment knowledge', in *23rd IEEE Conference on Computer Communications (INFOCOM)*, 2004.

[52] W. Du, J. Deng, Y. Han, and P. Varshney, 'A pairwise key pre-distribution scheme for wireless sensor networks', in *10th ACM Conference on Computer and Communications Security (CCS)*, 2003, pp. 42–51.

[53] C. Dwork and M. Naor, 'Pricing via processing or combatting junk mail', in *Advances in Cryptology (CRYPTO)*, 1992, Springer LNCS, vol. 740, pp. 139–147, 1993.

[54] L. Eschenauer and V. D. Gligor, 'A key-management scheme for distributed sensor networks', in *9th ACM Conference on Computer and Communications Security (CCS)*, 2002, pp. 41–47.

[55] A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, S. Shenker, 'On a network creation game', in *22nd ACM Symposium on Principles of Distributed Computing (PODC)*, 2003, pp 347–351.

[56] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker, 'A BGP-based mechanism for lowest-cost routing', in *21st ACM ACM Symposium on Principles of Distributed Computing (PODC)*, 2002, pp. 173–182.

[57] M. Feldman, K. Lai, I. Stoica, and J. Chuang, 'Robust incentive techniques for peer-to-peer networks', in *5th ACM Conference on Electronic Commerce*, 2004, pp. 102–111.

[58] D. Florêncio and C. Herley, 'Evaluating a trial deployment of password re-use for phishing prevention', in *Anti-Phishing Working Group eCrime Researcher's Summit (APWG eCrime)*, 2007, pp. 26–36.

[59] D. Fudenberg and J. Tirole, *Game Theory*, Cambridge: MIT Press, 1991.

[60] E. Gal-Or and A. Ghose, 'The economic incentives for sharing security information', *Information Systems Research*, vol. 16, no. 2, pp. 186–208, 2005.

[61] A. Galetsas, 'Statistical data on network security', *European Commission, DG Information Society and Media*, 2007, `ftp://ftp.cordis.europa.eu/pub/ist/docs/trust-security/statistics-network-security-050307_en.pdf`.

[62] S. Ganeriwal and M. B. Srivastava, 'Reputation-based framework for high integrity sensor networks', in *2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2004, pp. 66–77.

[63] M. Gardner, 'Mathematical games: the fantastic combinations of John Conway's new solitaire game "Life"', *Scientific American*, vol. 223, pp.120–123, Oct. 1970.

[64] Gartner Inc. , 'Gartner says number of phishing e-mails sent to U. S. adults nearly doubles in just two years', 9 Nov. 2006, `http://www.gartner.com/it/page.jsp?id=498245`.

[65] G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar, 'State of the art in ultra-low power public key cryptography for wireless sensor networks', in *3rd IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2005, pp. 146–150.

[66] G. Gaubatz, J.-P. Kaps, and B. Sunar, 'Public key cryptography in sensor networks – revisited', in *1st European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, 2004, Springer LNCS, vol. 3313, pp. 2–18, 2005.

[67] P. Golle, D. Greene, and J. Staddon, 'Detecting and correcting malicious data in VANETs', in *1st ACM International Workshop on Vehicular Ad hoc Networks (VANET)*, 2004, pp. 29–37.

[68] S. Gordon and R. Ford, 'When worlds collide: information sharing for the security and anti-virus communities', IBM research paper, 1999.

[69] L. Gordon, M. Loeb and W. Lucyshyn, 'Sharing information on computer systems security: an economic analysis', *Journal of Accounting and Public Policy*, vol. 22, no. 6, pp. 461–485, 2003.

[70] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. 'Comparing elliptic curve cryptography and RSA on 8-bit CPUs', in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Springer LNCS, vol. 3156, pp.119–132, 2004.

[71] M. Halldórsson, J. Halpern, L. Li, and V. Mirrokni, 'On spectrum sharing games', in *23rd ACM ACM Symposium on Principles of Distributed Computing (PODC)*, 2004, pp. 107–114.

[72] G. Hancke and M. Kuhn, 'An RFID distance bounding protocol', in *IEEE International Conference on Security and Privacy in Communications Networks (SecureComm)*, 2005, pp. 67–73.

[73] G. Hardin, 'The tragedy of the commons', *Science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[74] J. Hirshleifer, 'From weakest-link to best-shot: the voluntary provision of public goods', *Public Choice*, vol. 41, pp. 371–386, 1983.

[75] Honeynet Project and Research Alliance, 'Know your enemy: fast-flux service networks, an ever changing enemy', 2007, `http://www.honeynet.org/papers/ff/fast-flux.pdf`.

[76] R. Housley, W. Polk, W. Ford, and D. Solo, 'Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile', Internet Engineering Task Force RFC 3280, 2002.

[77] Y. C. Hu, A. Perrig, and D. B. Johnson, 'Packet leashes: A defense against wormhole attacks in wireless networks', in *22nd IEEE Conference on Computer Communications (INFOCOM)*, 2003.

[78] T. Hwang, 'Herdict: a distributed model for threats online', *Elsevier Network Security*, pp. 15–18, Aug. 2007.

[79] IEEE P1609.2 Version 1, Standard for Wireless Access in Vehicular Environments – Security Services for Applications and Management Messages (in development).

[80] J. Ioannidis and S. M. Bellovin, 'Implementing pushback: router-based defense against DDoS attacks', in *8th Network and Distributed System Security Symposium (NDSS)*, 2002.

[81] M. Jakobsson and S. Myers, Eds., *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*, New York: Wiley, 2006.

[82] L. Kahney, 'Cheaters bow to peer pressure', *Wired*, 15 Feb. 2001, `http://www.wired.com/news/technology/0,1282,41838,00.html`.

[83] S. Kamvar, M. Schlosser, and H. Garcia-Molina, 'The EigenTrust algorithm for reputation management in P2P networks', in *12th International World Wide Web Conference (WWW)*, 2003, pp. 640–651.

[84] K. Kannan and R. Telang, 'Market for software vulnerabilities? Think again', *Management Science*, vol. 51, no. 5, pp. 726–740, 2005.

[85] M. Katz and C. Shapiro, 'Network externalities, competition, and compatibility', *The American Economic Review*, vol. 75, no. 3, pp. 424–440, Jun. 1985.

[86] D. Kleinbard, 'More sites hacked in wake of Yahoo!', *CNN Money*, 8 Feb. 2000, `http://money.cnn.com/2000/02/08/technology/yahoo/`.

[87] J. Kong, H. Luo, K. Xu, D. Gu, M. Gerla, and S. Lu, 'Adaptive security for multilevel ad hoc networks', *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 533–547, 2002.

[88] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, 'Providing robust and ubiquitous security support for mobile ad hoc networks', in *IEEE International Conference on Network Protocols (ICNP)*, 2001, pp. 251–260.

[89] E. Koutsoupias and C. Papadimitriou, 'Worst-case equilibria', in *16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, Springer LNCS, vol. 1563, pp. 387–396, 1999.

[90] B. Krebs, 'Defcon speakers team up to fight "queen bots"', *Washington Post*, 9 Aug. 2006, `http://blog.washingtonpost.com/securityfix/2006/08/defcon_speakers_team_up_to_fig.html`.

[91] H. Kunreuther and G. Heal, 'Interdependent security', *Journal of Risk and Uncertainty*, vol. 26, nos. 2–3, pp. 231–249, Mar.–May 2003.

[92] L. Lamport, 'Constructing digital signatures from a one-way function', SRI Computer Science Laboratory Technical Report SRI-CSL-98, 1979.

[93] L. Lamport, R. Shostak, and M. Pease, 'The Byzantine generals problem', *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[94] E. Larkin, 'Online thugs assault sites that specialize in security help', *PC World*, 11 Sep. 2007, `http://www.pcworld.com/businesscenter/article/137084/online_thugs_assault_sites_that_specialize_in_security_help_.html`.

[95] B. Laurie and R. Clayton, '"Proof-of-work" proves not to work', in *3rd Workshop on the Economics of Information Security (WEIS)*, 2004.

[96] M. Lee, R. Newman, H. Latchman, S. Katar, and L. Yonge, 'HomePlug 1.0 powerline communication LANs – protocol description and performance results', *International Journal of Communication Systems*, vol. 16, no. 5, pp. 447–473, 2003.

[97] T. Leinmüller, C. Maihöfer, E. Schoch, and F. Kargl, 'Improved security in geographic ad hoc routing through autonomous position verification', in *3rd ACM International Workshop on Vehicular Ad hoc Networks (VANET)*, 2006, pp. 57–66.

[98] R. Lemos, 'Anti-virus groups fight over Crossover sharing', *The Register*, 6 Mar. 2006, `http://www.theregister.co.uk/2006/03/06/crossover_ruling_angers_industry/`.

[99] R. Levien, 'Attack resistant trust metrics', PhD thesis (draft), University of California at Berkeley, 2004.

[100] D. Liu, 'The economics of proof-of-work', *I/S Journal of Law and Policy for the Information Society*, vol. 3, no. 2, 2007.

[101] D. Liu and P. Ning, 'Establishing pairwise keys in distributed sensor networks', in *10th ACM Conference on Computer and Communications Security (CCS)*, 2003, pp. 52–61.

[102] D. Liu and P. Ning, 'Location-based pairwise key establishments for static sensor networks', in *1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2003, pp. 72–82.

[103] D. Liu, P. Ning, and W. Du, 'Detecting malicious beacon nodes for secure location discovery in wireless sensor networks', in *25th International Conference on Distributed Computing Systems (ICDCS)*, 2005, pp. 609–619.

[104] S. Marti, T. J. Giuli, K. Lai, and M. Baker, 'Mitigating routing misbehavior in mobile ad hoc networks', in *6th ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 2000, pp. 255–265.

[105] F. Massey, 'The Kolmogorov-Smirnov test for goodness of fit', *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, Mar. 1951.

[106] R. McMillan, "'Rock Phish" blamed for surge in phishing', *InfoWorld*, 12 Dec. 2006, `http://www.infoworld.com/article/06/12/12/HNrockphish_1.html`.

[107] R. C. Merkle, 'A certified digital signature', in *Advances in Cryptology (CRYPTO)*, Springer LNCS, vol. 435, pp. 218–238, 1989.

[108] R. C. Merkle, 'Protocols for public-key cryptosystems', in *IEEE Symposium on Research in Security and Privacy*, 1980, pp. 122–134.

[109] P. Michiardi and R. Molva, 'Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks', in *Advanced Communications and Multimedia Security, IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security*, Vol. 228, IFIP Conference Proceedings, Kluwer: Deventer, The Netherlands, 2002, pp.107–121.

[110] Microsoft Inc., 'Phishing filter: help protect yourself from online scams', 28 Oct. 2006, `http://www.microsoft.com/athome/security/online/phishing_filter.mspx`.

[111] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, 'SPS: a simple filtering algorithm to thwart phishing attacks', in *Asian Internet Engineering Conference (AINTEC)*, 2005.

[112] E. Montelbano, 'Mozilla: Firefox antiphishing tool better than IE 7', *InfoWorld*, 15 Nov. 2006, `http://www.infoworld.com/article/06/11/15/HNmozillaphishing_1.html`.

[113] T. Moore, 'A collusion attack on random pairwise key predistribution schemes for distributed sensor networks', in *IEEE International Workshop on Pervasive Computing and Communications Security (PerSec)*, 2006, pp. 251–255.

[114] T. Moore, 'Countering hidden-action attacks on networked systems', in *4th Workshop on the Economics of Information Security (WEIS)*, 2005.

[115] T. Moore and R. Clayton, 'Evaluating the wisdom of crowds in assessing phishing websites', in *12th International Financial Cryptography and Data Security Conference (FC)*, Springer LNCS (to appear), 2008.

[116] T. Moore and R. Clayton, 'Examining the impact of website take-down on phishing', in *Anti-Phishing Working Group eCrime Researcher's Summit (APWG eCrime)*, 2007, pp. 1–13.

[117] T. Moore and R. Clayton, 'The consequence of non-cooperation in the fight against phishing', submitted for publication, 2008.

[118] T. Moore and J. Clulow, 'Secure path-key revocation for symmetric key pre-distribution schemes in sensor networks', in *New Approaches for Security, Privacy and Trust in Complex Environments, Proceedings of the IFIP TC 11 22nd International Information Security Conference (SEC 2007)*, Vol. 232, IFIP, H. Venter, M. Eloff, L. Labuschagne, J. Eloff, and R. von Solms, Eds., Heidelberg: Springer, 2007, pp. 157-168.

[119] T. Moore, J. Clulow, S. Nagaraja, and R. Anderson, 'New strategies for revocation in ad-hoc networks', in *4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, Springer LNCS, vol. 4572, pp. 232–246, 2007.

[120] T. Moore, M. Raya, J. Clulow, P. Papadimitratos, R. Anderson, and J.-P. Hubaux, 'Fast exclusion of errant devices from vehicular networks', in *5th IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2008 (to appear).

[121] Mozilla Corp., 'Phishing protection', 2006, `http://www.mozilla.com/en-US/firefox/phishing-protection/`.

[122] R. Mullen, 'The lognormal distribution of software failure rates: origin and evidence', in *9th IEEE International Symposium on Software Reliability Engineering*, 1998, pp. 134–142.

[123] M. Newman 'Power laws, pareto distributions and Zipf's law', *Contemporary Physics*, vol. 46, no. 5, pp. 323–351, 2005.

[124] J. Newsome, E. Shi, D. X. Song, and A. Perrig, 'The Sybil attack in sensor networks: analysis and defenses', in *ACM Information Processing and Sensor Networks (IPSN)*, 2004, pp. 259–268.

[125] Network Simulator, `http://www.isi.edu/nsnam/ns/`.

[126] N. Nisan and A. Ronen, 'Algorithmic mechanism design (extended abstract)', in *31st ACM Symposium on Theory of Computing (STOC)*, 1999, pp. 129–140.

[127] P. Ohm, 'The myth of the superuser: fear, risk, and harm online', University of Colorado Law Legal Studies Research Paper No. 07-14, May 2007.

[128] OpenDNS, 'OpenDNS shares April 2007 PhishTank statistics', 1 May 2007, `http://www.opendns.com/about/press_release.php?id=14`.

[129] A. Ozment and S. Schechter, 'Bootstrapping the adoption of Internet security protocols', in *5th Workshop on the Economics of Information Security (WEIS)*, 2006.

[130] Y. Pan and X. Ding, 'Anomaly based web phishing page detection', in *22nd Annual Computer Security Applications Conference (ACSAC)*, 2006, pp. 381–392.

[131] P. Papadimitratos, V. Gligor, and J.-P. Hubaux, 'Securing vehicular communications – assumptions, requirements, and principles', in *Workshop on Embedded Security in Cars (escar)*, 2006.

[132] B. Parno and A. Perrig, 'Challenges in securing vehicular networks', in *ACM Workshop on Hot Topics in Networking (HotNets)*, 2005.

[133] B. Parno, A. Perrig, and V. D. Gligor, 'Distributed detection of node replication attacks in sensor networks', in *IEEE Symposium on Security and Privacy (S&P)*, 2005, pp. 49–63.

[134] PhishTank, `http://www.phishtank.com`.

[135] M. Raya and J.-P. Hubaux, 'The security of vehicular ad hoc networks', in *3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2005, pp. 11–21.

[136] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, and J.-P. Hubaux, 'Eviction of misbehaving and faulty nodes in vehicular networks', *IEEE Journal on Selected Areas in Communication*, vol. 25, no. 8, pp. 1557–68, 2007.

[137] M. Reiter and S. Stubblebine, 'Toward acceptable metrics of authentication', in *IEEE Symposium on Security and Privacy (S&P)*, 1997, pp. 10–20.

[138] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell, 'Stronger password authentication using browser extensions', in *14th USENIX Security Symposium*, 2005, p. 2.

[139] T. Roughgarden and É. Tardos, 'How bad is selfish routing?', *Journal of the ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[140] J. Quarterman, 'PhishScope: tracking phish server clusters', *Digital Forensics Practice*, vol. 1, no. 2, pp. 103–114, 2006.

[141] A. Saha and D. Johnson, 'Modeling mobility for vehicular ad-hoc networks', in *1st ACM International Workshop on Vehicular Ad hoc Networks (VANET)*, 2004, pp. 91–92.

[142] S. Schechter, R. Dhamija, A. Ozment and I. Fischer, 'The emperor's new security indicators: an evaluation of website authentication and the effect of role playing on usability studies', in *IEEE Symposium on Security and Privacy (S&P)*, 2007, pp. 51–65.

[143] Security Focus Inc., 'Bugtraq mailing list', `http://www.securityfocus.com/archive/1`.

[144] Security of Vehicular Networks@EPFL, http://ivc.epfl.ch.

[145] A. Serjantov and R. Anderson, 'On dealing with adversaries fairly', in *3rd Workshop on the Economics of Information Security (WEIS)*, 2004.

[146] S. Seys and B. Preneel, 'Power consumption evaluation of efficient digital signature schemes for low power devices', in *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, 2005, pp. 79–86.

[147] A. Shamir, 'How to share a secret', *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[148] X. Shi, B. Tseng, and L. Adamic, 'Looking at the blogosphere topology through different lenses', in *International Conference on Weblogs and Social Media*, 2007.

[149] C. Shirky, 'Power laws, weblogs, and inequality', 2003, http://www.shirky.com/writings/powerlaw_weblog.html.

[150] J. Shneidman, D. Parkes and L. Massoulié, 'Faithfulness in internet algorithms', in *ACM SIGCOMM Workshop on Practice and theory of Incentives in Networked Systems (PINS)*, 2004, pp. 220–227.

[151] Stop Badware, http://www.stopbadware.org.

[152] J. Surowiecki, *The Wisdom of Crowds: Why the Many are Smarter than the Few*, New York: Doubleday, 2004.

[153] P. Syverson, 'A different look at secure distributed computation', in *10th IEEE Computer Security Foundations Workshop (CSFW)*, 1997, pp. 109–115.

[154] R. Thomas and J. Martin, 'The underground economy: priceless', *USENIX ;login*, vol. 31, no. 6, pp. 7–16, Dec. 2006.

[155] F. Tobagi and L. Kleinrock, 'Packet switching in radio channels: part II – the hidden terminal problem in carrier sense multiple access and the busy-tone solution', *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1417–1433, Dec. 1975.

[156] L. Uhsadel, A. Poschmann, and C. Paar, 'Enabling full-size public-key algorithms on 8-bit sensor nodes', in *4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)*, Springer LNCS, vol. 4572, pp. 73–86, 2007.

[157] H. Varian, 'System reliability and free riding', in *Economics of Information Security*, Vol. 12, Advances in Information Security, L. J. Camp, S. Lewis, Eds., Boston: Kluwer Academic Publishers, 2004, pp. 1–15.

[158] Vehicle Safety Communications Project, http://www-nrd.nhtsa.dot.gov/pdf/nrd-12/CAMP3/pages/VSCC.htm.

[159] J. Venn, 'On the diagrammatic and mechanical representation of propositions and reasonings', *Dublin Philosophical Magazine and Journal of Science*, vol. 9, no. 59, pp. 1–18, 1880.

[160] Vipul's Razor, `http://razor.sourceforge.net`.

[161] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn and P. Kruus, 'TinyPK: securing sensor networks with public key technology', in *2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2004, pp. 59–64.

[162] R. Weaver and M. Collins, 'Fishing for phishes: applying capture-recapture to phishing', in *Anti-Phishing Working Group eCrime Researcher's Summit (APWG eCrime)*, 2007, pp. 14–25.

[163] Webalizer, `http://www.mrunix.net/webalizer/`.

[164] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min and X. Deng, 'Detection of phishing webpages based on visual similarity', in *14th International World Wide Web Conference (WWW)*, 2005, pp. 1060–1061.

[165] P. Wohlmacher, 'Digital certificates: a survey of revocation methods', in *ACM Workshops on Multimedia*, 2000, pp. 111–114.

[166] R. Wright, P. Lincoln, and J. Millen, 'Efficient fault-tolerant certificate revocation', in *7th ACM Conference on Computer and Communications Security (CCS)*, 2000, pp. 19–24.

[167] M. Wu, R. Miller and S. Garfinkel, 'Do security toolbars actually prevent phishing attacks?', in *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'06)*, 2006, pp. 601–610.

[168] W. Ye, J. Heidemann and D. Estrin, 'An energy-efficient MAC protocol for wireless sensor networks', in *21st IEEE Conference on Computer Communications (INFOCOM)*, 2002, pp. 1567–1576.

[169] M. E. Zarki, S. Mehrotra, G. Tsudik, and N. Venkatasubramanian, 'Security issues in a future vehicular network', in *European Wireless*, 2002.

[170] J. Zhang, M. Ackerman, and L. Adamic, Expertise networks in online communities: structure and algorithms, in *16th International World Wide Web Conference (WWW)*, 2007, pp. 221–230.

[171] Y. Zhang, S. Egelman, L. F. Cranor and J. Hong, 'Phinding phish: evaluating anti-phishing tools', in *14th Network and Distributed System Security Symposium (NDSS)*, 2007.

[172] P. Zheng, 'Tradeoffs in certificate revocation schemes', *SIGCOMM Computing Communication Review*, vol. 33, no. 2, pp. 103–112, 2003.

[173] L. Zhou and Z. Haas, 'Securing ad hoc networks', *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[174] L. Zhou, F. Schneider, and R. van Renesse, 'COCA: A secure distributed online certification authority', *ACM Transactions on Computer Systems*, vol. 20, no. 4, 2002, pp. 329–368.

[175] S. Zhu, S. Setia, and S. Jajodia, 'LEAP: efficient security mechanisms for large-scale distributed sensor networks', in *10th ACM Conference on Computer and Communications Security (CCS)*, 2003, pp. 62–72.

[176] S. Zhu, S. Xu, S. Setia, and S. Jajodia, 'Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach', in *IEEE International Conference on Network Protocols (ICNP)*, 2003, pp. 326–335.