Nation-State Attackers and their Effects on Computer Security

by

Andrew Springall

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2018

Doctoral Committee:

Professor J. Alex Halderman, Chair
Research Professor Peter Honeyman
Professor Atul Prakash
Assistant Professor Florian Schaub

Andrew Springall
aaspring@umich.edu
ORCID: 0000-0001-5999-9881

2018

Acknowledgments

This dissertation, the research inside and outside of it, and so many more things would not be possible without the support of an innumerable number of people. I first want to thank my advisor, J. Alex Halderman for everything. The guidance, encouragement, advice, and freedom he's given me over the last five years is nothing short of miraculous. I also want to thank Peter Honeyman for his steadfast viewpoints and willingness to openly challenge my perspective over the years.

I'd also like to thank my labmates Zakir Durumeric, Eric Wustrow, James Kasten, David Adrian, Ariana Mirian, Benjamin VanderSloot, Matthew Bernhard, Allison McDonald, and Chris Dzombak for their help, willingness to put up with me, and the wonderful discussions.

To Mom, the prayers and pleadings to finish high school, and later college, have gotten me here and although I dismissed them at the time, they were undoubtedly needed and welcome. To Dad…you know. To the Marines, Soldiers, Sailors, and Airmen from Parris Island, California, Maryland, Iraq, and Afghanistan, I can credit you for the mindset and insanity needed to get me this far. To Goose, Katherine, Ariana, and Cecylia, you've helped me keep what little of my sanity I have left throughout all this.

Throughout the good and the bad, you've all been willing to help, mentor, counsel, and inspire me to do things I honestly never even imagined were possible. So to everyone listed above and everyone else, whether they know it or not: Thank You for everything you've done for me.

TABLE OF CONTENTS

Acknowledgments . . . . . **i**

List of Figures . . . . . **v**

List of Tables . . . . . **viii**

Abstract . . . . . **x**

Chapter

1 Introduction **1**

1.1 Computer Security and Nation-State Attackers 2
1.2 Overview 4

2 Nation-State Attacker Model **6**

2.1 Lexicon 7
2.1.1 Actors 8
2.1.2 Actions 9
2.1.3 Descriptive 10
2.1.4 Usage 11
2.2 Characteristics of a Nation State Attacker 13
2.2.1 Sovereignty 13
2.2.2 Access 14
2.2.3 Money 15
2.3 Advantages of a Nation State Attacker 17
2.3.1 Near Superset Attacker 17
2.3.2 Specialization 19
2.3.3 Non-Symmetric Defeats 21
2.3.4 Distant Return Horizons 24
2.4 Constraints of a Nation State Attacker 25
2.4.1 Scale 25
2.4.2 Human Capital 28
2.4.3 Oversight 31
2.4.4 Required Hard Victims 33
2.5 Differences between Nation State Attackers 35
2.5.1 Treatment of Domestic Victims 35
2.5.2 Acceptability of Public Attribution 36
2.5.3 Delegating Operations 37

2.6 Accounting for NS-Attackers . . . 39
2.6.1 Individual Perspective . . . 40
2.6.2 Organizational Perspective . . . 42
2.6.3 Researcher Perspective . . . 44

3 Impact on Computer Security . . . 47

3.1 Background . . . 49
3.1.1 National ID Cards . . . 49
3.1.2 I-Voting Server Infrastructure . . . 51
3.1.3 Voting Processes . . . 53
3.2 Observations . . . 55
3.2.1 Inadequate Procedural Controls . . . 55
3.2.2 Lax Operational Security . . . 56
3.2.3 Insufficient Transparency . . . 57
3.2.4 Vulnerabilities in Published Code . . . 60
3.3 Experimental Methodology . . . 60
3.3.1 Mock Election Setup . . . 61
3.3.2 Threat Model . . . 61
3.4 Attacks . . . 62
3.4.1 Client-side Attacks . . . 63
3.4.2 Server-side Attacks . . . 66
3.4.3 Attacking Ballot Secrecy . . . 68
3.5 Discussion . . . 69
3.6 Conclusions . . . 70

4 Explaining and Evaluating Operations . . . 73

4.1 Diffie-Hellman Cryptanalysis . . . 75
4.2 Attacking TLS . . . 78
4.2.1 TLS and Diffie-Hellman . . . 79
4.2.2 Active Downgrade to Export-Grade DHE . . . 80
4.2.3 512-bit Discrete Log Computations . . . 82
4.2.4 Active Attack Implementation . . . 84
4.2.5 Other Weak and Misconfigured Groups . . . 85
4.3 State-Level Threats to DH . . . 87
4.3.1 Scaling NFS to 768- and 1024-bit DH . . . 88
4.3.2 Is NSA Breaking 1024-bit DH? . . . 92
4.3.3 Effects of a 1024-bit Break . . . 95
4.4 Recommendations . . . 97
4.5 Disclosure and Response . . . 99
4.6 Conclusion . . . 99

5 Identifying and Analyzing Vulnerabilities . . . 102

5.1 Background . . . 103
5.1.1 Forward Secrecy in TLS . . . 104
5.1.2 Session Resumption . . . 104

5.1.3 Reusing Ephemeral Values . . . 106
5.1.4 Changes in TLS 1.3 . . . 106
5.2 Data Collection . . . 107
5.3 TLS Secret State Longevity . . . 108
5.3.1 Session ID Lifetime . . . 109
5.3.2 Session Ticket Lifetime . . . 110
5.3.3 STEK Lifetime . . . 110
5.3.4 EC(DHE) Value Lifetime . . . 112
5.4 TLS Secret State Sharing . . . 114
5.4.1 Shared Session ID Caches . . . 115
5.4.2 Shared STEKs . . . 116
5.4.3 Shared (EC)DHE Values . . . 116
5.5 Crypto Shortcut Dangers . . . 118
5.5.1 Exposure from Session Tickets . . . 118
5.5.2 Exposure from Session Caches . . . 119
5.5.3 Exposure from Diffie-Hellman Reuse . . . 120
5.5.4 Combined Exposure . . . 121
5.6 Nation-state Perspective . . . 121
5.6.1 The STEK as an Enabling Vector . . . 122
5.6.2 Target Analysis: Google . . . 123
5.7 Discussion . . . 125
5.7.1 Security Community Lessons . . . 125
5.7.2 Server Operators Recommendations . . . 126
5.8 Conclusion . . . 127

6 Future Work and Conclusion . . . 129

6.1 Going Forward . . . 129
6.2 Conclusion . . . 131

Bibliography . . . 133

LIST OF FIGURES

1.1 **NSA's Scale of Attackers**—Stratification of attackers according to an internal NSA presentation [74]. The classes are reasonably believed to be grandmother, 13 year old in the US, script kiddie, 16 year old in Romania, has some talent, has talent, and nation-state [380, 397]. 3

2.1 **Definition of "person"**—Privacy and Civil Liberties Oversight Board's definition of "persons" in relation to Section 702 [234]. 7
2.2 **Area Specialization**—NSA's collaborative effort for exploiting VPN traffic for intelligence [309]. 20
2.3 **Single-Target Pseudocode** 22
2.4 **Multi-Target Pseudocode** 22
2.5 **Tiered Data Types**—The tiers of network traffic and the NSA systems which store and handle that traffic [425]. 26
2.6 **NSA's Technology Risk Matrix**—NSA's risk analysis matrix used to prioritize technologies for resource expenditure [288]. 30
2.7 **Tactical Choke Point**—An example of a tactical choke point with one force stationed north of the river, and its opposition located across the river to the south [62]. 42
2.8 **Network Choke Point**—An example of network structure with an underlying choke point [57]. 42
2.9 **Vuvuzela System Architecture**—The design of the Vuvuzela private messaging system [411]. 44
2.10 **Hornet System Architecture**—The design of the HORNET onion routing system [55]. 44

3.1 **I-voting client**—Estonians use special client software and national ID smartcards to cast votes online. 50
3.2 **Verification app**—A smartphone app allows voters to confirm that their votes were correctly recorded. We present two strategies an attacker can use to bypass it. 51
3.3 **I-voting system overview**—Major components of the system, and how information flows among them. 52
3.4 **Vote casting process** 53
3.5 **Vote verification process** 53
3.6 **Vote tabulation process** 54

3.7 **Official build system is multi-use**— Operators used a PC containing other software, including PokerStars.ee, to sign the official voting client for public distribution. This risks infecting the client with malware spread from the PC. . 58
3.8 **Insecure software downloads**—Operators downloaded software over insecure connections for use in pre-election setup. An attacker who injected malware into these downloads might be able to compromise the process. 58
3.9 **Keystrokes reveal root passwords**—Videos posted by officials during the election show operators typing, inadvertently revealing root passwords for election servers. 58
3.10 **Video shows national ID PINs**—During pre-election setup, someone types the secret PINs for their national ID card in full view of the official video camera. 58
3.11 **Wi-Fi credentials posted**—The official video of the pre-election process reveals credentials for the election officials' Wi-Fi network, which are posted on the wall. 58
3.12 **Personal USB stick**—Against procedures, an official used a USB stick, containing personal files, when moving the official election results off of the counting server. 58
3.13 **Malware records secret PINs**—Estonians use their national ID smartcards to sign and cast ballots. We developed demonstration client-side malware that captures the smartcard PINs and silently replaces the user's vote. 64

4.1 **The number field sieve algorithm for discrete log**—This algorithm consists of a precomputation stage that depends only on the prime p and a descent stage that computes individual logs. With sufficient precomputation, an attacker can quickly break any Diffie-Hellman instances that use a particular p. 75
4.2 **Running time of the NFS Algorithm** 77
4.3 **The Logjam attack**—A man-in-the-middle can force TLS clients to use export-strength DH with any server that allows DHE_EXPORT. Then, by finding the 512-bit discrete log, the attacker can learn the session key and arbitrarily read or modify the contents. $\texttt{Data}^{fs}$ refers to False Start [210] application data that some TLS clients send before receiving the server's Finished message. 80
4.4 **Individual discrete log time for 512-bit DH**—After a week-long precomputation for each of the two top export-grade primes (see Table 4.1), we can quickly break any key exchange that uses them. Here we show times for computing 3,500 individual logs; the median is 70 seconds. 83
4.5 **NSA's VPN Decryption Infrastructure**—Acquired IKE handshakes are passed to CES who uses high-performance computing (HPC) resources to generate the symmetric session keys for ESP traffic [302]. For clarity, POISONNUT is a codename sometimes used in place of VAO [274]. 94

5.1 **Session ID Lifetime**—We measured how long Session IDs were honored by HTTPS websites in the Alexa Top Million. 109
5.2 **Session Ticket Lifetime**—We measured advertised session ticket lifetime and how long tickets were honored by Alexa Top Million websites. 109

5.3 **STEK Lifetime**—TLS connections cannot achieve forward secrecy until the STEK (the key used by the server to encrypt the session ticket) is discarded. . . 111
5.4 **STEK Lifetime by Alexa Rank**—We found 12 Alexa Top 100 sites that persisted STEKs for at least 30 days. 111
5.5 **Ephemeral Exchange Value Reuse**—We measured how long Alexa Top Million websites served identical DHE and ECDHE values (note vertical scale is cropped). 113
5.6 **STEK Sharing and Longevity Visualization**—Each box in this illustration is sized proportionally to the number of domains in that service group and colored according to the observed longevity of the key. Solid red boxes represent groups of domains that shared a key that persisted for at least 30 days. 117
5.7 **Visualizing Session Caches and Diffie-Hellman Reuse**—For comparison with Fig. 5.6, we show similar illustrations of the longevity and cross-domain sharing exhibited by session caches (*left*) and repeated Diffie-Hellman values (*right*). 120
5.8 **Overall Vulnerability Windows**—This CDF depicts the combined effects of exposure from session tickets, session caches, and Diffie-Hellman reuse. 122

LIST OF TABLES

2.1 **Nation-State Attackers**—Widely recognized NS-Attackers, their host country, and previous names which may be seen in other publications and source documents. . . . 12
2.2 **Terminology Usage**—Examples of the terminology used within this text. . . . 12
2.3 **Various NS-Attacker Budget Estimates**—Intelligence Budget Sources: U.S. [409], Russia (FSB, SVR, FBO) [66], U.K. (MI5, SIS, GCHQ) [4] . . . 16
2.4 **Tools: NS-Attacker vs. public**—A comparison of NS-Attacker tools and their publicly available equivalents. . . . 18
2.5 **Proposed Special-Purpose Defeat Hardware**—A sampling of proposed custom hardware for various cryptosystems at various points. All costs are in 2017 dollars and with the exception of DES Cracker are the original authors' estimates. . . . 23
2.6 **Retrospective World Leader Position** . . . 24
2.7 **Tiered NSA Databases**—A sampling of databases NSA analysts use grouped by their type and tier of data. . . . 27

4.1 **Top 512-bit DH primes for TLS**—8.4% of Alexa Top 1M HTTPS domains allow DHE_EXPORT, of which 92.3% use one of the two most popular primes, shown here. . . . 78
4.2 **Estimating costs for factoring and discrete log**—For sieving, we give two important parameters: the number of bits of the smoothness bound `B` and the sieving region parameter `I`. For linear algebra, all costs for DH are for safe primes; for DSA primes with q of 160 bits, this should be divided by 6.4 for 1024 bits, 4.8 for 768 bits, and 3.2 for 512 bits. . . . 88
4.3 **Estimated impact of Diffie-Hellman attacks**—We use Internet-wide scanning to estimate the number of real-world servers for which typical connections could be compromised by attackers with various levels of computational resources. For HTTPS, we provide figures with active downgrade (denoted by "†") and without. All others are passive attacks. . . . 95

5.1 **Support for Forward Secrecy and Resumption** . . . 107
5.2 **Top Domains with Prolonged STEK Reuse**—We show the most popular domains (by average Alexa rank) that reused a STEK for at least 7 days. . . . 112
5.3 **Top Domains with Prolonged DHE Reuse**—We show the most popular domains (by average Alexa rank) that reused a DHE value for at least 7 days. . . . 114

5.4 **Top Domains with Prolonged ECDHE Reuse**—We show the most popular domains (by average Alexa rank) that reused an ECDHE value for at least 7 days. 114
5.5 **Largest Session Cache Service Groups** 115
5.6 **Largest STEK Service Groups** 116
5.7 **Largest Diffie-Hellman Service Groups** 116

ABSTRACT

Nation-state intelligence agencies have long attempted to operate in secret, but recent revelations have drawn the attention of security researchers as well as the general public to their operations. The scale, aggressiveness, and untargeted nature of many of these now public operations were not only alarming, but also baffling as many were thought impossible or at best infeasible at scale. The security community has since made many efforts to protect end-users by identifying, analyzing, and mitigating these now known operations.

While much-needed, the security community's response has largely been reactionary to the oracled existence of vulnerabilities and the disclosure of specific operations. Nation-State Attackers, however, are dynamic, forward-thinking, and surprisingly agile adversaries who do not rest on their laurels and are continually advancing their efforts to obtain information. Without the ability to conceptualize their actions, understand their perspective, or account for their presence, the security community's advances will become antiquated and unable to defend against the progress of Nation-State Attackers.

In this work, we present and discuss a model of Nation-State Attackers that can be used to represent their attributes, behavior patterns, and world view. We use this representation of Nation-State Attackers to show that real-world threat models do not account for such highly privileged attackers, to identify and support technical explanations of known but ambiguous operations, and to identify and analyze vulnerabilities in current systems that are favorable to Nation-State Attackers.

Thesis Statement: By identifying and understanding the characteristics, advantages, and constraints of Nation-State Attackers, we can more easily explain their known operations, identify and analyze current vulnerabilities that they may exploit, and build future systems less susceptible to their abuse.

CHAPTER 1

Introduction

In recent years, nation-states and their associated intelligence apparatuses have been thrust unwillingly into the public perception. Whistleblowers like Edward Snowden, Bill Binney, and Thomas Drake as well as public service organizations such as the Electronic Frontier Foundation, the American Civil Liberties Union, and the Electronic Privacy Information Center have greatly expanded the amount of publicly available information about these intelligence agencies' operations through the release of documents describing not only their operations, but also their oversight and compliance histories. This new wealth of information has spawned thousands of news stories across the world as well as an uncountable number of discussions at all levels of society about the proper balance of security and privacy.

Within the United States, reactions have ranged from approval to indifference to outrage with debates about the appropriateness of the Intelligence Community's (IC) actions have raged in court rooms, legislative floors, and living rooms across the country. Some argue that the IC has run roughshod over the very freedoms that this country was built on as an overreaction to an ever-looming but intangible threat. Others argue that the IC's actions are the only available response given the constant threats to the U.S. population and are necessary for maintaining relative peace and personal safety across the world.

As of yet, neither side has succeeded in convincing the other that their balancing point is the correct choice and it's unlikely that either side ever will. But largely overlooked by many is the fact that the fundamental equilibrium between liberty and privacy has been debated since before the Revolutionary War. And in the two and half centuries since, we've repeatedly found ourselves struggling with analogous and often identical questions.

Presently, we are questioning the National Security Agency's collection of digital information on nearly every person in the U.S. and large portions of the rest of the world. In 1970s, we were questioning the FBI's surveillance of celebrities and covert actions against social-movement groups. In the 1920s, we were questioning the Justice Department's harassment and arrest of political activists. As far back as the 1700s, we were questioning

the English government's use of general warrants and "writs of assistance" to carry out law enforcement activities. While similar in concept, one of the major distinctions between previous and current discussions is not the awareness of the tactic, but the lack of understanding of the methods used to accomplish those tactics.

The ability to open physical mail and record telephone calls was an explainable techniques, whether legal or illegal, prior to the Church Committee's revelations of the FBI and CIA's actions [56, 70]. Using human spies to gather and report information on adversaries was a common sense technique long before the 1920s. Even the simple process of physically searching a house was well known in the 1700s. With regard to the digital age, many of the basic methods used to monitor Internet communications en masse, and thereby surveil the population, were either unknown or assumed to be infeasible. In addition, strong encryption and other proactive protective methods were thought to make even the feasible surveillance methods ineffective [69, 79, 342]; yet we've seen time and time again that their impact is less than sufficient [359, 383].

The revelations that not only were governments around the world performing these methods at mass-scale but also able to defeat many of the proactive protections created chaos and confusion within the computer security community. Researchers, hackers, lawyers, and security practitioners reacted with amazement, curiosity, and anger as the existence of NSA operations were revealed. Not only were many of them unexplainable, many of those that were understood on a technical level were scaled to almost unimaginable levels.

1.1 Computer Security and Nation-State Attackers

Within the computer security community, this newly available wealth of information has been a window into the most secretive and advanced technical organizations in the world. Organizations that control million and billion dollar budgets, employ tens of thousands of workers, and operate on the fringes of the technological frontier by design. These organizations have succeeded in integrating visionaries, theorists, mathematicians, and engineers to a degree that would amaze even the most agile Silicon Valley start-up. Just as amazing is that much of this had been accomplished in near-absolute secrecy with only rumors of vague capabilities escaping the confines of the intelligence services.

As information became publicly available, it became apparent that Nation-State Attackers are fundamentally different from the types of attackers that developers and engineers are accustomed to addressing. Nation-State Attackers themselves have noted the difference between various other tiers of attackers (Figure 1.1) and the "apex predator"-like nature of one another [192]. Their use of mass-scale operations, James Bond Q-like custom hardware,

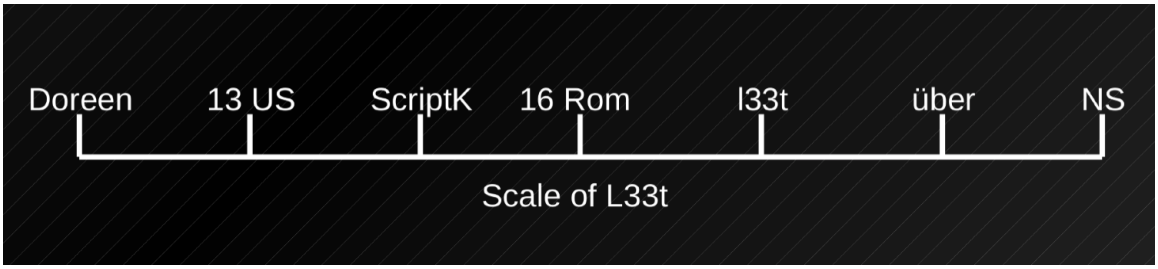

Figure 1.1: **NSA's Scale of Attackers**—Stratification of attackers according to an internal NSA presentation [74]. The classes are reasonably believed to be grandmother, 13 year old in the US, script kiddie, 16 year old in Romania, has some talent, has talent, and nation-state [380, 397].

aggressive maliciousness, and broad reaching judicial abilities have long been speculated but rarely accounted for in the fundamental systems, protocols, and structures of the Internet.

The most straight-forward example of this is the Public Key Infrastructure used for SSL certificates within the Transport Layer Security (TLS) protocol. By validating those who request certificates, limiting the trusted issuers, properly checking the chain of signatures, and verifying the possession of the associate private key, strong guarantees about the identities of TLS endpoints can be achieved. This technique has not only transparently improved the security of nearly all Internet users, but it has also allowed e-commerce to flourish on the Internet. The base assumption of this type of third party attestation is that it is prohibitively difficult for an attacker to obtain a valid certificate and its associated private key required to maliciously impersonate another entity. While neither the protocol, the implementations, nor the base assumption are infallible [8, 177, 396], they have withstood many attempts to compromise connections and protected users' security and privacy for two decades.

Largely safe with regard to many types of attackers an end-user may face, the base assumption of TLS authentication fails spectacularly when confronted with Nation-State Attackers. In reaction to the emergence of the Snowden Documents, the Federal Bureau of Investigations sought and received a court order requiring Lavabit (an e-mail service provider believed to be used by Edward Snowden) to surrender the private key for their browser trusted certificate [336]. The trusted root stores for the majority of devices on the Internet contain root Certificate Authorities (CAs) that are explicitly controlled by various government entities [387–389]. There has even been speculation that root CAs not directly operated by government entities may be compelled to issue fraudulent leaf or intermediate certificates to government entities on demand [381]. Any one of these possible Nation-State Attacker actions leads to a complete compromise of TLS's authentication. Mechanisms such as Certificate Transparency [50] and HTTP Public Key Pinning [178] provide challenges to

abuse, even from Nation-State Attackers, but do not provide complete protection.

While this is a simplistic demonstration of how security protections fail when confronted with Nation-State Attackers, it is nonetheless an ominous one. Without properly addressing Nation-State Attackers in threat models, security and privacy on the Internet can't be adequately protected. Identifying and understanding Nation-State Attackers' behaviors, capabilities, and relative advantages and disadvantages is a necessary precondition to confronting potential abuse by Nation-State Attackers.

1.2 Overview

Since 2013, the year that the Snowden documents were first published, the security community has advanced by leaps and bounds in addressing Nation-State Attackers' known operations and protecting end-users against mass-scale surveillance efforts. Internet destinations are moving to protect client-to-server connections through HTTPS [119]. Major providers now protect internal server-to-server traffic against passive eavesdropping [215, 230]. The IETF approved RFC 6973 to address privacy-related issues in the design phase of standards and protocols [65]. Developers have begun to widely deploy end-to-end encryption in interpersonal communications apps [373, 419]. And many legal challenges have been launched to end certain practices as well as provide transparency into the use of others [6, 88, 202]. While all of these actions are useful, helpful, and needed, they are reactionary to specific programs or techniques that were exposed.

I believe that this type of reactive action does not adequately protect end-users against future Nation-State Attackers' actions (or possibly current actions). Nation-State Attackers are a mobile and future-conscious adversary who are always hunting for an exploitable weakness in *next* protocol and the *next* encryption scheme. When their current techniques, tactics, and procedures cease to provide the wealth of information that they are accustomed to, they will evolve their strategy to return to their previous position of information dominance.

Thesis Statement By identifying and understanding the characteristics, advantages, and constraints of Nation-State Attackers, we can more easily explain their known operations, identify and analyze current vulnerabilities that they may exploit, and build future systems less susceptible to their abuse.

Structure In Chapter 2, we will outline a framework for understanding Nation-State Attackers including their characteristics, advantages, constraints, and discuss accounting for Nation-State Attackers. In Chapter 3, we will discuss the impact of Nation-State

Attackers on real-world threat models as described in our 2014 publication "Security Analysis of the Estonian Internet Voting System" [386]. In Chapter 4, we will discuss explaining and evaluating known Nation-State Attacker operations as described in our 2015 publication "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice" [8]. In Chapter 5, we will discuss identifying and analyzing vulnerabilities which favor Nation-State Attackers as described in our 2016 publication "Measuring the Security Harm of TLS Crypto Shortcuts" [385]. In Chapter 6, we will briefly discuss appealing directions for future efforts, review the material presented, and conclude.

CHAPTER 2

Nation-State Attacker Model

> *It is by comparing a variety of information, we are frequently enabled to investigate facts, which were so intricate or hidden that no single clue could have led to the knowledge of them. In this point of view, intelligence becomes interesting which but from its connection and collateral circumstances, would not be important.*
>
> – General George Washington [189]

In order to appropriately account for and protect against Nation-State Attackers (NS-Attackers), we must first attempt to understand them at a fundamental level. Unfortunately, the opaque and secretive nature of NS-Attackers renders the standard application of the scientific method impractical for many reasons. Chief among them is the inability to formulate and conduct an experiment that produces an observable result (an NS-Attacker behavior).[1] Any effort to elicit a measurable reaction from an NS-Attacker that would constitute a concrete experimental result is fraught with real danger to the researchers' security and privacy as well as ethical concerns of intentionally diverting NS-Attackers' resources from more pressing matters.

Instead, we take an inductive approach to extract insights into NS-Attacker operations and behaviors. While less common in engineering disciplines, this approach is commonly applied in fields such as economics, physics, and biology where real-world experimentation is infeasible or similarly constrained by practicality. Inductive reasoning allows us to draw conclusions based on a set of assumed true and representative observations but not to the rigor found with deductive reasoning [171]. With regard to NS-Attackers, we have a set of data-points in the form of publicly known operations and source documents from which we can generalize and extract higher-order information. We can not, however, assert that our conclusions are correct or all encompassing; only that, given the information we currently

[1] In later chapters, we will show that the scientific method is directly applicable to studying and evaluating their potential impact.

First, Section 702 authorizes the *targeting of persons*.[47] FISA does not define what constitutes "targeting," but it does define what constitutes a "person." Persons are not only

---

individuals, but also groups, entities, associations, corporations, or foreign powers.[48] The definition of "person" is therefore broad, but not limitless: a foreign government or international terrorist group could qualify as a "person," but an entire foreign country cannot be a "person" targeted under Section 702.[49] In addition, the persons whom may be

Figure 2.1: **Definition of "person"** — Privacy and Civil Liberties Oversight Board's definition of "persons" in relation to Section 702 [234].

have, our conclusions are supported by known data-points and are logically consistent with our understanding of the world. When additional data-points, observations, or source documents become available, it may be necessary to revisit our analysis and ensure that it is still consistent with the known information.

In this chapter, we present our results from an in-depth study of the available data-points and observations of Nation-State Attackers. At a high level, we propose a model of how we believe NS-Attackers behave and view the world. Specifically, we propose a lexicon to use when discussing NS-Attackers, describe the characteristics that set NS-Attackers apart from other attackers, discuss the distinct advantages and constraints of NS-Attackers, contrast various NS-Attackers' behavior patterns, and identify concepts which individuals, organizations, and researchers should recognize and take into account when operating in scenarios that may include Nation-State Attackers.

2.1 Lexicon

In this section, we define a lexicon for use regarding Nation-State Attackers. Computer science already possesses a wide array of precise vocabulary to define attackers and attacker actions, but in many cases, it does not easily translate to NS-Attackers. In addition to this, "wordsmithing" is a commonly used tactic used by NS-Attackers when forced to publicly address aspects of their operations. This creates a confusing and counter-intuitive set of self-referential and intentionally misleading statements by "down-shift[ing]" language or opaquely responding to broad questions with answers only accurate to a much more restricted version of the question [89].

A simplistic example of this is the U.S. Government's definition of the word "person". As seen in Figure 2.1, this definition is significantly broader than would be commonly assumed and this subtlety can significantly alter the meaning of otherwise obvious statements. Counter-intuitive terminology of this type is pervasive in source documents and legal interpretations regarding Nation-State Attackers and a misunderstanding in the vocabulary or proper connotation can severely degrade comprehension of our Nation-State Attacker model.

In order to provide a foundation for our discussion, this section presents a lexicon for the NS-Attacker context that allows us to speak in a direct and concise manner without sacrificing accuracy. An effort has been made to avoid collisions with other lexicons used by NS-Attackers or other groups (e.g. two words with conflicting definitions) and we do attempt to include all reasonable uses of other lexicons' terminology within ours. Additionally, we strive to use language whose colloquial definitions closely match the full definition within our lexicon in order to make it approachable to technical and non-technical persons alike.

2.1.1 Actors

Victim We define a victim as *an entity directly affected by an action*. Similar to "person" in Figure 2.1, a victim may be an individual or group of individuals. Additionally, a victim may or may not be related to the action's intent but need only be affected by the action. For example, if an NS-Attacker infects a public library computer with malware that reports back all keystrokes in order to surveil a particular user of that computer, all users of that computer are victimized by the action.

We should note that the term "victim" is not intended to imply guilt or innocence, nor should it be construed to indicate approval or disapproval of any individual action. When necessary, we use the phrases "justifiable victim" and "unjustifiable victim" to delineate between specific victims who would or would not be commonly considered an appropriate recipient of an NS-Attacker's actions given sufficient characterization.

Nation-State Attacker We define a Nation-State Attacker (NS-Attacker) as *an entity operating on behalf of a recognized government with the characteristics of Sovereignty, Access, and Money* (described in-depth in Section 2.2). While commonly interpreted as state-sponsored intelligence agencies, our terminology applies to a broader group of organizations who act directly on behalf of and with the authority of a government similar to a state-sponsored intelligence agency but not directly attached (described further in Section 2.5.3). Table 2.1 lists widely acknowledged NS-Attacker entities for major countries.

For conciseness, we will use the associated abbreviation in instances where we are discussing a specific NS-Attacker.

2.1.2 Actions

Acquire We define acquisition as the act of *deliberately obtaining and storing information*. Whether it is an RF receiver collecting over-the-air WiFi content, malware reporting keystrokes, or the date-time of a cellphone call, the important distinction is that the NS-Attacker retains a record of it. Acquisition does not include ephemeral, non-persistent storage such as a packet traversing an infected router or being temporarily maintained in-memory while it is decided whether a record should be created or not.

Intrude We define intrusion as the act of *altering the state of a device in a way that is detrimental to the users of that device*. This action covers insertion of functionality (adding or modifying applications or dependencies), altering configuration, or otherwise interfering with the standard operation of a device including the degradation of the device's usefulness.

Within the source documents, this is often referred to as "Computer Network Operations" (CNO) with sub-classes of "Computer Network Attack" (CNA) and "Computer Network Exploitation" (CNE). The delineation between CNE and CNA is that the intent of CNE is to acquire information where as the intent of CNA is to degrade the usefulness of the device [168]. CNE can be thought of as "surveillance intrusions" and CNA as "destructive intrusions".

Defeat We define defeat as the act of *overcoming intentional security protections*. A defeat could take the form of evading detection by a host-based antivirus application, using certain communications channels to avoid network-based detection, or recovering plaintext of encrypted communications. By definition, a defeat does not rely on the binary exploitation of any device as that would be considered an intrusion.

While this action could be split into multiple sub-types depending on what is being defeated, we believe that this broad definition significantly improves the usability of our lexicon. We should note that while source documents often use the term "defeat" in a way that closely aligns with our definition [135, 262], others use it in relation to responsive results from a database query that were suppressed [398, 427]. Our definition of defeat explicitly rejects this search related use of the term as it is a disparate from the colloquial definition and a collision within NS-Attackers' own terminology.

Monitor We define monitoring as the act of *retrieving acquired information for the purpose of analysis*. This may be information about a specific victim/device or group of victims/devices and may consist of content, metadata, or any other derived information. While many use the term "surveillance" in this way, we believe that this term is overly broad and encompasses other actions such as acquisition or intrusion. Some make the distinction between retrieval and use by a human (as a search result) versus an algorithm (as an input to data-mining) [142] but we do not separate these two cases and consider use of acquired information in any form to be monitoring.

2.1.3 Descriptive

Active vs. Passive The delineation between active and passive is fairly straightforward and indicates the *ability of the victim to detect the NS-Attacker's action*. An action is considered detectable (and thereby active) if proof positive technical evidence *could* be presented by the victim (with or without attribution) to be independently verified. If a victim *could not* present such evidence, it would be considered a passive action.

Proof positive evidence includes a fraudulent SSL certificate or a copy of malware presented to others who would be able to verify that it exists. More generically for actions at a network layer, if the end-points of a conversation are able to compare transcripts via a pristine out-of-band channel, they would be able to determine that a they differ in a meaningful way (e.g. both sides' packet captures of a TLS handshake). Conversely, an example of a passive action is a receive-only RF listening station, a network tap, or a cryptographic defeat accomplished using only previously acquired ciphertext.

By definition, intrusion actions are always considered active and monitor actions are (nearly) always passive. In the case of an NS-Attacker who decrypts communications via stolen cryptographic material, theft of the key material from an end-point is designated as an active intrusion while the decryption of the communication with that key material is a passive defeat. Information released by an NS-Attacker or associated entity that could have been obtained only through the monitoring of a victim (e.g. discussion in an e-mail, participation in an online community, etc.) is an example of active monitoring.

Targeted vs. Untargeted As opposed to active vs. passive, targeted vs. untargeted is a more nuanced delineation. An action is designated as targeted or untargeted based on the *relationship of the action to the victim*. Targeted actions involve an identity "selector" of some sort, such as an IP address, e-mail address, or web cookie [2, 142]. Selectors reflect characteristics of a specific victim or device and not other victims or devices. For example,

if a source or destination e-mail address determines whether a network tap stores a copy of a specific e-mail for later monitoring, we label it as targeted acquisition.

Untargeted action either do not use a selector or use selectors that are shared between both "interesting" victim as well as "uninteresting" victims. If all traffic that passes a network tap is stored (often described as "full-take" [139, 425] or "bulk unselected" [136]), we label it untargeted acquisition. Similarly, actions that rely on keyword selectors are considered to be untargeted as the action's trigger does not reflect identity information [277].

Terminal vs. Enabling The distinction between terminal and enabling actions is also less rigid than active vs. passive. Terminal vs. enabling indicates the *intent or end-goal of an action*. An action intended to produce actionable intelligence[2] or answer crucial analytic questions is labeled terminal. Acquiring a victim's e-mail messages in order to receive early warning of events is terminal acquisition. Conversely, an action that is intended to facilitate follow-on actions is labeled enabling. Acquiring and monitoring a network administrator's communications to learn information about a router or network configuration for follow-on intrusions is enabling acquisition [134].

In some circumstances, an action can be considered both enabling and terminal. For example, an intrusion against a network router to steal a point-to-point IPsec PSK for external traffic is an intrusion to *enable* a passive defeat of VPN encryption. An intrusion against a network router install malware that looks for the presence of selectors and reports communications with those selectors to an external device is a *terminal* intrusion. If both the IPsec PSK is stolen and the malware installed by the same intrusion, it is both an enabling and terminal intrusion.

2.1.4 Usage

For context and real-world usage of our lexicon, Table 2.2 provides example techniques that an NS-Attacker may employ along with their appropriate classification.

[2] Actionable intelligence is information that an NS-Attacker can use to accomplish a broader goal such as counter-terrorism or economic/diplomatic espionage.

Country	Current Organizational Name	Abbreviation	Previous/Alternative Organizational Names and Abbreviations
U.S.	National Security Agency	NSA	-
China	Third Department of the People's Liberation Army General Staff Headquarters Department	GSD 3rd Dept	Unit 61398 and APT1 as members of the subordinate Second Bureau [225, 394]
Russia	Special Communications and Information Service of the Federal Protective Service of the Russian Federation	Spetssvyaz	Federal Agency of Government Communications and Information (FAPSI/FAGCI)
U.K.	Government Communications Headquarters	GCHQ	-
Germany	Federal Intelligence Service	BND	Bundesnachrichtendienst

Table 2.1: **Nation-State Attackers**—Widely recognized NS-Attackers, their host country, and previous names which may be seen in other publications and source documents.

Description	Action	Active vs. Passive	Targeted vs. Untargeted	Terminal vs. Enabling
– Searching a database for a specific person's communications to find indications of an attack	Monitor	Passive	Targeted	Terminal
– MitM-ing a network administrator to capture SSH password	Acquire	Active	Targeted	Enabling
– Storing all data received from a network tap regardless of user or content	Acquire	Passive	Untargeted	Dual-Intent
– Requiring a service provider to turn over all call records for all users	Acquire	Passive	Untargeted	Terminal
– Factoring Facebook's RSA SSL modulus in order to recover the private key	Defeat	Passive	Untargeted	Enabling
– Using a GCD algorithm on all SSL public keys to find shared factors	Defeat	Passive	Untargeted	Enabling
– Storing TCP streams from a network tap that contain a specific suspected terrorist's e-mail address	Acquire	Passive	Untargeted	Terminal

Table 2.2: **Terminology Usage**—Examples of the terminology used within this text.

2.2 Characteristics of a Nation State Attacker

In common discussions, NS-Attackers are characterized with hyperbolic language on the order of "unequaled Gods of all technical matters". With their army of mathematicians, cadre of elite hackers, and Enemy of the State-style tactical operations, NS-Attackers constitute a undoubtedly a formidable adversary. While none of these attributed capabilities are unwarranted [10, 17, 156], they are not a complete, or necessarily accurate, representation of NS-Attackers and their abilities. Additionally, more academic descriptions such as "global passive adversary" [79] or "Advanced Persistent Threat" [35] also fail to fully describe the attributes of NS-Attackers. In our analysis, NS-Attackers are rational actors whose seemingly unique and unparalleled abilities stem from three distinct characteristics. In this section, we identify, discuss, and give known examples of NS-Attacker characteristics: Sovereignty, Access, and Money.

2.2.1 Sovereignty

A defining characteristic of a NS-Attacker is that it acts with the authority of a sovereign entity — the respective government — that maintains control of a geographic region and judicial systems. While various governments and their views on citizens' freedom of speech and expression differ, citizens' rights, and the limits to those rights, are decided by their respective government.

Legal Writs NS-Attackers exercise this sovereignty by using legal writs to increase their acquisition capabilities. These writs commonly take the form of search warrants issued by a judge or magistrate by which an individual or organization can be required to disclose information to government entities. Depending on the country and specific writ, they may be unrelated to information about users [336], about *all* users without regard to identity [235, 414], or without direct approval from a judge/magistrate [87].

Publicly available transparency reports from major technology companies show that many countries make use of these writs and that they often affect large numbers of users [115, 160, 241]. While their use has been successfully challenged in some cases [48, 87], compliance is usually encouraged through the use of, sometimes exorbitant, fines [336,350,408,428] or imprisonment [408].

Policy Influence Another way NS-Attackers use their characteristic of Sovereignty is by influencing the creation of policy within their respective countries. Most directly, this approach can be used to either increase their ability to acquire information via legal writs or

to help shape the interpretation of existing policies more the NS-Attacker's liking (discussed more in-depth in Section 2.4.3) [261]. The Russian Yarovaya law [347] and the U.K. Investigatory Powers Act 2016 [404] explicitly require data retention by providers such that government entities can later request it. While the USA Freedom Act does not require retention explicitly for intelligence purposes, retention of certain information is required for other purposes [362] and the bill explicitly lays out the process for access to that data [221]. In all of these instances, NS-Attackers (Spetssvyaz, GCHQ, and NSA respectively) improve the usefulness of legal writs they already have the power to leverage.

Another way that NS-Attacker use policy influence is to bound the effort they must invest for successful defeats. The Russian Yarovaya law requires backdoors in encrypted communications which allows access to the plaintext contents on-demand [130]. Russia and China also are exerting pressure on companies who offer VPN and other anonymization services to reduce the effort required to deanonymize individual users and expose their communications [251]. While the U.S. does not currently have legal requirements to ensure law enforcement or intelligence access to information, this was not always the case [125] and is a source of frequent debate [43, 318].

2.2.2 Access

Another characteristic of NS-Attackers is their vast amount of access to information. As described above, NS-Attackers are able to acquire information via legal writs issued directly to a victim or a victim's service provider. In contrast to this, the Access characteristic is focused directly on acquiring raw information. While information compelled from others is highly useful, raw information, such as streaming TCP connections or individual messages and their timing, allows the NS-Attacker to be more focused on victims' communications and avoids the complications of organizations who begrudgingly comply with these legal writs to the minimum amount necessary [374].

Traffic Concentrations One way of leveraging this access is by taking advantage of Internet traffic concentrations that exist due to the physical attributes of the Internet as a whole. While we'll discuss "choke points" more in-depth in Section 2.6, the broad idea is that while Internet topology is commonly referred to as a web of connections, it is in reality more similar to hub-and-spoke configuration. Whether by undersea connections, high-bandwidth fiber links, or Tier 1 ISPs, paths between endpoints often traverse a relatively small collections of entities rather than being distributed across all possible paths.

NS-Attackers are aware of and use these concentrations to acquire raw data in massive

quantities. NSA's many "Upstream" [23] and "Special Source Operations" programs are known to "[l]everage unique key corporate partnerships" [390] to acquire data while it transits one of these links. The GCHQ, BND, and New Zealand's Government Communications Security Bureau (GCSB) are believed to possess analogous operations under the "Special Source Access" [146], `WHARPDRIVE` [364], and `SPEARGUN` [250] programs respectively.

Uncommon Vantage Points In addition to network access in locations controlled by others, NS-Attackers also have the ability to gain access to a specific vantage point if the need exists. NS-Attackers have a long history of going to great lengths to gain access to communications they deem important. During the Cold War, Operation Ivy Bells consisted of specially fitted submarines capable of tapping underseas cables in order to acquire the information passing through them [153]. The digital age sees an update to that same process. While it is relatively straightforward to tap fiber-optic cables in ISP facilities [377], it is widely speculated that underwater fiber-optic tapping is within the repertoire of NS-Attackers like the NSA [267, 337]. NSA also is known to use U.S. Navy subs to act as network bridges to transit SIGINT data from forward deployed sites to central processing facilities [256].

An especially useful vantage point that NS-Attackers possess is through satellites and other "overhead" acquisition platforms [372]. These vantage points provide insight into the base-station-to-satellite transmissions ("uplink") [167] and other types of transmitters for both data acquisition [167] and geolocation [371]. The advantage of overhead acquisition platforms is twofold. First, they are able to cover a substantially larger geographic area than terrestrial platforms which are limited by radio horizons[3]. Second, overhead platforms are also often mobile and can change coverage areas on-demand with substantially less effort than moving terrestrial RF or link-based platforms.

2.2.3 Money

The final characteristic of NS-Attackers is their ability to leverage resources which would otherwise be unattainable due to financial constraints. With the exception of the 2013 U.S. National Intelligence Program budget [254], little is known about the budgets of individual NS-Attackers. Instead, allocations to parent entities at various levels and of unknown completeness have been released and estimated.

[3] A radio horizon is the furthest distance a ground wave can travel and still reach its recipient. While some frequency bands such as HF can propagate over the horizon via atmospheric refraction, most common consumer frequency bands are in the VHF/UHF/SHF range and are limited by line-of-sight propagation.

Country	Military Budget [247]	Intelligence Budget	"Cyber-war" Budget [429]	Est. NS-Attacker Budget (2016)
U.S.	$611B (2016)	$70.7B (2016)	$7,000M (2016)	~$6,110M
China	$215B (2016)	-	$1,500M (2016)	~$2,250M
Russia	$69B (2016)	$5.4B (2010)	-	~$693M
U.K.	$48B (2016)	$3.3B (2016)	$450M (2016)	~$483M

Table 2.3: **Various NS-Attacker Budget Estimates**—Intelligence Budget Sources: U.S. [409], Russia (FSB, SVR, FBO) [66], U.K. (MI5, SIS, GCHQ) [4]

Own Funding Table 2.3 compares the budgets of the four countries at various granularities. The U.S. National Security Agency (NSA) is the only NS-Attacker with a known budget datapoint of $10.4B in direct funding for 2011 [254] (13.23% of the entire U.S. IC budget), but it is assumed that other comparable countries' NS-Attackers operate on similar portions of their overall military (~1%) and their intelligence budgets (~10%). The estimate for each NS-Attacker's 2016 budget (using 1% of Military Spending) are shows in Table 2.3.

With these massive budgets, NS-Attackers are able to invest in capabilities unavailable to other types of attackers. Large workforces [4, 254], custom small-scale efforts (e.g. `ANT` products) [17], and dedicated super computers [298] are just the start. In the next section, we will investigate the advantages that this financial capability facilitates.

Others' Funding While some intelligence services are not as financially well resourced as others, they are nonetheless characterized as a NS-Attackers. NS-Attackers are also known to share intelligence [18], techniques [260], expertise [261], and even direct funding [144, 253] with their other NS-Attackers in 'friendly' countries. This generosity allow NS-Attackers who lack their own financial resources to invest in research and development to benefit from the fruits of others' investments if they are able to coordinate with NS-Attackers that can.

An example of this is the Bundesnachrichtendienst (BND), Germany's Foreign Intelligence Service. With a 2016 public budget of approximately $712M [206] to cover both SIGINT and HUMINT (as well as others) [37], BND is significantly more financially constrained compared to NSA or GCHQ. Through partnerships, BND gained access to `XKEYSCORE` and cryptanalysis capabilities [310] as well as proven tradecraft and methodologies [261] in exchange for language support and "unique accesses in high interest target areas" [261]. By doing this, BND avoids devoting financial and manpower resources to duplicate already proven systems and techniques. Instead, they can barter for them at little direct cost.

2.3 Advantages of a Nation State Attacker

While NS-Attackers' characteristics set them apart from other types of attackers, there are distinct advantages that they have compared to other types of attackers. In this section, we step back and look at four advantages with regard to computer security and exploitation. Specifically, we discuss: Near Superset Attacker, Specialization, Non-Symmetric Defeats, and Distant Return Horizons.

2.3.1 Near Superset Attacker

> *X-KEYSCORE is Bro plus memory. FOXACID is Metasploit with a budget. QUANTUM is AirPwn with a seriously privileged position on the backbone.*
>
> – Bruce Schneier [360]

First, NS-Attackers' capabilities are a near superset of those available other types of attackers. By following and staying up to date on the advances in both the academic and non-academic computer security research communities, NS-Attackers have a consistent inbound flow of tactics, techniques, software, and discoveries. In some cases, NS-Attackers may be able to incorporate these external advances with their own in-house advances, but they can also use them to blend-in with other types of attackers. NS-Attackers are cognizant that they must "Walk The Line" of advertising their capabilities and "[b]e awesome enough to be useful [...] but not as awesome as a state actor" [74]. By doing this, NS-Attackers significantly complicate attribution to the extent that researchers and defenders "can't easily tell the difference between a couple of guys in a basement apartment and the North Korean government with an estimated $10 billion military budget" [361].

Hardware and Software Tools Much of the highly specialized software and hardware that NS-Attackers use is comparable to publicly available implementations. Table 2.4 shows a sample of public and private tools with similar functionality. NS-Attackers also are known to buy Commercial Off The Shelf (COTS) tools to incorporate in their systems. NSA and GCHQ have experimented and used Cavium hardware accelerators [49] for high-speed decryption in conjunction with `LONGHAUL` [132, 306] and the `BADDECISION` suite is known to include macchanger, wireshark, nmap, and ettercap [21]

In addition to the tools, many vulnerabilities and exploits are also shared between NS-Attackers and other classes. Exploit Database [114] and other similar websites provide annotated, categorized, and ready-to-use exploits for publicly known vulnerabilities. The former head of NSA's TAO even claims that any sufficiently large network can be exploited

Type	Description	NS-Attacker	Publicly Available
Hardware	RF collection	DRT [81]	HackRF [321]/RTL-SDR [349]
	IaaS	Private Cloud [379]	Amazon AWS
	Programable Hardware	ICEPIC [280]	Papilio [326]
Collection and Processing Software	Packet Collection	Packet Splatter [264]	tcpdump
	TCP Re-assembly	xFip [264]	libtins [214]/gopacket [161]
	Protocol Disector	Promotor/GENESIS [264]	WireShark
	Document Extraction	XKEYSCORE [311]	NetworkMiner [1]
	Logo Detection	XKEYSCORE [311]	Google Cloud Vision API [159]
	Network Analysis	IMMINGLE [138]	NetMiner [249]

Table 2.4: **Tools: NS-Attacker vs. public** — A comparison of NS-Attacker tools and their publicly available equivalents.

with persistence and focus without a 0-day vulnerability [192]. NS-Attackers are also have the ability to pay the high-prices for vulnerabilities companies that cater to nation-states as well as the traditional grey/black vulnerability market [5].

Public Information In addition to hardware and software, NS-Attackers also share access to large amounts of information with other types of attackers in the form of social media, academic publications, and successful techniques. These shared information sources can then be imported by an NS-Attacker for direct use or can be leveraged to improve existing capabilities.

One example is the use of Open-Source Intelligence (OSINT) which NSA and GCHQ are known to use for advanced warning, target tracking, and target development [192, 320]. By monitoring public forums and communications channels, NS-Attackers can freely acquire data with very little investment. NSA also published a 650 page guide on Internet research covering such topics as "Search Fundamentals", "Google Hacking"[4], "Using the Internet to Research Companies", and "Uncovering the "Invisible" Internet" [3].

NSA's TREASUREMAP program for "a near real-time, interactive map of the global internet" relies on OSINT, academic, and commercial data sources of information in addition to internal sources [273]. The evolution of Internet-scale scanning and probing projects such as Shodan [369] and Censys [82] provide an extra boost to NS-Attackers' Near Superset Attacker advantage as they can directly replace internal port and protocol scanning. Of the four ports/protocols known to be of interest to GCHQ's HACIENDA program (80/HTTP, 443/HTTPS, 21/FTP, and 111/RPC) [200], half (80/HTTP and 443/HTTPS) are provided in full and a third (21/FTP) in part by Censys on a rolling basis in addition to other potentially

[4] Special/advanced techniques for using the Google search engine

useful protocols such as 22/SSH and 25/SMTP.[5]

Social Attacks NS- and non-NS-Attacker also share the ability to mix technical and non-technical approaches. Technical approaches can be used to enable or improve operations that rely on social or HUMINT (HUMan INTelligence) techniques. An example of this would be monitoring a victim's online presence to schedule and improve the success of in-person actions [143, 183]. In addition, attackers can use traditionally social/HUMINT operation tactics to improve technical attacks. Phishing, waterholing, and others use trust, behavior patterns, and/or psychology to improve technical attacks [192].

2.3.2 Specialization

> *[W]e've got people who will know the security functionalities of those devices [inside the victim network] better than the people who developed the actual devices*
>
> –Rob Joyce (Former head of NSA's Tailored Access Operations (TAO)) [192]

NS-Attackers can dedicate resources and effort to specific areas or techniques. While the overall scope of NS-Attackers' responsibilities are often enormous, the workforce is able to specialize into narrow areas of concentration and this is reflected in the fractal nature of NS-Attackers' organization structure [12, 36, 394]. Each NSA unit is given a designator of [Directorate][indirect report n][indirect report $n-1$]...[direct report][unit].[6] Even within units, individual personnel have specific areas of responsibility whether its a protocol, database, or victim/geographic area [308].

Area Focus The onboarding process for new NS-Attacker employees is undoubtedly an exhausting and demanding learning process above and beyond the base skill-set required for other types of attackers[7]. Learning the internal minutia, needed historical cultural and behavioral aspects, opaque program names, and alphabet soup of acronyms [291] is a one-time cost though. Afterwards, the new employee is able to specialize on their specific duties due to the expansive workforce. A linguist doesn't need to know how to most effectively

[5] In the case of research that does not release raw scan data [384], NS-Attackers can acquire much of the raw scanning results from the network traffic itself.

[6] For instance S31176 "Custom Thread Development for Network Encryption" reports to S3117 "Cryptanalytic Exploitation & Discovery" who reports to S311 "Office of Target Pursuit", etc. all within the Signals Intelligence Directorate [308].

[7] Examples of base skill-sets needed by other NS-Attacker and non-NS-Attacker workers are linguistic training, computer science education, understanding of advanced mathematics, etc..

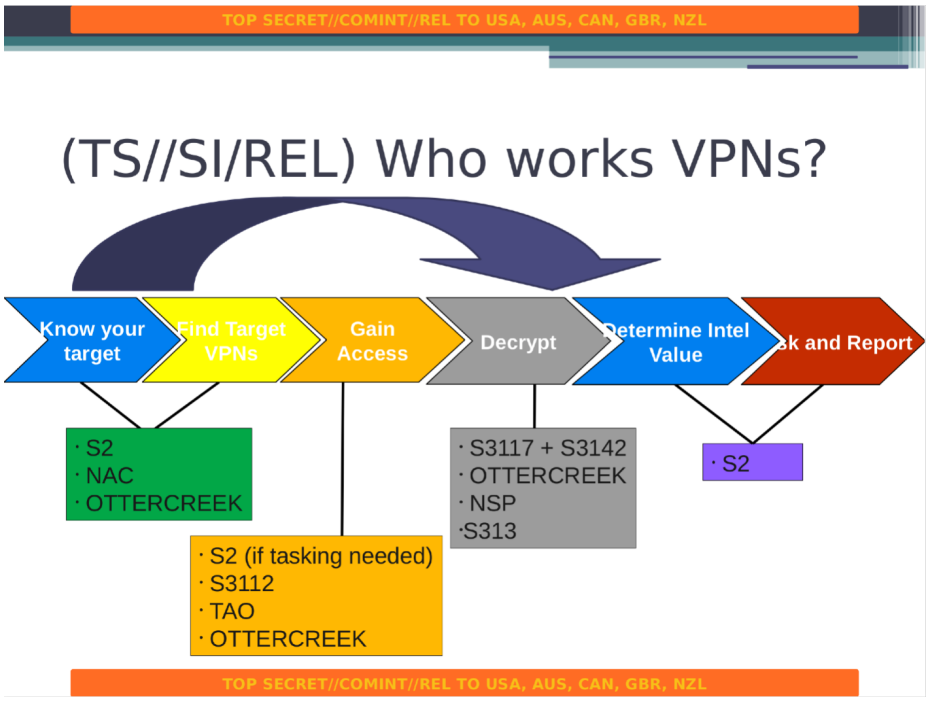

Figure 2.2: **Area Specialization**—NSA's collaborative effort for exploiting VPN traffic for intelligence [309].

acquire content. An analyst doesn't need to know how to defeat encryption to gain access to plaintext content. And a cryptographer doesn't need to know how to read Russian, Chinese, or Arabic.

While area focus does exists with non-NS-Attacker communities [325,400], NS-Attackers are able to leverage this to much more significant levels. Although analysts require understanding of the largest diversity of technologies, tools, and techniques to perform their duties, their knowledge is mostly used to know when to use what technique with which tool to gain actionable intelligence from which technology [287]. Separate groups and teams of employees are dedicated to each technology and each tool to relieve the analyst from having to understand each area in depth [309]. Figure 2.2 shows the interaction of these different groups to successfully monitor victims' VPN traffic.

Mission Focus NS-Attacker employees have the ability to specialize on a subset of victims. While some of it is out of practicality (e.g. a Russian linguist will focus on Russian-speaking victims instead of German), the advantage comes from the deep understanding that employees are able to build over time. General Hayden describes the usefulness of this type of focus and deep understanding as it applies to identifying victims by voice, even in the background, as well as interpreting the "elliptical, metaphorical, indirect, nuanced . . . and

clever" manner of speech used by non-military victims [168].[8] Mission focus applies to technical matters as well. Setting up a system for vulnerability analysis can take a month or more and the actual time spent looking for a vulnerability may be multiple months [5].

2.3.3 Non-Symmetric Defeats

> *So in war, the way is to avoid what is strong and to strike at what is weak.*
>
> – Sun Tzu [407]

NS-Attackers have the ability to leverage non-symmetric defeats against victims. In this context, we use the term "non-symmetric" to indicate that the total resource cost does not scale linearly with the number of instances defeated. While we will focus on cryptographic defeats, analogous constructions are available for other types of security protections. We'll discuss three specific forms of non-symmetric defeats: *multi-target*, *asymmetric*, and *guided*.

Multi-target Defeat When defeats are evaluated in academic and industry literature, the dominant metric is usually the cost to defeat a single instance. For example, the security of a single block AES-128 encrypted message depends on the effort it takes to recover the plaintext from *that* ciphertext. Currently, the best known key recovery attack against full strength AES-128 costs $2^{126.1}$ [39] which is approximately 75% less computationally intensive than the complete 2^{128} brute-force attack but is still infeasible.

With multi-target defeats, the construction changes from an attacker wishing to defeat a specific instance to the attacker wishing to defeat *any of a set* of instances. Expanding the set of success outcomes substantially alters the investment profile of the defeat. The computational effort required test n keys against a single ciphertext can instead be used to test a smaller number of keys against a larger number of ciphertexts in order to change the expected cost for *a* successful outcome.

For example, consider a modified version of Bernstein's "Break a dozen secret keys, get a million more for free" [30]. Imagine using an exhaustive search of the key space search against a theoretical protocol which uses AES-128-ECB encryption and whose first ciphertext message is a confirmation of correct session keys defined as $E_k(\texttt{is-key-correct??})$. Against one ciphertext, an attacker expects to invest 2^{127} work to find the correct key and a worst-case of 2^{128}. But if the attacker wishes to find the keys associated with 2^{40} ciphertexts, they can reuse the work for on attempt against all 2^{40} ciphertexts. This results in an

[8] In some cases, this focus may be less of an advantage. General Hayden also recounts how "targeted killings" can take an emotional toll on the linguists who have been monitoring the victim [168].

```
ct = read_ciphertext()
for poss_key in range(2^128):
  poss_ct = encrypt_first(poss_key)
  if poss_ct == ct:
    return poss_key
```

Figure 2.3: **Single-Target Pseudocode**

```
b_filter = bloom_filter(ciphertexts)
for i in range(2^128):
  poss_key = rand()
  poss_ct = encrypt_first(poss_key)
  if poss_ct in b_filter:
    yield poss_key
```

Figure 2.4: **Multi-Target Pseudocode**

expected cost of 2^{88} to find *a* key for *a* ciphertext. Figures 2.3 and 2.4 show pseudocode implementations of both the single-target and multi-target version of this theoretical attack.

While not all attacks nor all protocols lend themselves to this type of construction, ones that do are highly advantageous to NS-Attackers due to the scale of their operations and their ability to invest heavily in a small number of defeats.[9] Similar to the well-known space-time tradeoff, these types of attacks often have multiple parameters that can be tuned for feasibility or a more advantageous trade-off.

Asymmetric Defeats Asymmetric attacks allow the attacker to leverage a one-time resource expenditure to reduce the cost of per-instance defeats. A well known example of this is the use of rainbow tables for pre-imaging hashes. The one-time resource investment to compute adequate rainbow tables allows a given pre-image defeat to be accomplished with substantial less effort.

Some asymmetric defeats are made feasible through the use of special-purpose hardware. Although few special-purpose devices are known to ever been created, many attacks can be improved through the use of GPUs, FPGAs, or ASICs [131]. These types of performance enhancements trade generality and money for a reduction in wall-clock time. Precise estimates of ASIC-based attacks is difficult due to intellectual property constraints and non-disclosure agreements, but it is estimated that design costs start at $1M and grow with the complexity [317].

Figure 2.5 shows the estimated cost and performance of various proposed special-purpose devices of which only EFF's DES Cracker is known to have been built. The underlying technology for the year proposed as well as the theoretical nature of these devices explains the apparent disparate estimates in both cost and time. Attacks using such costly device are largely out of reach for non-NS-Attackers, but their characteristic of Money puts them within reach for NS-Attackers. In Chapters 4 and 5, we'll discuss two of these asymmetric attacks

[9] Any resources invested on an expensive defeat against a single or small number of instances can instead be invested in large numbers of instances.

Cryptosystem	Year Proposed	Name	Estimated Cost per Device	Time per Defeat
DES	1977	unnamed [77]	$84.38M	12 hours
DES	1993	unnamed [420]	$1.73M	3.5 hours
DES	1998	DES Cracker [155]	$0.38M	3 days
RSA-512	1999	TWINKLE [366]	$1.50M	9-10 weeks [375]
RSA-512	2003	TWIRL [367]	$2.01M	10 minutes
RSA-1024	2003	TWIRL [367]	$13.58M	1 year
RSA-1024	2005	SHARK [124]	$258.86M	1 year

Table 2.5: **Proposed Special-Purpose Defeat Hardware**—A sampling of proposed custom hardware for various cryptosystems at various points. All costs are in 2017 dollars and with the exception of DES Cracker are the original authors' estimates.

in depth and analyze not only their feasibility, but also the impact if they were actualized.

Guided Defeats Like asymmetric defeats, guided defeats leverage a one-time effort to reduce the cost of per-instance defeats. The difference is twofold the one-time investment for guided defeats is a direct enabling effort against the security protection. Guided defeats are also comparable to using policy influence to bound the cost of a defeat (Section 2.2.1), but guided defeats rely on surreptitiously doing so.

The most direct example of this is back doors in cryptographic standards. While not provably certain, many believe that the Dual EC Random Number Generator is a backdoored standard due to NSA generating the parameters. It is known that the parameters can be created in a manner that would also create a secret parameter which allows any entity with that knowledge to observe output of the RNG, reconstruct the RNG's internal state, and then predict future RNG output [32]. When used with TLS, the secret parameter is used to recreate the random number generator's state from the plaintext TLS handshake various values, predict the output used to create the key share, and derive session keys for content decryption [54]. NSA's one-time investment of resources [237] and "finesse" [332] towards standardization and implementation results in a small and parallelizable per-instance defeat cost [54].

Guided defeats may originate from many sources including standards (Dual EC), source code (Juniper's replaced Dual EC constants [53]), or hardware ("enabling for [redacted] encryption chips" [289]). The important distinction between an asymmetric defeat and a guided defeat is that a guided defeat is explicitly created by an NS-Attacker and an asymmetric defeat is discovered.

Person	Country	2017 Position	2007 Position
Donald Trump	U.S.	President	Reality TV Host
Angela Merkel	Germany	Chancellor	Chancellor
Theresa May	U.K.	Prime Minister	Shadow Leader of the House of Commons
Vladimir Putin	Russia	President	President
Xi Jinping	China	General Secretary	Member of Politburo Standing Committee
Hassan Rouhani	Iran	President	Member of Assembly of Experts

Table 2.6: **Retrospective World Leader Position**

2.3.4 Distant Return Horizons

If it's not exploitable now, that doesn't mean it won't be later

– NSA's OTP VPN Exploitation Team [308]

The last advantage of NS-Attackers that we'll discuss is the ability to invest resources into programs that may never bear fruit in the form of a strategic/tactical advantage or actionable intelligence. This is similar to standard military endeavors such as nuclear weapons which are expensive in both finances, manpower, and resources to research, build, maintain, protect, and destroy but in the best-case scenario will not be used for their intended purpose.

This NS-Attacker advantage is not purely based on the number of months or years between initial investment and fruition. While NS-Attackers are known to devote resources to preparing an action before it is conducted and even intentionally delay an operation until an opportunity presents itself [192], these are both available to other types of attackers. Network and personnel reconnaissance is a well known technique used by script-kiddies and highly talented hackers alike. An intelligent attacker can even gain access to a network, plant backdoors, and then go dormant for months before returning to profit from their efforts.

One example of this is the practice of storing currently undefeatable (e.g. strongly encrypted) ciphertext for substantial amounts of time. Without the use of forward secrecy in TLS connections, a passively acquired TLS connection can be decrypted by any attacker who gains access to the long-term SSL private key after the fact. As shown in Table 2.5 and as we'll discuss in Chapter 4, well resourced attackers can achieve feasibility long before the general public. While most SSL certificates have moved to 2048-bit RSA moduli [84], 88.35% of TLS connections in 2007 used a 1024-bit RSA modulus [212] and would be decryptable once the RSA public key was factored. As far back as 2000, 25% of SSL certificates surveyed were 512 bit or less RSA keys [212] which is now defeatable by anyone with $75 and access to Amazon EC2 [410].

For NS-Attackers, their inherent longevity makes these types of attacks feasible as well as useful. Acquiring ciphertext without a usable defeat does not provide immediate actionable intelligence, but it does provide a long-term retrospection window into previous events and behaviors. An NS-Attacker who acquires ciphertext from an adversary's embassy, stores it for 10-20 years, and then defeats its protections learns a information about behavior patterns and individuals' thought process.

Table 2.6 shows the current major world leaders and their position 10 years ago. With the exception of Donald Trump, all were high-ranking officials in important positions within their respective countries and would likely have been targeted for acquisition and monitoring. If their communications from 10 years ago becomes decryptable today, it provides insights into that person's personality and decision making process that are useful in current and future interactions and conflicts.

2.4 Constraints of a Nation State Attacker

While the advantages of being an NS-Attacker are numerous, there are also distinct constraints on NS-Attacker actions. In this section, we identify four of these constraints, discuss how they can impact NS-Attacker actions, and outline steps an NS-Attacker may take to mitigate these constraints. The four constraints we will discuss are: *Scale*, *Human Capital*, *Oversight*, and *Required Hard Targets*.

2.4.1 Scale

> *We live in an Information Age when we have massive reserves of information and don't have the capability to exploit it.*
>
> – 2011 Deputy Director for Analysis and Production at NSA [282]

One disadvantage of NS-Attackers is the sheer number and diversity of threats and potential victims that they must account for. Terrorists, drug dealers, kidnappers, and child pornographers are often cited as the "Four Horsemen of the Information Apocalypse" [358] that governments (and their associated intelligence agencies) are charged with monitoring. In addition to these, nation-states must be vigilant of other nation-states who wish to monitor and/or interfere with the country's internal operations. With these diverse set of actors and a multitude of possible communications channels that must be monitored, the scale becomes apparent.

This has created a goal of "collecting it all" [280] such that the fear of missing a useful data-point causes them to acquire all potentially useful data-points. By the end of 2011,

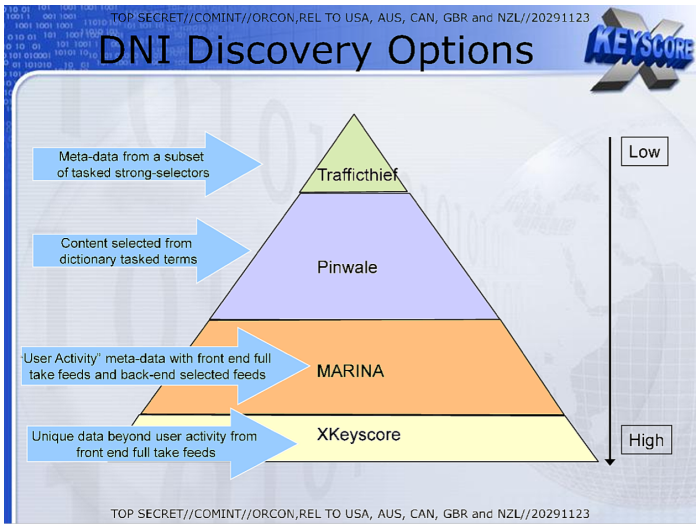

Figure 2.5: **Tiered Data Types**—The tiers of network traffic and the NSA systems which store and handle that traffic [425].

NSA expected to receive 1 TB of data per second from its various sources (discussed more below) [282]. One way to mitigate the challenges created by this large amount of information is to partition acquired data into multiple tiers of usefulness. Each tier is designed and used for different types of analysis. Because of this, the retention periods of the tiers can be tuned to account for the usefulness of the data and its quantity. Table 2.7 and Figure 2.5 show examples of these tiers found in the publicly available source documents.

Pertinent Data The highest tier is *pertinent* or "tasked" data which consists of data known to or most likely to contain useful information or actionable intelligence. In Figure 2.5, "Traffichief" and "Pinwale" are NSA's representation of this tier of data. An analyst can either "task" a selector for a specific victim to ensure that the acquired data is automatically promoted to this tier [304] or can manually promote data from the other tiers to ensure that it is retained [425]. In some cases, the presence/lack-of a tasked selector determines whether the communication is acquired in the first place [234]. In other cases, the data may already be acquired through bulk/full-take systems and pertinent data is simply marked for specific handling and routing [265].

By focusing on this data, analysts can ensure that they can effectively monitor these victims' communications for actionable intelligence. In addition to reducing the probability that a useful communication is not evaluated, this also allows analysts, to monitor these victims and extract higher-order information that may not be apparent from a single communication. Programs such as `TRAFFICTHIEF` and `SAM PEPYS` even provide real-time alerts to analysts when targeted victims are online and communicating [133, 403].

Data Type	Pertinent	Metadata	Indiscriminate
VoIP	NUCLEON [307]	FASCIA [259]	XKEYSCORE [307]
Telephony	NUCLEON [276]	FASCIA II [285]	SOMALGET [382]
General Internet Traffic	PINWALE [426]	MARINA [425]	XKEYSCORE [425]
VPN	PRESSUREWAVE [265]	TOYGRIPPE [308, 309]	XKEYSCORE [265, 308]

Table 2.7: **Tiered NSA Databases** — A sampling of databases NSA analysts use grouped by their type and tier of data.

Metadata While the full content of a communication is undoubtedly the most useful for actionable intelligence, the requirement of a priori knowledge of the victims and their associated selectors is an obvious limiting factor. To address this, the next tier of data is *metadata* or "data about data" (MARINA in Figure 2.5). Metadata consists of elements such as the endpoints of the communication, time of communication, and the like, but, loosely speaking, not contain the content of the communication.

While the acquired and stored metadata does not contain directly actionable intelligence (e.g. "Attack at Dawn"), it is highly useful for what's called "target development" [382] or "SIGINT development" [425]. Here, an analyst is not necessarily looking for actionable intelligence, but is instead searching for new or additional selectors for acquiring actionable intelligence.

A common way to do this is through "contact chaining" where an analyst is able to monitor the communications metadata between victims in a dataset [64, 285]. If many tasked victims are communicating directly or indirectly with a central victim, it may indicate that the central victim would be of use in gaining actionable intelligence. This same methodology can be applied to location metadata and an analysts can determine who was in the same physical area of a targeted victim, an indicator that they too may be a useful source to task and promote to pertinent data [278].

Indiscriminate Data The largest tier of data is from indiscriminate or "full-take" [139, 425] acquisition (XKeyscore in Figure 2.5). In this tier, data is acquired and processed with little consideration of the victim's identity. This type of data may be filtered for "low value traffic" [148] such as P2P downloads or publicly available streaming videos[10], but removal of this sort does not affect its label of indiscriminate acquisition.

Compared to the tasked and metadata tiers, indiscriminate data dwarfs them in pure quantity of information and is vastly overshadowed by them for usable information. But like metadata, indiscriminate data does have valid use cases from an analyst's perspective,

[10]The access to this type of data is likely recorded, but the storage costs of saving the same YouTube or Netflix video for each victim who watched it is likely too high to be practical.

which we will discuss more in-depth below.

From the above description, a reader may appropriately ask why pertinent and metadata are partitioned from indiscriminate rather than being sub-tiers of indiscriminate data. The reason for this is that the indiscriminate data tier is *not* intended to be a long-term monitoring tool. The sheer amount of indiscriminate data fundamentally constrains how long it can be stored. The actual retention time depends on many factors including the storage location and date of the source document. One available datapoint from 2009 states that indiscriminate data is stored for between 1–5 days, metadata for 30 days, and tasked data can be copied to "any other database for longer retention" [425].

Opportunistic Data The final tier of data is distinct from the pertinent, metadata, and indiscriminate tiers in that its purpose is explicitly not to provide actionable intelligence or allow target development. Instead, this type of data which is exceedingly difficult to acquire on-demand, but is relatively simple to acquire when it is available. The obvious logic process for opportunistic data is effectively "we might not needed this... but we may". This tier consists of two types: secrets and measurement data.

Secrets are those which may be disclosed inadvertently by the victim but are not necessarily useful at the time of acquisition. NSA's `DISCOROUTE` tool recognizes and stores router configurations [269] which, among other things, can contain a VPN Pre-Shared Key. `HYDROCASTLE` stores "802.11 [WiFi] configuration data extracted from CNE activity" [273] and likely includes WiFi PSKs and passwords. Either of these would be useful for defeating encryption if needed, but may require an intrusion or other significant resource expenditure to obtain if not acquired opportunistically.

The other type of opportunistic data is measurement data which is colloquially described as "data about data about data" as it is high-level, low granularity metrics. Measurement data includes information about cryptographic parameter usage [71] or trending technologies [288] but is specifically not a victim's information other than what tools/encryption/etc. that victim is using. While some may consider it metadata, the use case is to provide a basis for resource allocation and planning instead of target development and actionable intelligence. As we'll discuss shortly, the planning and resource allocation decisions are important for NS-Attackers to maintain the effectiveness.

2.4.2 Human Capital

> *Special people. And rare. One day I was returning from a meeting at Langley and my security detail looked a little impatient, as they had to wait for a stream*

of pedestrian traffic flowing outward from the NSA headquarters building. As they inched the SUV forward, I only half in jest cautioned, "Be careful. They could be linguists."

– General Hayden, former Director of the NSA [168]

While NS-Attackers' Scale constraint is fairly obvious, the Human Capital constraint is less so given the Specialization advantage is in large part due to the size of their workforce. Although NS-Attackers' employees are able to specialize in a relatively small portions of the entire organization, the employee manpower and skill sets are still a finite resource. To put it simply, an employee can be the number one cryptographer/linguist/analyst/engineer in the world, but there is a limit to the number of defeats/victims/systems that person can work on at a time.

Former NSA Director General Michael Hayden describes this constraint in depth with regard to linguists. The linguistic manpower needed to effectively monitor even a single important victim is immense. Every phone call, text message, e-mail, and web-search must be translated before an analyst can determine whether it is contains useful information [168]. NS-Attackers can take three approaches to mitigate this constraint: *deliberate allocation*, *automation*, and *contracting*.

Deliberate Allocation The first way to mitigate this constraint is for NS-Attackers to use manpower and skill-sets efficiently by shrewdly selecting which of the many possible objectives should take precedence. At a high level, this comes in the form of explicit strategic plans [290, 296]. These documents lay out the specific countries and topics which are deemed to be worth the resource investment as well as those of interest but are an "Accepted Risk" for which resources should not be allocated[11].

This approach filters down to the tactical level when deciding protocols, applications, and technologies to invest employee time and energy into. The specific decision are made by monitoring the different tools and technologies used by victims and ensuring that any trends or shifts are noticed. An example of this is shown in Figure 2.6 where the users of and current capabilities against those technologies are used to determine priorities with conventional risk management techniques [288].

The tiered data approach described above also accounts for this. The pertinent tier is highly likely to be useful and therefore likely to be worth allocating disk space, analytic

[11] As an example, the sale of conventional weapons by "gray arms dealers" is identified as topic for which NSA resources should be allocated but less so than monitoring the safety of Russian nuclear weapons where as the state of Egypt's ballistic missile program is a topic of strategic significance but will not be emphasized by US SIGINT systems and will knowingly suffer from intelligence gaps [296].

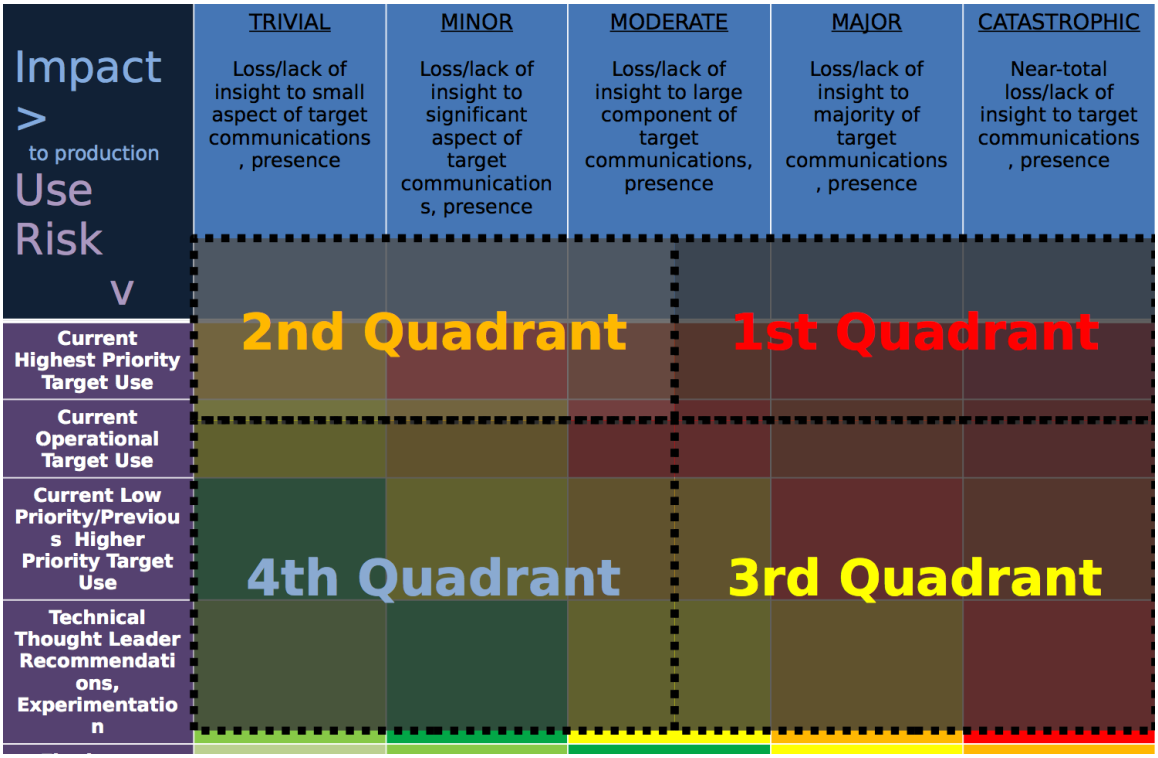

Figure 2.6: **NSA's Technology Risk Matrix**—NSA's risk analysis matrix used to prioritize technologies for resource expenditure [288].

effort, and defeat resources for whereas the indiscriminate tier is less likely. This is especially applicable to encryption schemes which have a known but expensive defeat. The resources to carry out the defeat are much more likely to be invested against pertinent data than a randomly selected indiscriminate communication.

Automation The second approach to accommodating the human capital constraint is to automate as many tasks possible. This "[l]iberates operators for high-order tasks" [270] so that manpower is used more effectively at the individual level. In order to address the time and skill requirements of our linguist example above, NSA has invested in machine translation of text documents [279, 355] as well as speech [145].

In addition to fully automated solutions, machine learning is used to surface data which is more likely to be worth an employee's attention. `FORESTPROWL` ranks audio files for transcription based on what is likely to be the most useful [276]. Locating potentially useful data from the sea of indiscriminate and metadata is also an improvement for finding victims when they change devices [268] or when their behavior is divergent from the norm [293].

Outsourcing A third method of addressing the Human Capital constraint is through outsourcing. This can come in the form of hiring contractors, coordinating with non-SIGINT entities [282], or partnering with other NS-Attackers (Section 2.2.3). In this way, the NS-Attacker places the onus for manpower and/or skill sets onto another entity.

A simple example of this is the use of linguistic contractors who work for the NSA but are employed by a contracting company. In in some arrangements, an NS-Attacker can contract-in such that the contract company finds the required skill set and that linguist then works alongside direct NS-Attacker linguists and analysts [117]. Alternatively, arrangements can contracting-out such that the NS-Attacker may send documents, including classified documents, to a contracting company and receive back the translated text. The National Virtual Translation Center is an interesting example of this type of outsourcing as it operates similar to Amazon Mechanical Turk's approach to 'gig economy' human capital [168].

While contracting is a mitigation to Human Capital constraints, it may be a self-defeating response. General Hayden acknowledges that while effective, reliance on contractors creates a counter-productive set of incentives to the point that the CIA had become "a bit of a farm team for our contractors" where employees with the highly needed skill-sets would develop and practice those skills at the cost of the US government and then leave and join contractors for more money [168]. As of 2011, contractors made up approximately 20% of the U.S. Intelligence Community workforce [254].

2.4.3 Oversight

> *[T]he only way to interpret Section 215 [of the PATRIOT Act] in that fashion [to allow collection of nearly all domestic records] is to add words to the statute that it does not contain, subtract words that it does contain, and reinterpret other words beyond recognition*
>
> – Privacy and Civil Liberties Oversight Board [235]

Another constraint that NS-Attackers must account for is the presence of oversight or approval entity. This usually comes in the form of judicial, legislative, or executive oversight body who approve the general details of the NS-Attacker's actions and are charged with ensuring their compliance with those rules. While other type of attackers must account for only punitive entities such as law enforcement and criminal courts for their actions, NS-Attackers must comply with their oversight bodies' regulations on how to carry out actions. In Section 2.2, we discussed that NS-Attackers had the ability to influence policy through the characteristic of sovereignty, but that does not mean that they have the ability to dictate policy to their will.

While much of this oversight and compliance occurs outside of the public eye, there has been a substantial increase in the number of rulings and opinions regarding the U.S. intelligence agencies available due to the actions of whistleblowers and civil liberties organizations. Because of this, we primarily focus on the U.S. intelligence community's oversight by the Foreign Intelligence Surveillance Court (FISC) and legislative entities. While judicial, legislative, and cultural differences prevent us from applying the U.S.'s oversight regime (and therefore NSA's mitigations) directly to other NS-Attackers, it does expose a set of behaviors that are likely available for use by others.

For all the privileges and abilities that the U.S. Government has given to NSA and other intelligence agencies through legislation such as the USA PATRIOT Act, the FISA Amendments Act of 2008, and the USA Freedom Act, those same laws also place restrictions on the intelligence agencies with regard to how those abilities can be used. The FISC and various legislative bodies ensure that the intelligence agencies' tactics and behavior comply with U.S. law and can prompt changes to the way NSA acquires, stores, and queries data.

Interpretation One way NS-Attackers can mitigate the effects of oversight on their operations is by portraying the intent of policy in a way favorable to their wishes. The text of laws rarely includes detailed descriptions of methods and technical specifications and instead provides guidance and intent. These texts can be massaged, equivocated, and mixed with various legal opinions and law texts to argue that a specific NS-Attacker technique/methodology is supported by statutes.

One example of this is the NSA's telephony metadata program in which call records (from, to, etc. but not content) were acquired in large numbers for data mining and call-chaining. Originally created as the "President's Surveillance Program"[12] in 2001 under executive order, it was for domestic-to-foreign phone calls where at least one participant was a non-U.S. person [168]. After disagreements with the Department of Justice in 2004, the program was transitioned to operate under FISC orders justified by Section 215 of the PATRIOT Act in 2006 as "business records". With this transition, the program expanded to explicitly include domestic-to-domestic call records[13] [314].

NSA then determined that the FISC rules on querying the metadata database only applied to "archived data" and not data acquired and in the process of being archived. This interpretations was later discovered and corrected [235]. NSA then used the "reenactment doctrine" after Congress renewed Section 215 in 2010 and 2011 to justify that Congress had adopted their interpretation used for the program [235]. In 2015, the USA Freedom Act

[12] Also known as the `STELLARWIND` program.

[13] It is thought that this expansion occurred earlier but this is uncertain from the available documents.

substantially altered the way that NSA acquires call records [221] but they still acquire and store large numbers of call records for analysis [313].

Stove-piped Knowledge Another approach to mitigating oversight is to control who has access to the existence, justification, and implementation of specific programs. Even among those who have access, NS-Attackers can control the extent of understanding the various oversight bodies have. Just as public awareness and understanding of NS-Attacker operations is often near zero due to classification and operational constraints, this technique is also applicable to oversight bodies.

The NSA's telephony metadata program appears to have relied heavily on this mitigation from its inception. Initially, the number of people who knew about its existence to an absolute minimum. From 2001-2003, only three people from the Department of Justice knew of its existence.[14] Even NSA's own Inspector General did not gain access until late 2002 [314]. Between 2001 and 2007, only 60 members of Congress and just over 3,000 people total (including NSA analysts and other employees) were ever cleared for the program [315].[15]

Even amoung those cleared for access, information and understanding is not spread evenly across all parties. In some cases, this is done by providing vague, form-letter justifications [300]. In other cases, it is accomplished by misrepresenting technical and implementation details to oversight bodies [120] or by overstating the usefulness of a program to disincentivize its alteration [424].

2.4.4 Required Hard Victims

The last constraint of NS-Attackers that we discuss is the requirement of operating against exceedingly difficult victims. Whereas many attackers focus on "low hanging fruit" in order to avoid investing time, energy, and resources into attacking well-defended victims. For many cases, poorly-defended victims provide the same result the attacker is interested in. The Mirai botnet, for instance, recruits new nodes largely by searching for default and common passwords [15]. The botmaster does not care who owned a newly infected node or what network it was attached to because of the small impact those factors play in participating in attacks. NS-Attackers, on the other hand, are tasked with monitoring specific victims (or types of victims) and acquiring specific information. These targets may

[14] One of these official was OLC Deputy Assistant Attorney General John Woo whose approach to executive authority has been described as "Article 2 über alles" [168] in a barely veiled comparison to Nazi Germany's use of "Deutschland über alles".

[15] For comparison, as of 2013, approximately 1.26M persons held Top Secret U.S. clearances [59].

be hard for one of two reasons: A) Nation-State Defenders and B) latent indicators.

Nation-State Defenders (NS-Defenders) are hard victims because they are defended by an entity with nation-state resources. While we do not discuss NS-Defenders within this dissertation, it is a safe to assume that nation-states are more capable of defending against all types of attackers, including NS-Attackers, than lesser victims. Detection and prevention could be through discovery of vulnerabilities via acquisition and monitoring [275], discovery of vulnerabilities through their own efforts [149], or by the use of non-standard algorithms [297] or equipment [80].

Latent indicators, on the other hand, make for hard victims not because it is difficult to act against them, but because it is difficult to identify them. The small-cell and "lone-wolf" nature of many current threats offers examples. One aspect of these type of victims is that they are often obvious in hindsight. NS-Attackers and law enforcement are often confronted with questions of why they did not stop a particular incident when the information existed to put the pieces together, but the connections were not made [168, 197].

Collect-it-all In order to find hard victims, some NS-Attackers adopt a "collect it all"[16] strategy as referenced above with regard to both metadata and indiscriminate data. Metadata allows analysts to trace social networks and identify new victims through contact chaining. Indiscriminate data allows analysts to find new victims based on the contents of their communications and to locate actionable intelligence from previously unknown victims. Buffer systems such as `XKEYSCORE` [226] and `TEMPORA` [148] are an example of this. Information acquired from many different sources is funneled into the buffer's databases and can be queried based on IP address, URLs, cookies, search terms, attached filenames, or numerous fields [257].

In addition, large data repositories are well suited for machine learning and similar automated analytic techniques. Machine learning is known to be used for identifying the behavior patterns of couriers [294] and detecting social structure changes and automatic data visualization [137]. Machine learning is well suited to training on large datasets and then classifying future inputs. While imbalanced datasets are a difficult problem, they are not an insurmountable one. These large datasets also improve NS-Attackers' ability to monitor victims protected by NS-Defenders. Instead of finding victims, the collect-it-all mentality helps NS-Attackers by increase the likelihood that any leak of information is acquired and available for exploitation. A victim employed by a government is likely to be well protected by their NS-Defender resources while at work, but if they are brushing up on public information for a meeting the next day by reading Wikipedia pages or standards from

[16] In our lexicon, this would be "acquire it all", but we will use the phrase as it is well known.

a residential or mobile network. By acquiring all data from those networks, NS-Attacker can identify the victim from other traffic and then monitor what they were viewing.

Retrospective Analysis Another mitigation to the Required Hard Targets constraint is the use of retrospective analysis in addition to prospective analysis. The difference between prospective and retrospective analysis is that prospective analysis is intended to monitor victims and provide information about events before they occur. Retrospective analysis, on the other hand, is meant to provide clarity to and initial leads for events after they occur. Whether a newly identified lone-wolf, a newly hired NS-Attacker employee, or in the aftermath of an unexpected event, historical actions and communications from previously acquired data are useful in extracting intelligence.

An example of this is an Improvised Explosive Device (IED) detonated remotely via a cellphone [365]. It is infeasible to classify an arbitrary call to an arbitrary cellphone as a detonation signal or not, especially in real-time. But after detonation, it is possible to determine the caller and callee from the call records. From there, the initiating handset can be targeted as it is now associated with a targeted hard victim. Similarly, retrospective analysis is useful for pruning investigations and allowing the NS-Attacker to focus on known-unknowns rather than unknown-unknowns.

2.5 Differences between Nation State Attackers

While the previous sections describe characteristics, advantages, and constraints common to all NS-Attackers, it would be improper to assume that NS-Attackers are a homogeneous group. The challenge in attempting to differentiate between sub-classes of NS-Attackers is that the publicly available information is heavily skewed towards NSA and other Five-Eyes organizations with a much smaller amount of information available on others. Even with this constraint, we observe three aspects that may indicate sub-categories of NS-Attackers: treatment of domestic victims, tolerance of public exposure, and delegating operations.

2.5.1 Treatment of Domestic Victims

An NS-Attacker's treatment of domestic victims appears to depend largely on the country's policy regarding its citizens' freedom and privacy. While we will leave the classification of a country's stance and relative attitude towards these issues largely up to other researchers, we do observe that NS-Attackers' large-scale censorship operations appear to provide an adequate proxy for their policies towards domestic victims. Unlike most large-scale

acquisition actions, censorship is largely an active action that can be both induced and measured. Because of this, a large amount of research exists measuring censorship tactics such as DNS manipulation [329], flow disruption [91], or active interrogation [90].

Three countries which are known to engage in widespread censorship as well as widespread internal NS-Attacker operations are China, Iran, and Kazakhstan. China reportedly monitors in- and out-bound calls in real-time, as well as the geolocation of cellphones in order to provide 'public safety' information [127]. Iran is actively working towards widespread monitoring of communications and ostensibly social networking usage, though reports differ about their current capabilities [122]. Kazakhstan even went so far as to require ISPs to Man-in-the-Middle TLS connections with a government issued root CA [196]. Stepping back, potentially useful proxy is less surprising given that the technology and tactics needed to perform widespread censorship is very similar to the technology needed to perform widespread domestic acquisition operations.

Even countries whose NS-Attacker entities are more respectful of citizens' personal freedom and privacy, actions are taken against domestic victims. While the U.S. is known to acquire large amounts of information about untargeted victims, the majority of efforts are focused on foreign communications. Obvious and well known counter-examples such as the now defunct Section 215 [235] telephony metadata program and the President's Security Program [314] for Internet metadata exist, but the vast majority of intentionally domestic actions appear to be much more targeted. Instead, the NSA focuses on acquiring the communications of non-U.S. persons and takes measures to reduce the possible impact on U.S. persons' privacy. These measures include "minimizing" data to U.S. persons' identifiers as well as segregating known U.S. persons' acquired data from other victims' data [175].[17]

2.5.2 Acceptability of Public Attribution

All NS-Attackers are secretive and attempt to avoid detection and attribution to prevent victims from avoiding their acquisition and monitoring operations. Quantifying this behavior pattern is difficult as attribution is challenging and more so with NS-Attackers because of their Near Superset Attacker advantage.[18] Further complications arise from known connections between NS-Defenders and public organizations that commonly discover and

[17] While this approach is far from ideal and is known to provide fewer protections than advertised [19], it is nonetheless a better scenario than the blanket acquisition, long-term storage, and monitoring seen in other countries.

[18] As discussed in Section 2.3.1, NS-Attacker are cognizant of their role and explicitly attempt to mimic other types of attackers.

attribute NS-Attacker operations [245, 345].[19]

Even with these constraints, we observe that different NS-Attackers have different views on public attribution of their operations. On one side, we see a cavalier attitude of "we're going to do X and if you find out, then we'll just keep doing X until it stops working". To be clear, this is not an indication of ability or competence to conduct discreet operations, but a decision that the penalty for detection and attribution is not worth the effort to avoid. These NS-Attackers are much more likely to conduct aggressive intrusions against large numbers of victims including corporate and other non-governmental entities. This type of behavior is seen from China (Operation Aurora [430], Operation Shady RAT [393]), North Korea (Sony Pictures breach [353], WannaCry [246]), and at times Russia (discussed below).

On the other side, we see a calculating and subtle approach to operations. Compared to the cavalier attitude described above, these NS-Attackers take a "we're going to do X, but you'll never know" approach. Because of this, the vast majority of known operations come from whistleblowers (e.g. Snowden Documents) or inadvertent disclosure (e.g. Shadow Brokers [165]). Operations discovered and attributed to group of NS-Attackers suggest an enormous amount of effort investment (e.g. Stuxnet [116]). This type of behavior is seen largely from the Five-Eyes entities (U.S., U.K., Canada, Australia, and New Zealand). The Snowden Documents include explicit risk-management processes that support this perspective [141].

2.5.3 Delegating Operations

Another difference we observe in NS-Attackers is their willingness to delegate operations to external entities. To be clear, we are not discussing the use of publicly available assets as described in Section 2.3.1 nor are we discussing outsourcing mitigations described in Section 2.4.2. In those cases, NS-Attackers are importing the product and using it themselves for their own purposes. With delegated operations, NS-Attackers export their characteristics, e.g. giving access or sovereignty, to a non-NS-Attacker for use in achieving the NS-Attacker's objectives. In our observations, we see three types of behavior: "in-house" operations, delegation to "cyber militias", and delegation to "cyber mercenaries".

In-House Delegation The first type comprises NS-Attackers who prefer to keep their operations "in-house" and coordinate or delegate operations mainly with other NS-Attackers. As described in Section 2.2.3, NS-Attackers commonly cooperate with NS-Attackers of

[19] An operation discovered, disclosed, and attributed to an NS-Attacker by a public entity such as Fireeye or Kaspersky Lab may or may not have been discovered by an NS-Defender using NS-Defender resources and the disclosure is laundered through the public entity.

'friendly' countries to mutually enhance their positions by sharing acquired data or technology. In this case, the NS-Attackers are not so much exporting their own characteristics as they are exchanging the products of their own characteristics with each other. These exchanges commonly come in the form of formal agreements that describe the parameters and constraints of the exchanges [261, 271, 281].

The advantage of inter-NS-Attacker coordination is that secrecy can be maintained. Tighter control of the actors improves the ability to limit and compartmentalize the effects of an operation. The disadvantage of this approach is that the NS-Attacker is limited by their own Human Capital constraint as well as that of the partner(s). Unsurprisingly, many of the NS-Attackers who are risk averse with regard to public exposure are also averse to delegating operations.

Cyber Militias The second type of delegation comprises NS-Attackers who delegate operations to loosely organized public actors often, with assurances of little or no punishment for otherwise criminal acts. In a way, this exports their characteristic of Sovereignty to the public actor. These public actors are often individual hackers or hacker collectives which act as a "cyber militia" [378] similar to traditional militias. The NS-Attacker operations are then carried out either through a direct order or through veiled persuasion. While toleration of criminal activity is a known issue in international politics, we will focus on the direct NS-Attacker operations by non-NS-Attackers.

China and Russia are thought to use this method extensively throughout various international incidents. Large-scale website defacements in the immediate aftermath of the 2001 collision of Chinese and American military planes have been attributed to encouragement from the Chinese government. The DDoS attacks against Estonia in 2007 and Georgia in 2008 were largely conducted by members of the Nashi, a Russian pro-Putin youth movement headed by an assistant to the Russian parliamentary leader [378].[20]

By delegating to these militias, the NS-Attacker is able to plead ignorance with regard to the attack while still accomplishing its goal. Additionally, some countries may use these militias as a way to recruit manpower and skill sets which they need for internal operations. The disadvantage, though, is that the NS-Attacker does not have complete control of the militia that they have trained and equipped. While once a tolerated side-effect, some countries have begun to crackdown on these types of groups due to the negative effects of their non-sanctioned operations [378].

[20] We should note that while Russia and China are the most obvious use of cyber militias, they are not the only ones. The Syrian Electronic Army and Iranian Cyber Army are thought to be sanctioned by their respective governments [378], but less is known so we do not focus on them.

Cyber Mercenaries The third type of delegation is NS-Attackers who delegate operations and their characteristics to a non-governmental entity (usually a company). As discussed above, this is not an NS-Attacker buying a product from another entity (e.g. a defense contractor) to incorporate into an internal system, this is an NS-Attacker either importing an entire, finished surveillance product or exporting the authorization to conduct operations and receiving back the final intelligence. In this way, the entities act as "cyber mercenaries" analogous to how traditional mercenaries are used in kinetic warfare.

Companies such as Hacking Team and Gamma International sell software and infrastructures for targeted acquisition, intrusion, defeat, and monitoring out of the box [421]. Bull/Amesys, Trovicor, and Shoghi offer large-scale acquisition and monitoring systems as a complete package to be installed in network choke points [370, 421]. Companies like Geofeedia, MediaSonar, and X1 Social Discovery acquire vast amounts of data from social media websites and simply sell access to their data repositories and monitoring tools [323].

Cyber mercenaries lower the required resource investments as well as barriers to entry. A single vulnerability can be used by the company in support of multiple NS-Attackers' goal thereby sharing the resource costs across all participating NS-Attackers. Similarly, the skill-sets and manpower required to develop large-scale acquisition, storage, and retrieval systems can be similarly shared. It is likely a consequence of this that the use of cyber mercenaries is largely by countries without significant technical economies or are in sudden and immediate need of monitoring. Bahrain, Morocco, and the United Arab Emirates have been found to use mercenaries' malware against journalists whereas Libya, Iran, and Yemen purchased and used the large-scale acquisition systems [421].

2.6 Accounting for NS-Attackers

As seen in many the source documents, all people and groups are potential NS-Attackers victims. Terrorists [284], activists [166], corporations [267], and other governments of both the friendly [283] and less-than-friendly [286] variety are known to be victims of NS-Attackers. Even without actual monitoring, the threat of surveillance is thought to affect individuals' behavior and cause self-censorship of lawful actions [330]. Because of this, it is important to discuss how NS-Attackers affect threat models as well as approaches that defenders and researchers can use combat those affects.

To be clear, the major take-away from our research and analysis is that preventing an NS-Attacker from gaining access to a network, devices, or communications is likely a Sisyphean effort. If a sufficiently advanced and well-resourced NS-Attacker is able to find a targeted victim and believes that victim is worth the resource investment, there is

little that can prevent the NS-Attacker's eventual success.[21] By their nature, software and hardware have bugs and some of these bugs will be exploitable. The networks, devices, and communications of relatives, friends, and co-workers are possible sources of information about the targeted victim. The victim's external dependencies (ISP, software vendor, supply-chain) may be used to gain access to the victim's information. Even if the targeted victim maintains a near-zero personal digital footprint, surveillance cameras, third-party records, or even the lack of a footprint may be used to acquire information about and monitor the victim.[22]

That being said, there are steps that potential victims can take to impede NS-Attackers' efforts. While these do not block NS-Attackers, they do "raise the bar" of required effort. As discussed in Section 2.4, the constraints of NS-Attackers require not only dedication to specific targets, but also disincentivize NS-Attackers investing resources in arbitrary victims. In this section, we identify attributes of NS-Attacker which potential victims and security researchers can use when threat modeling. The intention of this section is explicitly not to provide an outline of "how to avoid law enforcement" or "how to beat NSA", but to describe concepts that potential victims may find insightful and expose additional concerns they should be cognizant of with respect to NS-Attackers. Specifically, we will discuss accounting for NS-Attackers from the perspective of individuals, organizations , and researchers.

2.6.1 Individual Perspective

One improvement that an individual can make is to follow the common "Best Practices" for protecting against criminals and other non-NS-Attacker actors. Enumeration and discussion of practices such as strong passwords, HTTPS usage, and updating software are widely available and we omit their details. With the advantage of being a near-superset attacker, protecting against non-NS-Attackers also protects against capabilities of NS-Attackers. Specific to NS-Attackers, there are two concrete steps that individuals can use to improve their security posture: "tool upgrades" and "cloud aversion".

[21] The long delay in catching Osama bin Laden was not cause by the difficulty tracking and monitoring him; it was finding him in the first place. Once his location was determined, monitoring was relatively straight-forward [255].

[22] In the often hyerbolic but insightful words of James Mickens, if you threat is "Mossad doing Mossad things with your email account", then your possible solutions are "Magical amulets?", "Fake your own death, move into a submarine?", and "YOU'RE STILL GONNA BE MOSSAD'ED UPON [*sic*]" [238].

Tool Upgrades With regard to tool upgrades, the concept that users must be cognizant of is that there are thresholds for which an action's cost is acceptable and unacceptable.[23] As described in Section 2.4, each action an NS-Attacker conducts, whether acquiring communications or searching a database, has a cost in the form of resources such as time, manpower, or computation. Raising the cost of those actions decreases the likelihood that they will be undertaken without a sufficient incentive. Upgrading tools is the concerted awareness of and choice to use more secure software when available for little or no negative impacts.

One example of this is the choice of which instant-message application to use when communicating with friends, associates, and others. Many different services and protocols are available with varying threat models. These threat models vary from the "anyone can listen" model of IRC, forums, and chat-rooms to the "anyone on-path can listen" of SMS and non-end-to-end encrypted chat apps, to the "only endpoints can listen" of Signal or WhatsApp. Moving from SMS or Facebook Messenger to Signal forces NS-Attackers to expend more resources and effort to monitor a victim. To the best of our knowledge, the best defeat of strong end-to-end encrypted communications protocols such as Signal is through the use of an enabling intrusion on the endpoints. Even then, an NS-Attacker must detection of an active intrusion and disclosure of a vulnerability to acquire communications, decreasing their likelihood of doing so unless they are sufficiently interested in the victim.[24]

Cloud Aversion The second improvement is by re-evaluating the ubiquitous presence of cloud services in everyday life; especially when local implementations are available. A common, but not necessarily precise, idiom is "there is no cloud, just someone else's computer". While imperfect, it sufficiently conveys the risk of cloud providers. When individuals use remote services to store or process data, they are also giving up large amounts of control with regard to the security of that data.

One obvious example is the use of cloud backup services such as Carbonite [46] instead of backing up to a local network drive. The use of a cloud provider exposes an individual to the risk of an NS-Attacker gaining access to information from the provider through cooperation, legal writs, and/or enabling attacks. In many cases, cloud providers do not consider themselves within the threat model for user data. This gives NS-Attackers the opportunity to leverage forced cooperation of the provider against the individual. One possible form of this attack is a cloud provider who proactively encrypts data at rest to

[23] We do not attempt to quantify these thresholds due to the lack of data-points, but the analogy is appropriate for thought experiments.

[24] Even those NS-Attackers who are less risk averse to public exposure still risk exposure of vulnerabilities which are thought to be difficult to obtain.

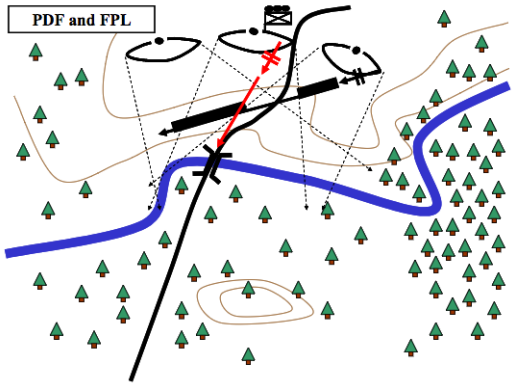

Figure 2.7: **Tactical Choke Point** — An example of a tactical choke point with one force stationed north of the river, and its opposition located across the river to the south [62].

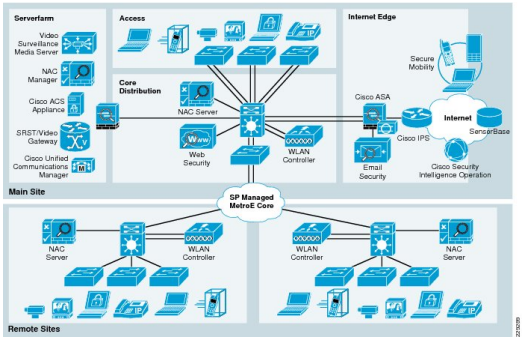

Figure 2.8: **Network Choke Point** — An example of network structure with an underlying choke point [57].

protect it against theft, but also controls the decryption key. Using influence or legal writs, an NS-Attacker could compel the provider to surrender the decryption key (à la LavaBit [336]) or to decrypt and surrender the plaintext (à la the Yarovaya law [130]).

While we encourage an aversion to cloud services, we do recognize that there are competing security advantages of cloud-based systems. The most compelling of these is that proactive cloud providers can dedicate resources to protect individuals' security and privacy. Most major cloud providers have large, talented, and dedicated security teams who are responsible for ensuring that their users' data is sufficiently protected from attackers. And while these efforts are not a panacea [331], are undoubtedly more of a barrier to NS-Attackers' action than the defenses of an average user. Because of these competing attributes, we do not advocate a blanket embargo of cloud services but a skeptical examination of and, when possible, aversion to the use of cloud services on the individual level.

2.6.2 Organizational Perspective

From an organization's perspective, there are also concepts specific to NS-Attackers that must be accounted for in their threat model. The use of Best Practices provides a base case identical to that of individuals, and we refer the reader to external sources for their details and discussion. For organizations, we identify two concepts for defenders: "choke points" and "value analysis".

Choke Points A "choke point" is a location of increased importance due to its impact on movement within the organization. While an odd comparison at first glance, Figures 2.7 and 2.8 show how these concepts are nearly identical with regard to analyzing a potential battlespace whether it is terrain or digital. In Figure 2.7, the river bisecting the map requires any north-south movement to traverse the single bridge where the opposing force can concentrate fire. In Figure 2.8, the central switch within the "Core Distribution" is traversed for nearly all traffic where an acquisition point would be the most effective. In both cases, movement of forces or information must traverse the choke point.

To an NS-Attacker, choke points are especially advantageous position not only because they allow the acquisition of large amounts of information from a single location, but also because they allow the NS-Attacker to compromise assumptions made by the defenders. In Figure 2.8, a presence on the choke point router allows attackers to avoid the internal "Web Security" device by simply not forwarding NS-Attacker traffic to it. When these type of vulnerable network locations are identified, there are two direct steps that defenders can take to improve their posture. The first is to restructure the network to avoid creating choke points, forcing an NS-Attacker to conduct multiple separate intrusions.[25] The second is to concentrate monitoring and defense on the choke points, which increases the likelihood that an NS-Attacker intrusion would be noticed.

Data Value Analysis The second concept important to defenders is to understand the value of the different types of information being stored by their organization. This element of a threat model would have been especially useful in the intrusion into the Office of Personnel Management (OPM). A central repository of detailed background investigations on nearly all persons who applied for security clearances from the U.S. Government is an attractive target for NS-Attackers for identifying undercover HUMINT assets as well as improve the recruitment of new HUMIT assets based on those who provided information adaptable for blackmail [208].

While few companies store information so directly and obviously useful to an NS-Attacker, it's hard to justify that any data would not be useful to an NS-Attacker. In order to address this, organizations can minimize the data that they collect and store from users. Not only does this reduce the incentive for NS-Attackers to target an organization for enabling operations, it also reduces the burden and impact of judicial writs on end user security and privacy [374].

[25] A choke point is also a single point of failure so restructuring also improves the network's resilience to disruption.

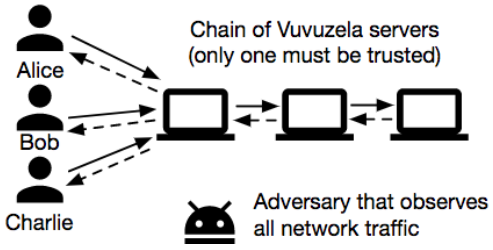

Figure 2.9: **Vuvuzela System Architecture**—The design of the Vuvuzela private messaging system [411].

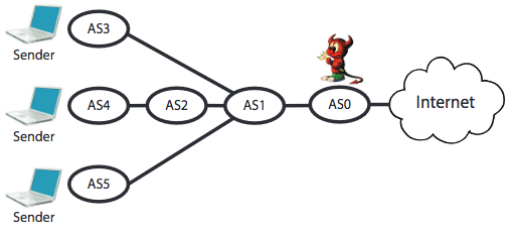

Figure 2.10: **Hornet System Architecture**—The design of the HORNET onion routing system [55].

2.6.3 Researcher Perspective

The fact that potential victims must alter their threat modeling to account for NS-Attackers is fairly straight forward, but it is important to address how security researchers and system designers must account for NS-Attackers. After all, if the security community can not provide potential victims with tools and techniques to make themselves more difficult to monitor, any improvement in an individual's security posture will be of minimal effectiveness. Here we discuss three concepts which researchers must be cognizant of: "system choke points" and potential harm.

System Choke Points Just as organizations must account for choke points within their network infrastructure, researchers must be cognizant of choke points in the design and deployment of systems. Distributed and federated system design has long been a goal of many security systems as a way to combat potential bad actors, but recent system proposals have eschewed this design goal in favor of improving usability and performance as seen by the user.

A recent example of this is the Vuvuzela private messaging system (Figure 2.9) which uses tunneled encryption and message shuffling to protect against an adversary who can observe all inter-participant traffic [411]. While a noble effort, the system requires a static set of n servers[26] to be used by all clients. These servers represent a choke point in system security as an NS-Attacker has a finite and unchanging set of hosts to which they must gain access to compromise the system's security. While defenses can be concentrated on these three systems, NS-Attackers' advantages of specialization and near-superset attacker status require a *infinitely powerful defender*[27] to prevent intrusion of each server.

[26] The authors propose three.

[27] A hypothetical person/organization that can defend against all types of attackers including NS-Attackers [118].

Along with choke points in system design, security researchers must also be aware of existing and likely choke points in deployment of systems. These deployment attributes may result in unexpected choke points that are advantageous to NS-Attackers. In Chapters 4 and 5, we discuss two examples: standardized Diffie-Hellman groups and TLS Session Ticket usage. The recent proposal of the HORNET onion routing system [55] also exemplifies how system design can lead to these types of choke points. HORNET is conceptually similar to the Tor network [79] in its use of tunneled encryption and multi-hop transportation, but HORNET makes the design choice to operate on network infrastructure as a platform to leverage the high bandwidth network connections.[28] This is in contrast to Tor's more general computing platform with the major use case being general purpose server installations.

Because of this design choice, the pool of possible HORNET node operators is heavily constrained to those entities with direct access to and ability to extend network infrastructure as opposed to Tor's operator pool of entities who are capable of running services on a public IP. This constrained operator pool is further restricted to those who are willing to endure the legal and administrative overhead required when participating in an anonymity network [123]. This logically results in two outcomes: 1) a small set of nodes participating which reduces the anonymity of the users and creates a choke point similar to Vuvuzela and 2) an increased likelihood that any user's path will traverse one or more nodes operated by law enforcement or NS-Attackers thereby fundamentally compromising many of the security guarantees.

To address these weaknesses, researchers and system designers should prioritize distributed systems and untrusted processing. Diversity of locations and operators should also be highly valued to remove choke points in design and operation. While the Tor network is not free of vulnerabilities when NS-Attackers are within the threat model [184], the diversity of nodes, locations, and operators removes many of the potential choke points of the network.

Potential Harm Finally, we consider ethics. While we do not wish to dive deeply into the ethics of NS-Attackers's known operations (from any side) we do believe it is important to discuss how NS-Attackers complicate ethical dilemmas. In most computer security research, the adversary is assumed to be be a villain (whether a criminal, hacktivist, or simply a mischievous actor) whose intent should be thwarted at all times, either by helping find and remedy weaknesses that are or may be exploited, or by building new systems to increase user protections Either way, the security researchers' end-goal is to stop their adversary without respect to their goal or victim.

[28] The authors' performance evaluation assume 10 Gbps links.

When NS-Attackers are the researchers' adversary[29], this is not as simple an assumption. NSA, GCHQ, CSE, FAPSI, and other NS-Attackers have an important and difficult duty to protect their respective countries and citizens which few would argue is conceptually unneeded or improper. The ethical dilemma for researchers is balancing the potential harm of publishing against the potential harm of not publishing.

In publishing, the potential harm is that the vulnerability or technique may be integral to operations against justifiable victims. Publishing alerts those victims that they are, or could be, monitored by an NS-Attacker thereby increasing the potential for harm against persons due to the NS-Attacker's inability to effectively monitor these justifiable victims. On the other hand, not publishing could expose unjustifiable victims to operations by criminals or other attackers thereby failing to decrease the potential harm to persons by leaving them vulnerable. Researchers must take a measured and careful approach to publishing known or plausible NS-Attacker operations or vulnerabilities.

Our approach to these dilemmas attempts to determine who would benefit more by publication and by what degree. A weakness in commonly used software or cryptography that makes all people less secure is likely to be published so that it can be patched, even though this may cause harm by decreasing the ability of NS-Attackers to monitor and thwart kinetic incidents.[30] A weakness that largely affects the security of justifiable victims and whose scope is limited to those justifiable victims will likely not be published, even though it may allow harm to unjustifiable victims by leaving them vulnerable. While these two examples represent extremes, there exists a wide array of possible scenarios between the two and of other constructions.[31] Most scenarios are significantly more complicated and involve many unknowns a researcher must account for.

[29] We intentionally use the term "adversary" and not "enemy".

[30] Disclosure policy in and of itself is a separate discussion which we believe is not heavily influenced by the presence or lack of an NS-Attackers.

[31] An especially difficult constructions is one where a researcher much decide whether to notify an NS-Attacker of an existing vulnerability whose disclosure could contribute to potential harm against justifiable victims.

CHAPTER 3

Impact on Computer Security

In Section 1.1, we briefly addressed how the SSL PKI's authentication mechanism as a defensive measure is fundamentally compromised when confronted with Nation-State Attackers (NS-Attackers). While a simple and direct example, it fails to show the impact that NS-Attackers have on threat models as a whole. In this chapter, we will present the findings from our Internet voting research into the 2013 Estonian Internet Voting System. By using our Nation-State Attacker model proposed in Section 2, we can see that the Estonian I-Voting systems is vulnerable to Nation-State Attackers who not only have the ability to interfere, but also the incentive.

Several countries have experimented with casting votes over the Internet, but no nation uses Internet voting for binding political elections to a larger degree than Estonia [190]. When Estonia introduced its online voting system in 2005, it became the first country to offer Internet voting nationally. Since then, it has used the system in local or national elections seven times, and, in the most recent election, over 30% of participating voters cast their ballots online [101]. People around the world look to Estonia's example, and some wonder why they can't vote online too [333].

Nevertheless, the system remains controversial. Many Estonians view Internet voting as a source of national pride, but one major political party has repeatedly called for it to be abandoned [113]. Although Estonia's Internet Voting Committee maintains that the system "is as reliable and secure as voting in [the] traditional way" [102], its security has been questioned by a variety of critics, including voices within the country (e.g. [219, 316]) and abroad (e.g. [376]). Despite these concerns, the system has not previously been subjected to a detailed independent security analysis.

For these reasons, the Estonian Internet voting (I-voting) system represents a unique and important case study in election security. Its strengths and weaknesses can inform other countries considering the adoption of online voting, as well as the design of future systems in research and practice.

In this study, we evaluate the system's security using a combination of observational and experimental techniques. We observed operations during the October 2013 local elections, conducted interviews with the system developers and election officials, assessed the software through source code inspection and reverse engineering, and performed tests on a reproduction of the complete system in our laboratory. Our findings suggest the system has serious procedural and architectural weaknesses that expose Estonia to the risk that attackers could undetectably alter the outcome of an election.

Most Internet voting schemes proposed in the research literature (e.g. [7, 52]) use cryptographic techniques to achieve a property called end-to-end (E2E) verifiability [51]. This means that anyone can confirm that the ballots have been counted accurately *without* having to trust that the computers or officials are behaving honestly. In contrast, Estonia's system is not E2E verifiable. It uses a conceptually simpler design at the cost of having to implicitly trust the integrity of voters' computers, server components, and the election staff.

Rather than proving integrity through technical means, Estonia relies on a complicated set of procedural controls, but these procedures are inadequate to achieve security or transparency. During our in-person observations and in reviewing official videos of the 2013 process, we noted deviations from procedure and serious lapses in operational security, which leave the system open to the possibility of attacks, fraud, and errors. Transparency measures, such as video recordings and published source code, were incomplete and insufficient to allow outside observers to establish the integrity of results.

The threats facing national elections have shifted significantly since the Estonian system was designed more than a decade ago. Cyberwarfare, once a largely hypothetical threat, has become a well documented reality [225, 354, 405, 406], and attacks by foreign states are now a credible threat to a national online voting system. As recently as May 2014, attackers linked to Russia targeted election infrastructure in Ukraine and briefly delayed vote counting [58]. Given that Estonia is an EU and NATO member that borders Russia, its threat model should not discount the possibility that a foreign power would interfere in its elections.

To test the feasibility of such attacks, we reproduced the I-voting system in a lab environment and played the role of a sophisticated attacker during a mock election. We were able to develop client-side attacks that silently steal votes on voters' own computers, bypassing safeguards such as the national ID smartcard system and smartphone verification app. We also demonstrate server-side attacks that target the implicitly trusted vote counting server. By introducing malware into this server, a foreign power or dishonest insider could alter votes between decryption and tabulation, shifting results in favor of the attacker's preferred candidate.

We conclude that there are multiple ways that state-level attackers, sophisticated online criminals, or dishonest insiders could successfully attack the Estonian I-voting system. Such an attacker could plausibly change votes, compromise the secret ballot, disrupt elections, or cast doubt on the integrity of results. These problems are difficult to mitigate, because they stem from basic architectural choices and fundamental limitations on the security and transparency that can be provided by procedural controls. For these reasons, we recommend that Estonia discontinue the I-voting system.

We returned to Estonia in May 2014 and shared these findings with election officials and the public. Unfortunately, government responses ranged from dismissive to absurd. The National Electoral Committee stated that the threat vectors we consider have already been adequately accounted for in the design, and that the attacks we describe are infeasible [108]. We disagree on both counts, but readers can review the evidence and reach their own conclusions. Prime Minister Taavi Rõivas and President Toomas Hendrik Ilves insinuated to the media that we had been bought off by a rival political party seeking to disparage the system. This we vehemently deny, but it illustrates how the Estonian public discourse concerning election technology has become dominated by partisanship. We hope that the country can separate technical reality from political rhetoric in time to avert a major attack.

3.1 Background

Our analysis focuses on the Estonian I-voting system as it was used for the 2013 municipal elections [104]. In these elections, Internet voting was available for seven days, from October 10–16, and the main in-person poll took place on October 20. Results were declared that evening. According to official statistics [101], 133,808 votes were cast online, corresponding to 21.2% of participating voters.[1] In this section, we review the design and operation of the I-voting system. Figure 3.3 gives an overview of the interactions between the main system components.

3.1.1 National ID Cards

An essential building block of the I-voting system is Estonia's national ID infrastructure [95], which plays a central role in the country's high-tech and e-government strategy [179]. Estonian national ID cards are smartcards with the ability to perform cryptographic functions. With the use of card readers and client software, Estonians can authenticate to websites (via

[1] Estonia used the system again, shortly after we made our findings public, for May 2014 European Parliament elections. There were only minor changes to the software and procedures. The fraction of votes cast online increased to 31%.

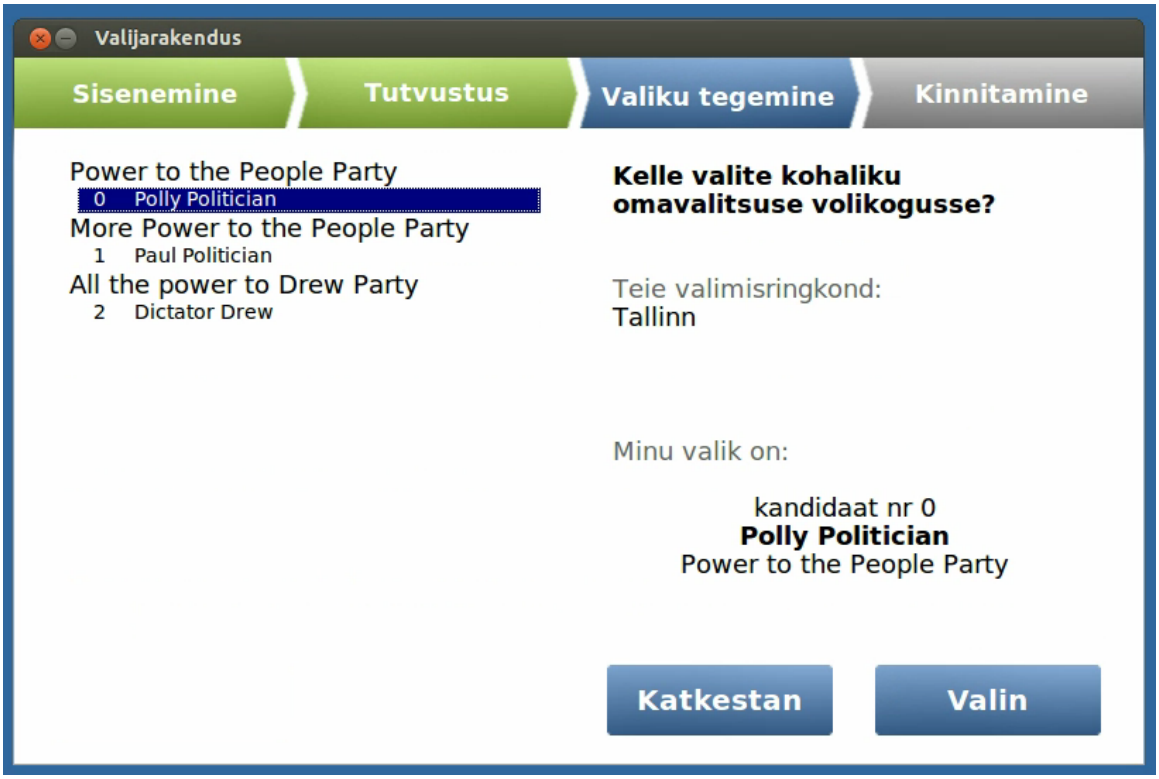

Figure 3.1: **I-voting client**—Estonians use special client software and national ID smartcards to cast votes online.

TLS client authentication [327]) and make legally binding signatures on documents [96]. The cards are popularly used for online banking and accessing e-government services [103]. In the I-voting system, voters use their ID cards to authenticate to the server and to sign their ballots.

Each card contains two RSA key pairs, one for authentication and one for making digital signatures. Certificates binding the public keys to the cardholder's identity are stored on the card and in a public LDAP database [98]. The card does not allow exporting private keys, so all cryptographic operations are performed internally. As an added safeguard, each key is associated with a PIN code, which must be provided to authorize every operation.

Estonians can also use mobile phones with special SIM cards for authentication and signing, through a system called Mobile-ID [97]. In the 2013 election, 9% of online votes were cast using this method [101]. We exclude Mobile-ID from our analysis because we did not have access to the external infrastructure that would be needed to test it.

Figure 3.2: **Verification app** — A smartphone app allows voters to confirm that their votes were correctly recorded. We present two strategies an attacker can use to bypass it.

3.1.2 I-Voting Server Infrastructure

The majority of the I-voting server source code is published to a GitHub repository 2–3 weeks prior to the election [112]. The server infrastructure is configured in a public ceremony one week before the election and consists of four machines:

Vote forwarding server (VFS/HES) The VFS (or HES in Estonian) is the only publicly accessible server. It accepts HTTPS connections from the client software, verifies voter eligibility, and acts as an intermediary to the backend vote storage server, which is not accessible from the Internet.

Vote storage server (VSS/HTS) The VSS is a backend server that stores signed, encrypted votes during the online voting period. Upon receiving a vote from the VFS, it confirms that the vote is formatted correctly and verifies the voter's digital signature using an external OCSP server.

Log server This server is an internal logging and monitoring platform that collects events and statistics from the VFS and VSS. The source code and design have not been published. While this server is not publicly accessible, it can be accessed remotely by election staff.

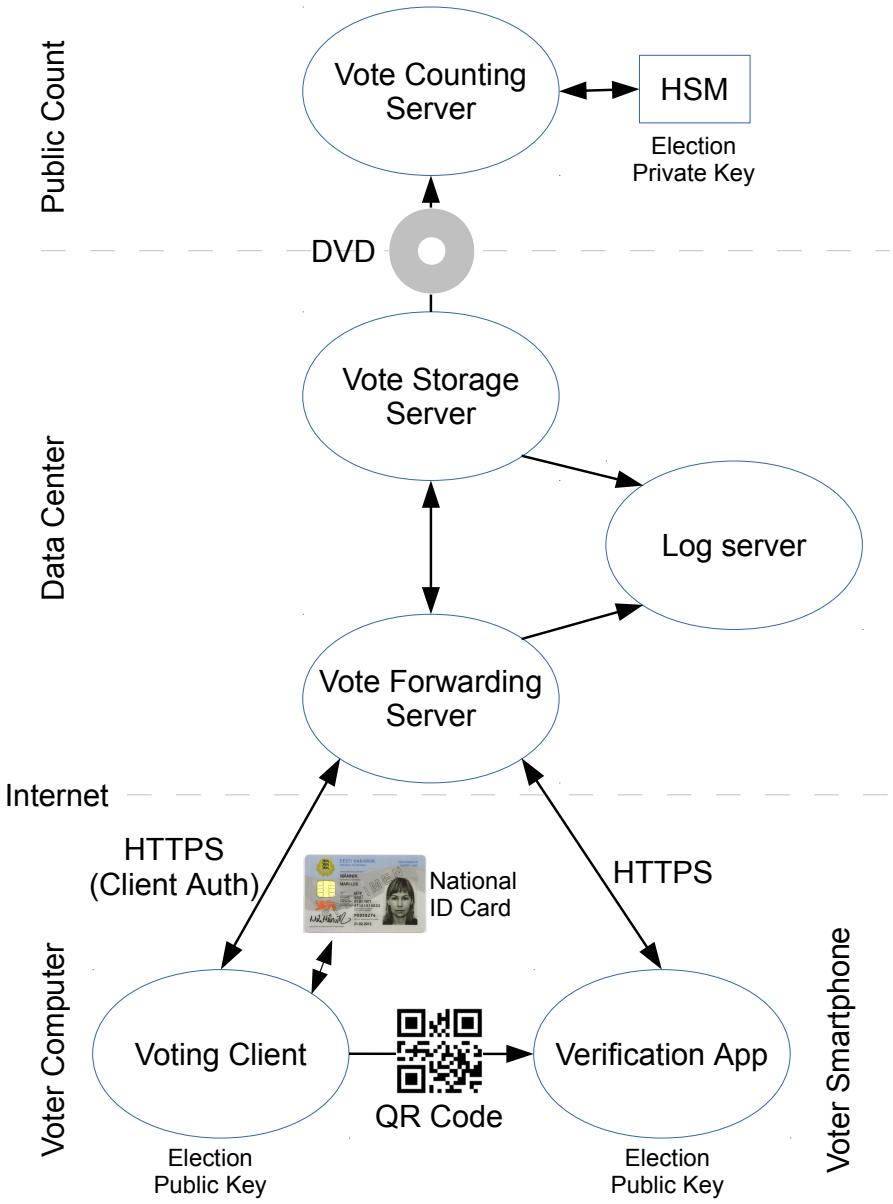

Figure 3.3: **I-voting system overview** — Major components of the system, and how information flows among them.

Vote counting server (VCS/HLR) The VCS is never connected to a network and is only used during the final stage of the election. Officials use a DVD to copy encrypted votes (with their signatures removed) from the VSS. The VCS is attached to a hardware security module (HSM) that contains the election private key. It uses the HSM to decrypt the votes, counts them, and outputs the official results.

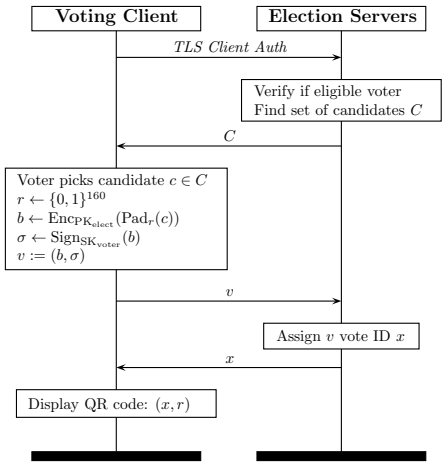

Figure 3.4: **Vote casting process**

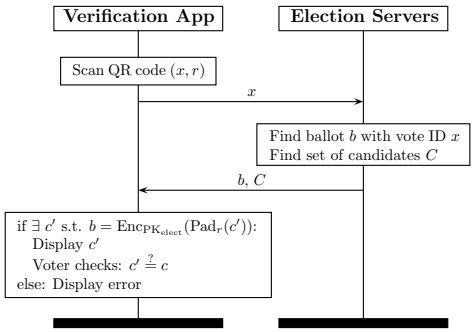

Figure 3.5: **Vote verification process**

3.1.3 Voting Processes

The I-voting system uses public key cryptography to provide a digital analog of the "double envelope" ballots often used for absentee voting [106]. Conceptually, an outer envelope (a digital signature) establishes the voter's identity, while an inner envelope (public key encryption) protects the secrecy of the ballot. Once each voter's eligibility has been established, the signature is stripped off, leaving a set of anonymous encrypted ballots. These are moved to a physically separate machine, which decrypts and counts them.

Casting At the start of each election, the election authority publishes a set of voting client applications for Windows, Linux, and Mac OS, which can be downloaded from https://valimised.ee. The client is customized for each election and includes an election-specific public key for encrypting the voted ballot and a TLS certificate for the server.

Figure 3.4 shows the protocol for casting a vote. The voter begins by launching the client application and inserting her ID card. She enters the PIN associated with her authentication key, which is used to establish a client-authenticated TLS connection to the VFS. The client verifies the server's identity using a hard-coded certificate. The server confirms the voter's eligibility based on her public key and returns the list of candidates for her district [72].

The voter selects her choice c and enters her signing key PIN. The client pads c using RSA-OAEP and randomness r, encrypts it with using the 2048-bit election public key, and signs the encrypted vote with the voter's private key. The signed and encrypted vote is sent to the server, which associates it with an unguessable unique token x and returns x to the client. The client displays a QR code containing r and x.

As a defense against coercion, voters are allowed to vote multiple times during the online

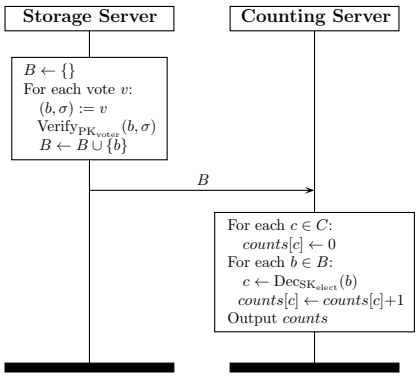

Figure 3.6: **Vote tabulation process**

election period, with only the last vote counted. All earlier votes are revoked but retained on the storage server for logging purposes. While the client indicates whether the user has previously voted, it does not display the number of times. The voter can also override her electronic vote by voting in person on election day.

Verification The voter can confirm that her vote was correctly recorded using a smartphone app provided by the election authority [107, 109, 110], as seen in Figure 3.2. This protocol is shown in Figure 3.5. The app scans the QR code displayed by the voting client to obtain r and x. It sends x to the election server, which returns the encrypted vote b (but not the signature) as well as a list of possible candidates. The app uses r to encrypt a simulated vote for each possible candidate and compares the result to the encrypted vote received from the server. If there is a match, the app displays the corresponding candidate, which the voter can check against her intended choice. The server allows verification to be performed up to three times per vote and up to 30 minutes after casting.

Tabulation Figure 3.6 shows the sequence that occurs at the conclusion of an election. After online voting has ended, the storage server processes the encrypted votes to reverify the signatures and remove any revoked or invalid votes. During a public counting session, officials export the set of valid votes after stripping off the signatures, leaving only anonymous encrypted votes. These are burned to a DVD to transfer them to the counting server.

The counting server is attached to an HSM that contains the election private key. The server uses the HSM to decrypt each vote and tallies the votes for each candidate. Officials export the totals by burning them to a DVD. These results are combined with the totals from in-person polling stations and published as the overall results of the election.

3.2 Observations

The first part of our analysis follows an observational methodology. Four of the authors (Halderman, Hursti, Kitcat, and MacAlpine) visited Estonia as officially accredited election observers during the October 2013 municipal elections and witnessed the operation of the I-voting servers. During that time, they also met with election officials and the I-voting software developers in Tallinn and Tartu. Later, we closely reviewed published artifacts from the election: the server source code [112], written procedures [99], and nearly 20 hours of official videos that recorded the I-voting configuration, administration, and counting processes [100]. Ultimately, we identified a range of problems related to poor procedural controls, lapses in operational security, and insufficient transparency measures.

3.2.1 Inadequate Procedural Controls

While the Internet Voting Committee (the administrative body that runs the system) has published extensive written procedures covering many steps in the election process [99], we observed that some procedures were not consistently followed and others were dangerously incomplete.

Procedures for handling anomalous conditions that could imply an attack appear to be inadequately specified or do not exist. For example, tamper-evident seals are used on the server racks in the data center.[2] When asked what would happen if the seals were found to be compromised, election staff responded that they were unsure.

Anomalous situations that occurred during the 2013 election were handled in an ad hoc manner, sometimes at the discretion of a single individual. On multiple occasions, we observed as data center staff restarted server processes to resolve technical glitches, and repeated failed commands rather than troubleshooting the root cause. Similar issues were observed during tabulation when the election officials attempted to boot the vote storage server to export the encrypted votes. The machine reported errors stating that the drive configuration had changed—a possible indication of tampering. Instead of investigating the cause of the alert, staff bypassed the message.

Some procedures appeared to change several times over the observation period. For example, observers were initially allowed to film and photograph inside the server room, but were prohibited the next day because of the unsubstantiated claim of "possible electronic interference." In a similarly abrupt change in procedure, observers were required to leave

[2] We note that tamper-evident seals like those used in Estonia are known to be easy to defeat using widely available tools [185–188]. Their usefulness for election security has been questioned in other countries [16, 422].

their mobile phones outside the data center after multiple days where this was not the policy. Rewriting the rules on the fly suggests that the procedures had not been adequately thought out or were insufficiently defined for staff to implement them consistently.

Even when procedural safeguards were clear, they were not always followed. For example, procedure dictates that two operators should be present when performing updates and backups [72]. Yet, on October 14, we observed that a lone staff member performed these tasks. Without a second operator present, the security of the system relies on the integrity of a single staff member.

3.2.2 Lax Operational Security

Since the I-voting system treats parts of the server infrastructure as implicitly trusted, the processes used to install and configure those servers are crucial for the security of the election. We witnessed numerous serious lapses in operational security both during our on-site observations and in the official videos released by the Internet Voting Committee.

Pre-election setup Several problems can be seen in the official videos of the pre-election setup process, which takes place in the National Electoral Committee's offices in Tallinn.

The videos show election workers downloading software for use in the setup process from a public website over an unsecured HTTP connection (Figure 3.8). A network-based man-in-the-middle attacker could compromise these applications and introduce malware into the configuration process.

In other instances. workers unintentionally typed passwords and national ID card PINs in view of the camera (Figures 3.9 and 3.10). These included the root passwords for the election servers. Similar problems occurred during daily maintenance operations in the data center. Physical keys to the server room and rack were revealed to observers; these keys could potentially be duplicated using known techniques [211].

The most alarming operational security weakness during pre-election setup was workers using an "unclean" personal computer to prepare election client software for distribution to the public. As seen in Figure 3.7, the desktop has shortcuts for an online gambling site and a BitTorrent client, suggesting that this was not a specially secured official machine. If the computer used to prepare the client was infected with malware, malicious code could have spread to voters' PCs.

Daily maintenance We observed further operational security weaknesses during daily maintenance procedures that took place during the voting period. The I-voting servers are

hosted at a government data center in Tallinn, and workers go there to perform operations at the server consoles. While there were security video cameras at the data center, there appeared to be no 24/7 security personnel nor any definitive information on who monitored the cameras.

Standard practice during daily maintenance is for workers to log in to the election servers under the `root` account and perform operations at the shell. Logging in as `root` is contrary to security best practice, as it simplifies many attacks, disables user-based privilege separation for operator functions, and increases the risk of human error.

Unencrypted daily backups were casually transported in workers' personal backpacks. DVDs holding updated voter lists from the population register were handled in a similarly casual way after having been created, we were told, by a member of staff at their own computer. We did not observe any audit trail or checks on the provenance of these DVDs, which were used daily at the heart of the I-voting system.

Tabulation The tabulation process at the end of the election was also concerning. After the votes were decrypted on the counting server, an unknown technical glitch prevented workers from writing the official counts and log files to DVD. Instead, officials decided to use a worker's personal USB stick to transfer the files to an Internet-connected Windows laptop, where the results were officially signed. This USB stick had been previously used and contained other files, as shown in Figure 3.12. This occurred despite protest from an audience member and deviated significantly from the written procedure, adding multiple potential attack vectors. Malware present on the laptop or USB stick could have altered the unsigned results, or malware on the USB stick could have been transferred to the trusted counting server.

These instances illustrate a pattern of operational security lapses on the part of the workers who operated the I-voting system. This is particularly alarming given the high degree of trust the I-voting system design requires of the election servers, client software, and the election workers themselves.

3.2.3 Insufficient Transparency

The election officials have implemented a number of transparency measures, including allowing in-person public observation, publishing videos of operator tasks, and releasing large parts of the server source code. While these measures appear to be well intended, they are incomplete and insufficient to fully establish the integrity of election results.

One limitation is that these measures cannot show whether malicious actions were

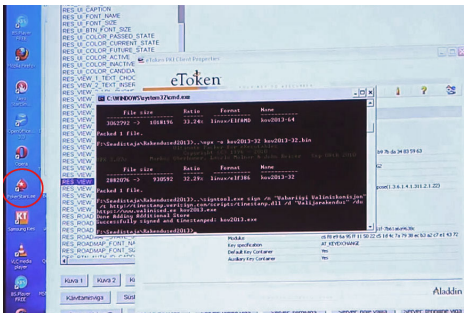

Figure 3.7: **Official build system is multi-use** — Operators used a PC containing other software, including PokerStars.ee, to sign the official voting client for public distribution. This risks infecting the client with malware spread from the PC.

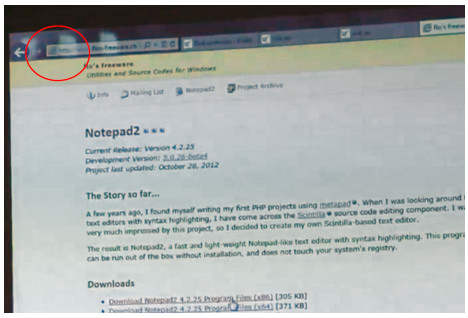

Figure 3.8: **Insecure software downloads** — Operators downloaded software over insecure connections for use in pre-election setup. An attacker who injected malware into these downloads might be able to compromise the process.

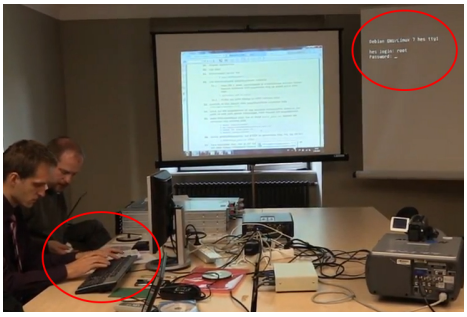

Figure 3.9: **Keystrokes reveal root passwords** — Videos posted by officials during the election show operators typing, inadvertently revealing root passwords for election servers.

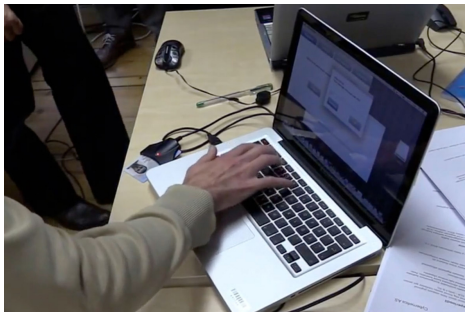

Figure 3.10: **Video shows national ID PINs** — During pre-election setup, someone types the secret PINs for their national ID card in full view of the official video camera.

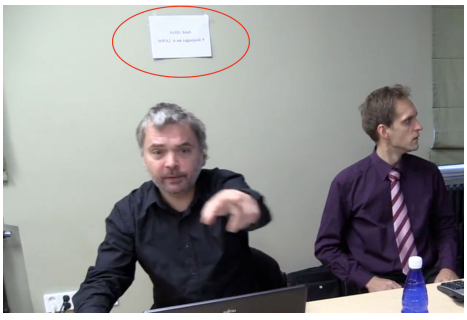

Figure 3.11: **Wi-Fi credentials posted** — The official video of the pre-election process reveals credentials for the election officials' Wi-Fi network, which are posted on the wall.

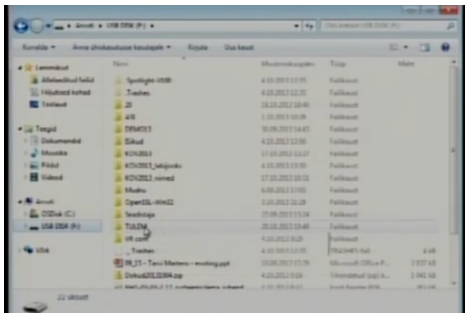

Figure 3.12: **Personal USB stick** — Against procedures, an official used a USB stick, containing personal files, when moving the official election results off of the counting server.

performed on the servers or hard disks before recording and observation commenced. In practice, they also do not capture everything going on in the facilities. On many occasions, there were multiple machines or screens in use simultaneously but only a single camera.

Although we attempted to follow these simultaneous operations during our in-person observations, the operators occasionally appeared to be deliberately evading us. On Monday, October 14, we were physically present in the server room when one of the servers produced an abnormal output which appeared to be a failure of the update operation. The official video recording was following the other display, and the operator, upon seeing the error message, quickly flushed it from the screen. In another instance when an error appeared on the server console, an election worker quickly cleared the display and then asked us to rotate out of the room and let other observers in, allowing him a block of observation-free time.

An auditor from a major international consulting firm had been hired by the Internet Voting Committee, and his report also documents procedural and operational shortcomings [340]. However, the auditor's role was chiefly to observe, and he was not provided with the access needed to confirm secure operation of the system.

Election officials have made large parts of the server software open source [112]. This is a positive measure as it allows independent review and assessment — indeed, our study made extensive use of the code. However, security-critical pieces of code are missing from the published sources, including the entire client application and code that is executed on every server machine (see Section 3.3.1).

Officials told us that the client source is not released (and furthermore, the client binary is obfuscated) because they are concerned that attackers might modify it and distribute a trojan lookalike. Creating client-side malware is feasible without the source, as we show in Section 3.4.1. At the same time, keeping source code secret prevents the public from understanding what they are being asked to run on their computers, and it increases the risk that any centrally introduced malicious changes to the client will go undetected.

As these transparency measures are practiced today, the videos, public observation, and open-source components risk providing a false sense of security. It is possible for the systems to be corrupted prior to videotaping, for media used to update servers to be maliciously modified, or for unpublished pieces of software to contain errors or malicious code — all invisible to the public under current transparency measures.

To illustrate these limitations, we conducted an experimental server-side attack on a reproduction of the system, as detailed in Section 3.4.2. We have published a series of videos (see https://estoniaevoting.org/videos/) matching the procedures in the official videos step-by-step, but where the result of the election is dishonest because of malware surreptitiously introduced before the start of pre-election setup.

3.2.4 Vulnerabilities in Published Code

The published portions of the I-voting server software [112] contain 17,000 lines of code, with 61% in Python, 37% in C++, and the remainder in shell scripts. The codebase is quite complex, with a large number of external dependencies, and exhaustively searching it for vulnerabilities would be a challenging task well beyond the scope of this project. We understand that volunteers from the Estonian security community have already audited it—a testament to the virtues of publishing code. Nonetheless, we discovered some minor bugs and vulnerabilities while examining the code in order to conduct our other experiments. We disclosed these issues to the Internet Voting Committee in May 2014.

One of the problems we found allows a denial-of-service attack against the voting process. If a client sends an HTTP request containing unexpected header fields, the server logs the field names to disk. By sending many specially crafted requests containing fields with very long names, an attacker can exhaust the server's log storage, after which it will fail to accept any new votes. In the 2013 election, the size of the log partition was 20 GB. We estimate that an attacker could fill it and disable further voting in about 75 minutes. Curiously, the vulnerable code is only a few lines from the comment, "Don't write to disk; we don't know how large the value is." This indicates that the developers were aware of similar attacks but failed to account for all variants.

A second problem we discovered is a shell-injection vulnerability in a server-side user interface that is intended to allow operators to perform pre-determined administrative tasks. The vulnerability would allow such an operator to execute arbitrary shell commands on the election servers with root privileges. Under current procedures, this is moot, since the same workers perform other administrative tasks at the command line as root. However, shell injection vulnerabilities can be exceedingly dangerous [423], and the fact that the issue was not detected in advance of the election is a reminder that open source cannot guarantee the absence of vulnerabilities [201].

3.3 Experimental Methodology

In order to further investigate the security of the I-voting system, we set up a copy of the system in our lab, reproducing the software and configuration used for the 2013 election. While pen testing during a real election would have involved numerous legal and ethical problems [346], our laboratory setup allowed us to play the role of attacker in our own mock election without any risk of interfering with real votes.

3.3.1 Mock Election Setup

To reproduce the I-voting servers, we used the source code published on GitHub by the election authority [112]. We set up the servers by following published configuration documents [99] and matching step-for-step the actions performed by election workers in the official videos [100]. As part of this process, we generated our own key pairs for the web server TLS certificate and for the election key.

Some components were missing from the published server code, but we attempted to recreate them as faithfully as possible. First, the software for the log server, the `ivote-monitor` package, was not made available; we operated a standard `rsyslog` [348] server instead. Additionally, there was no source provided for the `evote_post.sh` script, which runs on every server during installation of the packages. We attempted to reproduce its functionality based on output shown during server configuration in the official videos.

In real elections, Estonia uses a hardware security module (HSM) in order to handle the election private key and decrypt votes. Since we did not have compatible hardware available, we emulated the HSM in software using OpenSSL and Python. Since this deviates from the fielded setup, we ensured that none of our attacks depend on vulnerabilities in the HSM.

We set up our own certificate authority and OCSP responder as stand-ins for the national ID card PKI. This allowed us to generate identities for mock voters. Since we did not have access to actual Estonian ID cards, we had to emulate them. We replaced the ID card on the client with a software-based emulator that speaks the protocol expected by the voting client application. Once again, we ensured that the success of our attacks does not rely on the changes we made. We assume for purposes of this study that the ID cards and associated infrastructure are secure.

For the client software, we started with the official voting client from the 2013 election, which we downloaded from the election website [105] in October 2013. For convenience, we focused on the Linux version of the client. Since the election public key and server certificate are hard-coded into the client, we needed to patch it in order to replace these with the keys of our mock election and server. Similarly, we used the official source code for the Android-based verification app [111] and modified it to communicate with our server.

Virtual machines we used to reproduce the election, together with source code for our demonstration attacks, are available online at https://www.estoniaevoting.org.

3.3.2 Threat Model

After setting up the mock election, we attempted to compromise it, allowing ourselves the resources and capabilities of a sophisticated but realistic attacker. This attacker could be a

foreign state, a well-funded criminal organization, or a dishonest election insider. These kinds of attackers are difficult to defend against, but they represent a serious and realistic threat to modern elections given the enormous political and financial consequences at stake.

Since the time the Estonian system was introduced, cyberwarfare has become a well documented reality. Chinese espionage against U.S. companies [225], U.S. sabotage of Iran's nuclear enrichment program [354], and attacks by the U.K. against European telecommunications firms [406] are just a few examples. An increasing number of nations possess offensive computer security capabilities [324], and investment in these capabilities is reported to be growing at a significant rate [121]. Estonia itself suffered widespread denial-of-service attacks in 2007 that have been linked to Russia [405]. More recently, in May 2014, attackers linked to Russia targeted election infrastructure in Ukraine, which uses a computerized system to aggregate results from around the country. The attackers reportedly attempted to discredit the election process by disrupting tallying and causing the system to report incorrect results [58].

A state-sponsored attacker would have powerful capabilities. We assume that they could obtain a detailed knowledge of the I-voting system's operation, which can be gleaned from published sources and reverse engineering (as we did), from insider knowledge, or by compromising systems used by the software developers and election officials. We also assume that if reverse engineering is required, the attacker would have sufficient human and technical resources to accomplish this on a short timescale. For client-side attacks, we assume that the attacker has the ability to deliver malware to voters' home computers. This could be done externally to the voting system, either by purchasing pre-existing criminal resources, such as a botnet, or by buying or discovering zero-day vulnerabilities in popular software. Another route would be to compromise the voting client before its delivery to voters, either by a dishonest insider who can alter the software or by other attackers who can compromise the computers used to build or distribute it.

3.4 Attacks

We used our reproduction of the I-voting system to experiment with a range of attacks. The I-voting system places significant trust in client and server components, making these highly attractive targets for an attacker.

While certain server operations are protected by cryptography (e.g., cast votes cannot be decrypted on the front-end web server, since it lacks the requisite private key), in other instances the servers are completely trusted to perform honestly and correctly when handling votes. Similarly, while the smartphone verification app gives voters some ability to check

that the client software is behaving honestly, there are major limitations to this safeguard that can be exploited to hide malicious client behavior.

We experimentally verified that these trusted components are vulnerable by conducting two sets of demonstration attacks against them in our mock election setting. The first type are attacks on the client that are within reach of a financially capable attacker, in which an attacker can change votes in a retail manner for large numbers of individual voters. The second kind are server-side attacks within the reach of a well-resourced state-level attacker or dishonest insider, in which an attacker could change the wholesale results of the entire election by compromising the vote counting server.

3.4.1 Client-side Attacks

The voter's client machine is a trusted component of the I-voting system, and there are several ways that an attacker might try to infect a sufficient number of Estonian clients to alter election results in a close race. One is to rent bots from pre-existing botnets. Botnet operators frequently offer them for rent on the black market, and these can be targeted to a specific country or region [73]. A second way would be to discover or purchase a zero-day exploit against popular software used in Estonia. While this would be expensive, it would not be out of reach for a state-level attacker—several companies specialize in selling zero-day exploits to governments [164]. A third strategy would be to infect the official I-voting client before it is delivered to voters. The operational security problems documented in Section 3.2.2 suggest that this is a practical mode of attack.

If the attacker's goal is merely to disrupt the election, far fewer infected system would be required. Estonian law allows election officials to cancel online balloting if a problem is detected [169]. In that case, Internet voters would have their electronic votes canceled and be required to go to the polls on election day. This is a useful emergency measure, but it requires election officials to both detect the attack and make a snap decision about its severity. Suppose an attacker infected a small number of clients with vote-stealing malware, allowed some of them to be detected, and designed the attack such that it was difficult to quickly determine the size of the infected population. Under these circumstances, officials might be compelled to cancel the online portion of the election, yet the attacker would need relatively few resources.

In order to investigate how an attacker could modify votes through the client application, we implemented two experimental client-side attacks. Both assume that the attacker has used one of the techniques noted above to initially infect the client machine. Each attack uses a different mechanism to defeat the smartphone verification app (see Section 3.1.3), the

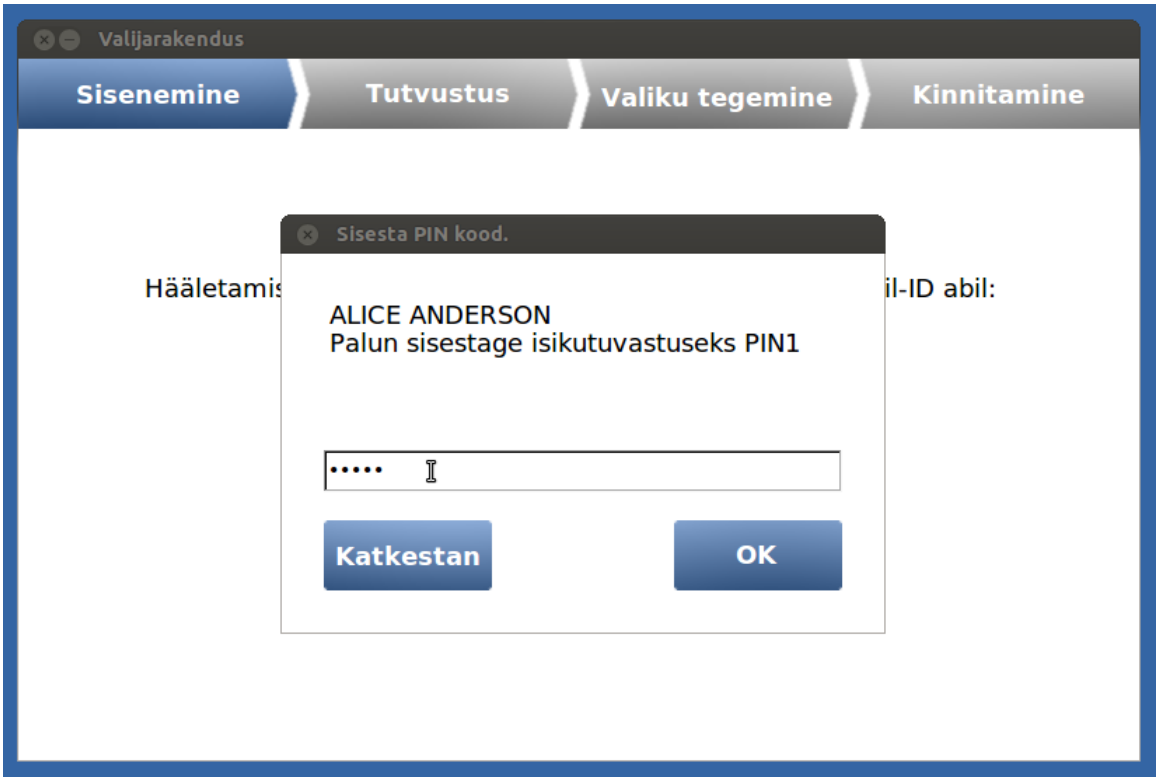

Figure 3.13: **Malware records secret PINs**—Estonians use their national ID smartcards to sign and cast ballots. We developed demonstration client-side malware that captures the smartcard PINs and silently replaces the user's vote.

only tool available to voters to detect whether their ballot choices have been manipulated by client-side malware.

Both attacks involve hidden malicious processes that run alongside the I-voting client application and tamper with its execution. In order to develop them, we first needed to reverse engineer parts of the client software used in 2013. The client is closed source, and the developers took measures to complicate reverse engineering. The executable is obfuscated using the UPX packing tool. Strings, public keys, and other resources are hidden by XORing them with the output of a linear congruential generator. These measures did not significantly complicate the construction of our attacks. We used the UPX application with the `-d` switch to unpack the binary and used the IDA Pro disassembler and Hex-Rays decompiler to reverse the portions necessary for our attacks.

Ghost Click Attack In the first attack, we use malware on the client machine to silently replace the user's vote with a vote in favor of an attacker-selected candidate. At a high level, the malware silently sniffs the victim's PIN during the original voting session. The real vote is cast, and everything appears normal, including the verification smartphone app if the

voter uses it. Then, the malware waits until it is too late to verify again—either until the 30 minute time limit has passed or until after the user closes the client software and the QR code can no longer be scanned.

At that point, the malware checks whether the voter's ID card is still present in the computer. If so, it opens a copy of the I-voting client in a hidden session and, through keystroke simulation, submits a replacement vote. If the ID card has already been removed, the malware remains dormant until the card is inserted again. Since Estonian ID cards are used for a variety of applications, many voters are likely to use their cards again within the week-long online voting period.

In our implementation, the malware attaches to the voting client process and captures PINs by setting breakpoints using `ptrace` [338]. Upon reaching a breakpoint, it reads the PIN from the client's memory and stores it for future use. Although our implementation runs in userspace, a kernel-level rootkit could be used to make the attack even more difficult to detect [174]. The malware could be extended to sniff PINs opportunistically any time voters use their ID cards, such as when logging into a bank.

Bad Verify Attack The Ghost Click attack defeats the verification app, but applying it on a large scale would lead to a suspiciously high number of replacement votes. We also experimented with a stealthier but more complicated style of attack that targets the verification app directly.

The verification app is premised on the notion that the smartphone is an independent device that is unlikely to be compromised at the same time as the client PC. However, modern smartphones are not well isolated from users' PCs, as there is typically regular communication between the two devices. Users frequently plug their phones into their PCs to charge them or to transfer files. User content is regularly synchronized between devices through Google Drive, Dropbox, and other cloud services. Android even allows users to remotely install applications on their phones from their PCs through the Google Play Store web interface, and other platforms have similar mechanisms [233]. As a result of this convergence, there are abundant means by which PC malware can attempt to infect the user's phone. This would allow the attacker to deploy a dishonest verification app that colludes with malware on the PC to fool the voter.

To experiment with such an attack, we implemented tandem PC and smartphone malware. Malware on the PC detects which candidate the voter selects and modifies the QR code shown by the I-voting client so that it encodes the voter's chosen candidate. A malicious verification app on the voter's phone behaves just like the real verification app, except that it displays whatever candidate is embedded in the QR code, rather than the candidate for

whom the vote was actually cast. This allows the PC malware to arbitrarily change the submitted vote without being detected by verification or causing a suspicious number of replacement votes.

This form of attack adds complexity, due to the need to compromise both devices simultaneously. It also carries an elevated possibility of detection if used on a large scale, since some voters may attempt verification with devices owned by others. However, it illustrates that as the PC and smartphone platforms continue to converge, it will become increasingly unsafe to treat them as independent devices.

3.4.2 Server-side Attacks

The integrity of the count depends on the correct operation of the counting server and its HSM, which are the only components with the ability to decrypt votes. Similarly, ballot secrecy depends on the counting server to not leak information about the correspondence between encrypted and decrypted votes. An attacker who infects the counting server with malicious software can violate these critical security properties.

Although the counting server is not connected to a network, there are a number of other means by which it might be attacked. State-level attackers are known to employ firmware-based malware [17], which could be used to infect the BIOS or hard disk before delivery to election officials. Sophisticated attackers can also target component supply chains and distribution infrastructure [193]. We were informed during our observation mission that the Internet Voting Committee has a bid process prior to every election to rent the servers that they use, and attackers could try to introduce subverted hardware through this process.

Another infection strategy would be to compromise the server software before it is installed at the beginning of the election. We pursued this route in our experiments.

Injecting malware Despite procedural safeguards [99], an attacker who strikes early enough can introduce malicious code into the counting server by using a chain of infections that parallels the configuration process. During pre-election setup, workers use a development machine, which is configured before setup begins, to burn Debian Linux installation ISOs to DVDs. These DVDs are later used to configure all election servers. If the machine used to burn them is compromised — say, by a dishonest insider, an APT-style attack on the development facility, or a supply-chain attack — the attacker can leverage this access to compromise election results.

We experimented with a form of this attack to successfully change results in our mock election setup. We first created a modified Debian ISO containing vote-stealing malware

intended to execute on the counting server. The tainted ISO is repackaged with padding to ensure that it is identical in size to the original. In a real attack, this malicious ISO could be delivered by malware running on the DVD burning computer, by poisoning the mirror it is retrieved from, or by a network-based man-in-the-middle.

Defeating integrity checking During the setup process, election workers check the SHA-256 hash of the ISO file against the `SHA256SUMS` file downloaded via anonymous FTP from debian.org. Since regular FTP does not provide cryptographic integrity checking, a network-based man-in-the-middle could substitute a hash that matched the malicious ISO. However, this hash would be publicly visible in videos of the setup process and might later arouse suspicion.

An attacker who had compromised the DVD burner computer could achieve greater stealth. To demonstrate this, we implemented a custom rootkit that defeats the hash verification. Our rootkit is a kernel module that hooks system calls in order to cause the hash verification to succeed and the original ISO's hash to be printed. Hash checks applied in this way are only a minor speedbump under our threat model.

Vote-stealing payload After passing the hash check, the tainted ISOs are used to install the OS on all election servers, spreading the infection. When the new OS boots, the malware checks whether the machine is configured as the counting server, in which case it launches a vote-stealing payload.

During the counting process, this payload acts as wrapper around the process responsible for using the HSM to decrypt votes. This allows the malware to alter the decrypted votes prior to returning them to the counting application. (In our demonstration, we change 100% of the votes, but it would be straightforward to implement a more subtle algorithm that manipulates an arbitrary fraction.) The altered votes are then counted and released as the official results. Such an attack would be unlikely to be detected, as there is no audit mechanism to check the accuracy of the decryption.

Other avenues for infection What we have described is far from the only means of injecting a malicious payload into the servers. Several other pieces of closed-source software of unknown or untrusted provenance could be vehicles for attacks. These include the `evote_post.sh` script, missing from the server source code repository, which runs on all servers, as well as the driver software for the HSM, which is a closed-source application manually installed to the counting server from a DVD. These programs all touch critical, trusted portions of the I-voting system, yet they are not reviewable by the public and not

integrity checked through any visible procedures.

In fact, during the pre-election server setup process in 2013, workers used an incorrect version of the `evote_post.sh` script that failed to install the `evote_analyzer` package on the VFS. Administrators later had to manually install this package during the voting period, after they realized that the server was not reporting all expected log data [340]. This provides a case-in-point example of a failure of the procedural protections to ensure that only the correct software gets installed on the server machines.

Zero-day exploits are yet another potential attack vector, and a source of many "known unknowns." One illustration of this is the OpenSSL Heartbleed bug [61], which was not disclosed until April 2014. The front-end server used during the 2013 election was vulnerable to Heartbleed, and an attacker who knew about the bug likely could have exploited it to extract the server's TLS private key. Then, using a man-in-the-middle attack on connections from voters, they could have selectively prevented certain voters' ballots from being received by the real server.

The key lesson from our server-side attack is that it is extremely difficult to ensure the integrity of code running on a critical system, particularly when faced with the possibility of sophisticated attacks or dishonest insiders. If any element in the lineage of devices that handle the software installed on the counting server is compromised, this could jeopardize the integrity of election results [402].

3.4.3 Attacking Ballot Secrecy

While our experiments focused on attacks against the integrity of election results, we also considered ballot secrecy issues, since the secrecy of the voter's ballot is a critical defense against voter coercion and vote buying. The I-voting system implements a relatively strong protection against in-person, individual coercion by allowing voters to cast replacement votes online or to cancel their electronic ballots entirely and vote in person on election day. More sophisticated attacks remain possible, however, including spyware on the voter's PC or smartphone, as well as server-side attacks.

Server-side attacks on ballot secrecy are particularly troubling, since preserving ballot secrecy is a main goal of the system's cryptographic double-envelope architecture. The I-voting design attempts to ensure that votes remain private by breaking the association between voters' digital signatures from their plaintext votes. The encrypted ballots are separated from the signatures and copied to an isolated machine before being decrypted and counted. Note that this machine, the counting server, has access to the complete association between the encrypted ballots and the plaintext votes. An attacker who can

smuggle this information out through a covert channel can compromise every voter's secret ballot.

Unfortunately, the tabulation procedures offer multiple possibilities for exfiltrating this information. When tabulation is complete, officials use the counting server to burn a DVD containing both vote totals and log files. Suppose for simplicity that the attacker is a dishonest insider with access to this DVD and to the complete set of signed, encrypted ballots (e.g. from a backup disk) and some mechanism for infecting the counting server with malicious code, such as the routes discussed above. The counting server malware can sort the encrypted ballots and leak the voter choices corresponding to each as a sequence of integers in the same order. Since there is typically only one race, only a few bits per ballot are needed to determine the choices of all voters. The malware could steganographically encode this data into the log files through the order of entries, or it could simply write this information to unallocated sectors of the disc. The attacker can then decode this information and use it to associate every voter's digital signature (and hence, their identity) with their vote.

3.5 Discussion

Though we have spent the majority of this report discussing weaknesses and risks, we would be remiss if we failed to acknowledge the great lengths that the I-voting system developers, security staff, and officials go to in their efforts to protect the election system.

One core strength of the I-voting system is Estonia's national ID card infrastructure and the cryptographic facilities it provides. While the ID cards cannot prevent every important attack, they do make some kinds of attacks significantly harder. The cards also provide an elegant solution for remote voter authentication, something few countries do well.

The Internet Voting Committee's willingness to release source code is a very positive step for transparency. This shows confidence in the software's developers and demonstrates officials' desire to work with the security community at large. Providing access to the source allows many parties to analyze it—not only international researchers like us but also the domestic security community, who have an even greater interest in the system's secure operation. For these reasons, we urge the committee to go further and release the source code to the I-voting client and the missing portions of the server code discussed in Section 3.3.1.

Finally, we commend the Internet Voting Committee for their dedication to improving the election system. Since its inception in 2005, the system has undergone significant changes. From the switch to a standalone client, to the deployment of the log server that

enhances forensic and monitoring capabilities, to the addition of the verification app, the I-voting system has not stood still. Yet as we have argued, even these and an array of other useful safeguards are not enough to secure Estonia's online elections in the face of a determined and well-resourced modern attacker.

Opportunities for innovation While the risks of Internet voting are clear, the benefits are uncertain. Many Estonians support I-voting because they believe there is widespread fraud in the country's paper-based system. Whether or not these concerns are founded, the I-voting system can do little to help, since nearly 80% of votes are still cast on paper. Fortunately, there are safe and effective ways to apply new technology to secure paper-based voting.

In recent years, researchers have developed methods that can dramatically increase the security of paper ballots. Statistical risk-limiting audits [41, 44, 217, 391] can minimize the risk of error or fraud during tabulation. Cryptographic techniques that achieve end-to-end verifiability [28, 52, 351] enable individual voters to verify that every vote has been counted accurately. Estonia has an opportunity to be the first country in the world to adopt these technologies on a national scale.

3.6 Conclusions

Compared to other online services like banking and e-commerce, voting is an exceedingly difficult problem, due to the need to ensure accurate outcomes while simultaneously providing a strongly secret ballot. When Estonia's I-voting system was conceived in the early 2000s, it was an innovative approach to this challenge. However, the designers accepted certain tradeoffs, including the need to trust the central servers, concluding that although they could take steps to reduce these risks through procedural controls, "the fundamental problem remains to be solved" [14]. More than a decade later, the problem remains unsolved, and those risks are greatly magnified due to the rapid proliferation of state-sponsored attacks.

As we have observed, the procedures Estonia has in place to guard against attack and ensure transparency offer insufficient protection. Based on our tests, we conclude that a state-level attacker, sophisticated criminal, or dishonest insider could defeat both the technological and procedural controls in order to manipulate election outcomes. Short of this, there are abundant ways that such an attacker could disrupt the voting process or cast doubt on the legitimacy of results. Given the current geopolitical situation, we cannot discount state-level attacks targeting the system in future elections.

Due to these risks, we recommend that Estonia discontinue use of the I-voting system. Certainly, additional protections could be added in order to mitigate specific attacks (e.g. [220]), but attempting to stop every credible mode of attack would add an unmanageable degree of complexity. Someday, if there are fundamental advances in computer security, the risk profile may be more favorable for Internet voting, but we do not believe that the I-voting system can be made safe today.

2017 Update In 2017, researchers from Masaryk University and Enigma Bridge released the existence of a vulnerability in the RSA prime generation of Infineon's cryptographic hardware. The researchers found that more that half of a 10% sample of Estonian national ID cards were vulnerable to this attack with an expected cost of 50 CPU-years and worst case of 100 CPU-years. While a non-trivial cost, the attack can be trivally parallelized such that the wall-clock time can be trivial [248]. Estonian officials have acknowledged that ID cards issued after October 2014 are vulnerable [93] but proceeded with the October 2017 municipal elections as normal [94]. While the parallel nature of the attack makes the attack feasible to even moderately resourced attackers, it is a symmetric attack and is not easily scalable to cast large numbers of fraudulent votes. However it would be feasible to attack the keys used by election officials to sign the voter roll updates, client binary, or official election results (assuming that the election officials' keys are vulnerable).

Nation-State Attacker Model When we look at this research in the context of our NS-Attacker model, we see many fitting facets. The first and most obvious is the use of the NS-Attacker's Near Superset Attacker advantage. In the context of the in-person observations, published videos, and publicly available documentation, we were able to find and identify many of the OPSEC and procedural failures and it is highly likely that an NS-Attacker would too. Additionally, our efforts to reverse engineer and deeply understand the protocol and infrastructure were successful implying that an NS-Attacker's would be as well.

Another NS-Attacker advantage is Specialization as it applies to two portions of the Estonian Internet Voting system. The first is combining skilled linguists to our OPSEC and procedural analysis. While we gleaned information mainly from recorded computer screens and Google Translate interpretations of the documents, NS-Attackers would likely be able to extract additional information by using highly skilled linguists with not only translation skills but and understanding of culture. A likely source of useful information that we were not able to exploit is the election officials' conversations during the configuration ceremony.

Specialization could also be leveraged in reverse-engineering the client application. Our efforts to deobfuscate and understand the application were successful over the course of a

few weeks worth of effort and were solely focused on patching the client to communicate with our server implementation. An NS-Attacker who employs highly skilled and highly experienced reverse-engineerers would likely not only be able to accomplish the same goal in significantly less time, but would also likely look for binary weaknesses which they may be able to leverage for more impact.

Lastly, the NS-Attacker's characteristic of Access is directly applicable to our server-side attacks. While we simulated privileged access to the servers, NS-Attackers are known to use both supply-chain attacks [299] as well as 0-day vulnerabilities [5, 157] as the situation dictates. Either of these approaches would provide the access needed for our server-side attacks.

CHAPTER 4

Explaining and Evaluating Operations

Of all the source documents that have become public in recent years, some of the most interesting and enigmatic are those related to NSA's ability to defeat many of the most popular Internet protocols' encryption schemes. While none of the documents suggest defeating these protective measures is trivial, the mere fact that they are exploitable means that they do not provide the protection expected from strong encryption. TLS, SSH, and IPsec are among some of the most studied protocols yet documents released by Der Spiegel indicate that the NSA has wide-reaching abilities to defeat their encryption and gain access to the plaintext content [272, 305, 308].

In this chapter, we present our 2015 research which provides a plausible technical explaination as to why the encryption of these often used protocols did not provide the security we believed it did. The shared facet among all of these breakable encryption schemes is the use of the Diffie-Hellman key exchange for negotiating the shared secret base for session keys. We examine how Diffie-Hellman is commonly implemented and deployed with these protocols and find that, in practice, it frequently offers less security than widely believed.

We propose two reasons for this unexpected vulerability. First, a surprising number of TLS servers use weak Diffie-Hellman parameters or maintain support for obsolete 1990s-era export-grade crypto. Second, the common practice of using standardized, hard-coded, or widely shared Diffie-Hellman groups has created an asymmetric vulnerability where resources invested towards "breaking" these common group can be used to break a large number of instances across a large number of implementations and endpoints.

The current best technique for attacking Diffie-Hellman relies on compromising one of the private exponents (a, b) by computing the discrete log of the corresponding public value ($g^a \bmod p$, $g^b \bmod p$). With state-of-the-art number field sieve algorithms, computing a single discrete log is more difficult than factoring an RSA modulus of the same size. However, an adversary who performs a large precomputation for a prime p can then quickly

calculate arbitrary discrete logs in that group, amortizing the cost over all targets that share this parameter. Although this fact is well known among mathematical cryptographers, it seems to have been lost among practitioners deploying cryptosystems. We exploit it to obtain the following results:

Active attacks on export ciphers in TLS. We introduce Logjam, a new attack on TLS by which a man-in-the-middle attacker can downgrade a connection to export-grade cryptography. This attack is reminiscent of the FREAK attack [33] but applies to the ephemeral Diffie-Hellman ciphersuites and is a TLS protocol flaw rather than an implementation vulnerability. We present measurements that show that as of 2015, this attack applies to 8.4% of Alexa Top Million HTTPS sites and 3.4% of all HTTPS servers that have browser-trusted certificates.

To exploit this attack, we implemented the number field sieve discrete log algorithm and carried out precomputation for two 512-bit Diffie-Hellman groups used by more than 92% of the vulnerable servers. This allows us to compute individual discrete logs in about a minute. Using our discrete log oracle, we can compromise connections to over 7% of Top Million HTTPS sites. Discrete logs over larger groups have been computed before [40], but, as far as we are aware, this is the first time they have been exploited to expose concrete vulnerabilities in real-world systems.

We were also able to compromise Diffie-Hellman for many other servers because of design and implementation flaws and configuration mistakes. These include use of composite-order subgroups in combination with short exponents, which is vulnerable to a known attack of van Oorschot and Wiener [413], and the inability of clients to properly validate Diffie-Hellman parameters without knowing the subgroup order, which TLS has no provision to communicate. We implement these attacks too and discover several vulnerable implementations.

Risks from common 1024-bit groups. We explore the implications of precomputation attacks for 768- and 1024-bit groups, which were widely used in practice and considered secure. We provide new estimates for the computational resources necessary to compute discrete logs in groups of these sizes, concluding that 768-bit groups are within range of academic teams, and 1024-bit groups may plausibly be within range of state-level attackers. In both cases, individual logs can be quickly computed after the initial precomputation.

We then examine evidence from published Snowden documents that suggests NSA may already be exploiting 1024-bit Diffie-Hellman to decrypt VPN traffic. We perform measurements to understand the implications of such an attack for popular protocols, finding that an attacker who could perform precomputations for ten 1024-bit groups could passively decrypt traffic to about 66% of IKE VPNs, 26% of SSH servers, 16% of SMTP servers, and

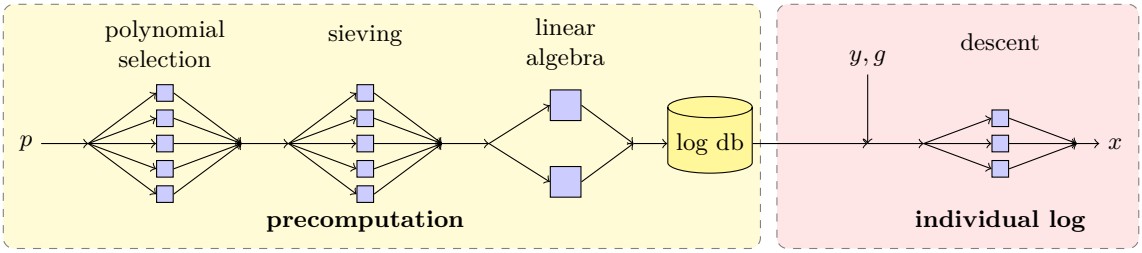

Figure 4.1: **The number field sieve algorithm for discrete log**—This algorithm consists of a precomputation stage that depends only on the prime p and a descent stage that computes individual logs. With sufficient precomputation, an attacker can quickly break any Diffie-Hellman instances that use a particular p.

24% of popular HTTPS sites.

Mitigations and lessons. As a short-term countermeasure in response to the Logjam attack, all mainstream browsers are implementing a more restrictive policy on the size of Diffie-Hellman groups they accept. We further recommend that TLS servers disable export-grade cryptography and carefully vet the Diffie-Hellman groups they use. In the longer term, we advocate that protocols migrate to stronger Diffie-Hellman groups, such as those based on elliptic curves.

4.1 Diffie-Hellman Cryptanalysis

Diffie-Hellman key exchange was the first published public-key algorithm [76]. In the simple case of prime groups, Alice and Bob agree on a prime p and a generator g of a multiplicative subgroup modulo p. Alice sends $g^a \bmod p$, Bob sends $g^b \bmod p$, and each computes a shared secret $g^{ab} \bmod p$. While there is also a Diffie-Hellman exchange over elliptic curve groups, we address only the "mod p" case.

The security of Diffie-Hellman is not known to be equivalent to the discrete log problem (except in certain groups [75, 227, 228]), but computing discrete logs remains the best known cryptanalytic attack. An attacker who can find the discrete log x from $y = g^x \bmod p$ can easily find the shared secret.

Textbook descriptions of discrete log can be misleading about the computational trade-offs, for example by balancing parameters to minimize overall time to compute a *single* discrete log. In fact, as illustrated in Figure 4.1, a single large precomputation on p can be used to efficiently break *all* Diffie-Hellman exchanges made with that prime.

The typical case Diffie-Hellman is typically implemented with prime fields and large group orders. In this case, the most efficient discrete log algorithm is the number field sieve (NFS) [163, 191, 357].[1] There is a closely related number field sieve algorithm for factoring [68, 213], and in fact many parts of the implementations can be shared. The general technique is called index calculus and has four stages with different computational properties. The first three steps are only dependent on the prime p and comprise most of the computation.

First is *polynomial selection*, in which one finds a polynomial $f(z)$ defining a number field $\mathbb{Q}(z)/f(z)$ for the computation. (For our cases, $f(z)$ typically has degree 5 or 6.) This parallelizes well and is only a small portion of the runtime.

In the second stage, *sieving*, one factors ranges of integers and number field elements in batches to find many relations of elements, all of whose prime factors are less than some bound B (called B-smooth). Modern implementations use *special-q* lattice sieving, which for each special q explores a sieving region of 2^{2I} candidates, where I is a parameter. Sieving parallelizes well since each special q is handled independently of the others, but is computationally expensive, because we must search through and attempt to factor many elements. The time for this step depends on heuristic estimates of the probability of encountering B-smooth numbers in this search; it also depends on I and on the number of special q to consider before having enough relations.

In the third stage, *linear algebra*, we construct a large, sparse matrix consisting of the coefficient vectors of prime factorizations we have found. A nonzero kernel vector of the matrix modulo the order q of the group will give us logs of many small elements. This database of logs serves as input to the final stage. The difficulty depends on q and the matrix size and can be parallelized in a limited fashion.

The final stage, *descent*, actually deduces the discrete log of the target y. We re-sieve until we can find a set of relations that allow us to write the log of y in terms of the logs in the precomputed database. This step is accomplished in three phases: an initialization phase, which tries to write the target in terms of medium-sized primes, a middle phase, in which these medium-sized primes are further sieved until they can be represented by elements in the database of known logs, and a final phase that actually reconstructs the target using the log database. Crucially, descent is the only NFS stage that involves y (or g), so polynomial selection, sieving, and linear algebra can be done once for a prime p and reused to compute the discrete logs of many targets.

Figure 4.2 shows the running time of the NFS precomputation algorithm. This is obtained

[1] Recent spectacular advances in discrete log algorithms have resulted in a quasi-polynomial algorithm for small-characteristic fields [25], but these advances are not known to apply to the prime fields used in practice.

$$L_p(1/3, (64/9)^{1/3}) = \exp\left((1.923 + o(1))(\log p)^{1/3}(\log\log p)^{2/3}\right)$$

Figure 4.2: **Running time of the NFS Algorithm**

by tuning many parameters, including the degree of f, the sieving region parameter I, and, most importantly, the smoothness bound B. Early articles (e.g. [163]) encountered technical difficulties with descent and reported that the complexity of this step would equal that of the precomputation; this may have contributed to misconceptions about the performance of the NFS for discrete logs. More recent analyses have improved the complexity of descent to $L_p(1/3, 1.442)$ [63], and later to $L_p(1/3, 1.232)$ [24], which is much cheaper than the precomputation in practice.

The numerous parameters of the algorithm allow some flexibility to reduce time on some computational steps at the expense of others. For example, sieving more will result in a smaller matrix, making linear algebra cheaper, and doing more work in the precomputation makes the final descent step easier. In Section 4.2.3, we show how exploiting these tradeoffs allows us to quickly compute 512-bit discrete logs in order to perform an effective man-in-the-middle attack on TLS.

Improperly generated groups A different family of algorithms runs in time exponential in group order, and they are practical even for large primes when the group order is small or has many small prime factors. To avoid this, most implementations use "safe" primes, which have the property that $p - 1 = 2q$ for some prime q, so that the only possible subgroups have order 2, q, or $2q$. However, as we show in Section 4.2.5, improperly generated groups are sometimes used in practice and susceptible to attack.

The baby-step giant-step [368] and Pollard rho [335] algorithms both take $\sqrt{q}$ time to compute a discrete log in any (sub)group of order q, while Pollard lambda [335] can find $x < t$ in time $\sqrt{t}$. These parallelize well [412], and precomputation can speed up individual log calculations. If the factorization of the subgroup order q is known, one can use any of the above algorithms to compute the discrete log in each subgroup of order $q_i^{e_i}$ dividing q, and then recover x using the Chinese remainder theorem. This is the Pohlig-Hellman algorithm [334], which costs $\sum_i e_i\sqrt{q_i}$ using baby-step giant-step or Pollard rho.

Standard primes Generating primes with special properties can be computationally burdensome, so many implementations use fixed or standardized Diffie-Hellman parameters. A prominent example is the Oakley groups [319], which give "safe" primes of length 768 (Oakley Group 1), 1024 (Oakley Group 2), and 1536 (Oakley Group 5). These groups were published in 1998 and have been used for many applications including IKE, SSH, and Tor.

Source	Popularity	Prime

Apache	82%	`9fdb8b8a004544f0045f1737d0ba2e0b`
		`274cdf1a9f588218fb435316a16e3741`
		`71fd19d8d8f37c39bf863fd60e3e3006`
		`80a3030c6e4c3757d08f70e6aa871033`

mod_ssl	10%	`d4bcd52406f69b35994b88de5db89682`
		`c8157f62d8f33633ee5772f11f05ab22`
		`d6b5145b9f241e5acc31ff090a4bc711`
		`48976f76795094e71e7903529f5a824b`

(*others*)	8%	(463 distinct primes)

Table 4.1: **Top 512-bit DH primes for TLS**—8.4% of Alexa Top 1M HTTPS domains allow DHE_EXPORT, of which 92.3% use one of the two most popular primes, shown here.

When primes are of sufficient strength, there seems to be no disadvantage to reusing them. However, widespread reuse of Diffie-Hellman groups can convert attacks that are at the limits of an adversary's capabilities into devastating breaks, since it allows the attacker to amortize the cost of discrete log precomputation among vast numbers of potential targets.

4.2 Attacking TLS

TLS supports Diffie-Hellman as one of several possible key exchange methods, and about two-thirds of popular HTTPS sites allow it, most commonly using 1024-bit primes. However, a smaller number of servers also support legacy "export-grade" Diffie-Hellman using 512-bit primes that are well within reach of NFS-based cryptanalysis. Furthermore, for both normal and export-grade Diffie-Hellman, the vast majority of servers use a handful of common groups.

In this section, we exploit these facts to construct a novel attack against TLS, which we call the Logjam attack. First, we perform NFS precomputations for the two most popular 512-bit primes on the web, so that we can quickly compute the discrete log for any key-exchange message that uses one of them. Next, we show how a man-in-the-middle, so armed, can attack connections between popular browsers and any server that allows export-grade Diffie-Hellman, by using a TLS protocol flaw to downgrade the connection to export-strength and then recovering the session key. We find that this attack with our precomputations can compromise about 7.8% of HTTPS servers among Alexa Top Million domains.

4.2.1 TLS and Diffie-Hellman

The TLS handshake begins with a negotiation to determine the crypto algorithms used for the session. The client sends a list of supported ciphersuites (and a random nonce *cr*) within the ClientHello message, where each ciphersuite specifies a key exchange algorithm and other primitives. The server selects a ciphersuite from the client's list and signals its selection in a ServerHello message (containing a random nonce *sr*).

TLS specifies ciphersuites supporting multiple varieties of Diffie-Hellman. Textbook Diffie-Hellman with unrestricted strength is called "ephemeral" Diffie-Hellman, or DHE, and is identified by ciphersuites that begin with `TLS_DHE_*`.[2] In DHE, the server is responsible for selecting the Diffie-Hellman parameters. It chooses a group (p, g), computes g^b, and sends a ServerKeyExchange message containing a signature over the tuple (cr, sr, p, g, g^b) using the long-term signing key from its certificate. The client verifies the signature and responds with a ClientKeyExchange message containing g^a.

To ensure agreement on the negotiation messages, and to prevent downgrade attacks [415], each party computes the TLS master secret from g^{ab} and calculates a MAC of its view of the handshake transcript. These MACs are exchanged in a pair of Finished messages and verified by the recipients. Thereafter, client and server start exchanging application data, protected by an authenticated encryption scheme with keys also derived from g^{ab}.

To comply with 1990s-era U.S. export restrictions on cryptography, SSL 3.0 and TLS 1.0 supported reduced-strength DHE_EXPORT ciphersuites that were restricted to primes no longer than 512 bits. In all other respects, DHE_EXPORT protocol messages are identical to DHE. The relevant export restrictions are no longer in effect, but many libraries and servers maintain support for backwards compatibility. Many TLS servers are still configured with two groups: a strong 1024-bit group for regular DHE key exchanges and a 512-bit group for legacy DHE_EXPORT. This has been considered safe because most modern TLS clients do not offer or accept DHE_EXPORT ciphersuites.

To understand how HTTPS servers in the wild use Diffie-Hellman, we modified the ZMap [86] toolchain to offer DHE and DHE_EXPORT ciphersuites and scanned TCP/443 on both the full public IPv4 address space and the Alexa Top 1M domains. The scans took place in March 2015. Of 539,000 HTTPS sites among Top 1M domains, we found that 68.3% supported DHE and 8.4% supported DHE_EXPORT. Of 14.3 million IPv4 HTTPS servers with browser-trusted certificates, 23.9% supported DHE and 4.9% DHE_EXPORT.

While the TLS protocol allows servers to generate their own Diffie-Hellman parameters,

[2] TLS also supports a rarely used "static" Diffie-Hellman format, where the server's key exchange value is fixed and contained in its certificate. New ciphersuites that use elliptic curve Diffie-Hellman (ECDHE) are gaining in popularity, but we focus exclusively on the traditional prime field variety.

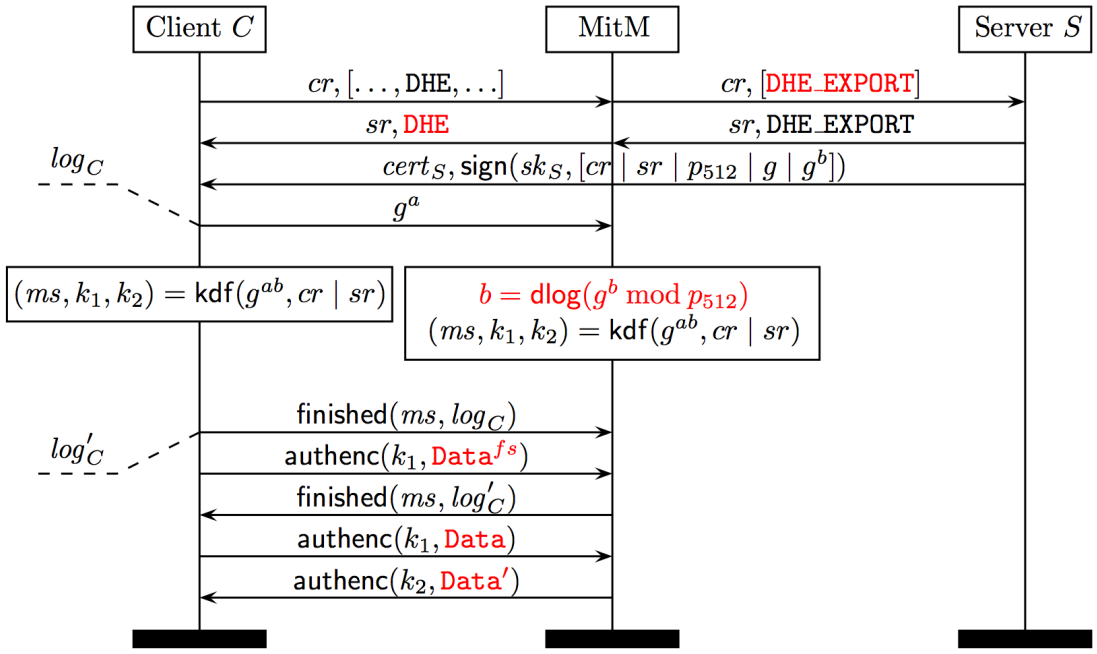

Figure 4.3: **The Logjam attack** — A man-in-the-middle can force TLS clients to use export-strength DH with any server that allows DHE_EXPORT. Then, by finding the 512-bit discrete log, the attacker can learn the session key and arbitrarily read or modify the contents. $\texttt{Data}^{fs}$ refers to False Start [210] application data that some TLS clients send before receiving the server's Finished message.

the overwhelming majority use one of a handful of primes. As shown in Table 4.1, just two 512-bit primes account for 92.3% of Alexa Top 1M domains that support DHE_EXPORT, and 92.5% of all servers with browser-trusted certificates that support DHE_EXPORT. (Non-export DHE follows a similar distribution with longer primes.) The most popular 512-bit prime was hard-coded into many versions of Apache. Introduced in 2005 with Apache 2.1.5, it was used until 2.4.7, which disabled export ciphersuites. We found it in use by about 564,000 servers with browser-trusted certificates. The second most popular 512-bit prime is the default used for DHE_EXPORT when using `mod_ssl`. It was introduced in version 2.3.0 in 1999. We found it in use by about 89,000 servers with browser-trusted certificates.

4.2.2 Active Downgrade to Export-Grade DHE

Given the widespread use of these primes, an attacker with the ability to compute discrete logs in 512-bit groups could efficiently break DHE_EXPORT handshakes for about 8% of Alexa Top 1M HTTPS sites, but modern browsers never negotiate export-grade ciphersuites. To circumvent this, we show how an attacker who can compute 512-bit discrete logs *in real*

time can downgrade a regular DHE connection to use a DHE_EXPORT group, and thereby break both the confidentiality and integrity of application data.

The attack, which we call Logjam, is depicted in Figure 4.3 and relies on a flaw in the way TLS composes DHE and DHE_EXPORT. When a server selects DHE_EXPORT for a handshake, it proceeds by issuing a signed ServerKeyExchange message containing a 512-bit p_{512}, but the structure of this message is identical to the message sent during standard DHE ciphersuites. Critically, the signed portion of the server's message fails to include any indication of the specific ciphersuite that the server has chosen. Provided that a client offers DHE, an active attacker can rewrite the client's ClientHello to offer a corresponding DHE_EXPORT ciphersuite accepted by the server and remove other ciphersuites that could be chosen instead. The attacker rewrites the ServerHello response to replace the chosen DHE_EXPORT ciphersuite with a matching non-export ciphersuite and forwards the ServerKeyExchange message to the client as is. The client will interpret the export-grade tuple (p_{512}, g, g^b) as valid DHE parameters chosen by the server and proceed with the handshake. The client and server have different handshake transcripts at this stage, but an attacker who can compute b in close to real time can then derive the master secret and connection keys to complete the handshake with the client, and then freely read and write application data pretending to be the server.

There are two remaining challenges in implementing this active downgrade attack. The first is to compute individual discrete logs in close to real time, and the second is to delay handshake completion until the discrete log computation has had time to finish. We address these in the next subsections.

Comparison with previous attacks Logjam is reminiscent of the recent FREAK [33] attack, in which an attacker downgrades a regular RSA key exchange to one that uses export-grade 512-bit ephemeral RSA keys, relying on a bug in several TLS client implementations. The attacker then factors the ephemeral key to hijack future connections that use the same key. The cryptanalysis takes several hours on commodity hardware and is usable until the server generates a fresh ephemeral RSA key (typically when it restarts).

In contrast, Logjam is due to a protocol flaw in TLS, not an implementation bug. From a client perspective, the only defense is to reject small primes in DHE handshakes. (Prior to this work, most popular browsers accepted p of size ≥ 512 bits.) Logjam affects fewer servers than FREAK, but, as we shall see, the cost per compromised connection is far lower, since the precomputation for each 512-bit group can be used indefinitely against all servers that use that group, and since each individual discrete log only takes about a minute.

Logjam and FREAK both follow the same pattern as other cross-protocol attacks discov-

ered in TLS. As early as SSL 3.0, Schneier and Wagner noted a related vulnerability that they called *key exchange rollback* [415]. Mavrogiannopoulos et al. showed how explicit-curve ECDHE handshakes could be confused with DHE handshakes [229]. All these attacks could be prevented by additionally signing the ciphersuite in the ServerKeyExchange message. We expect that TLS 1.3 will fix this protocol flaw. More generally, Logjam can also be interpreted as a backwards compatibility attack [181] where one party uses only strong cryptography but the other supports both strong and weak ciphersuites.

4.2.3 512-bit Discrete Log Computations

We modified CADO-NFS [22] to implement the number field sieve discrete log algorithm from Section 4.1 and applied it to three 512-bit primes, including the top two DHE_EXPORT primes shown in Table 4.1. Precomputation took 7 days for each prime, after which computing individual logs took a median of 70 seconds. We list the runtime for each stage of the computation below. The times were about the same for each prime.

Precomputation As illustrated in Figure 4.1, the precomputation phase includes the polynomial selection, sieving, and linear algebra steps. For this precomputation, we deliberately sieved more than strictly necessary. This enabled two optimizations: first, with more relations obtained from sieving, we eventually obtain a larger database of known logs, which makes the descent faster. Second, more sieving relations also yield a smaller linear algebra step, which is desirable because sieving is much easier to parallelize than linear algebra.

For the polynomial selection and sieving steps, we used idle time on 2000–3000 CPU cores in parallel, of which most CPUs were Intel Sandy Bridge. Polynomial selection ran for about 3 hours, which in total corresponds to 7,600 core-hours. Sieving ran for 15 hours, corresponding to 21,400 core-hours. This sufficed to collect 40,003,519 relations of which 28,372,442 were unique, involving 15,207,865 primes of at most 27 bits (hence bound B from Section 4.1 is 2^{27}).

From this data set, we obtained a square matrix with 2,157,378 rows and columns, with 113 nonzero coefficients per row on average. We solved the corresponding linear system on a 36-node cluster with two 8-core Intel Xeon E5-2650 CPUs per node, connected with Infiniband FDR. We used the block Wiedemann algorithm [67, 401] with parameters $m = 18$ and $n = 6$. Using the unoptimized implementation from CADO-NFS [22] for linear algebra over GF(p), the computation finished in 120 hours, corresponding to 60,000 core-hours. We expect that optimizations could bring this cost down by at least a factor of three.

In total, the wall-clock time for each precomputation was slightly over one week. Each

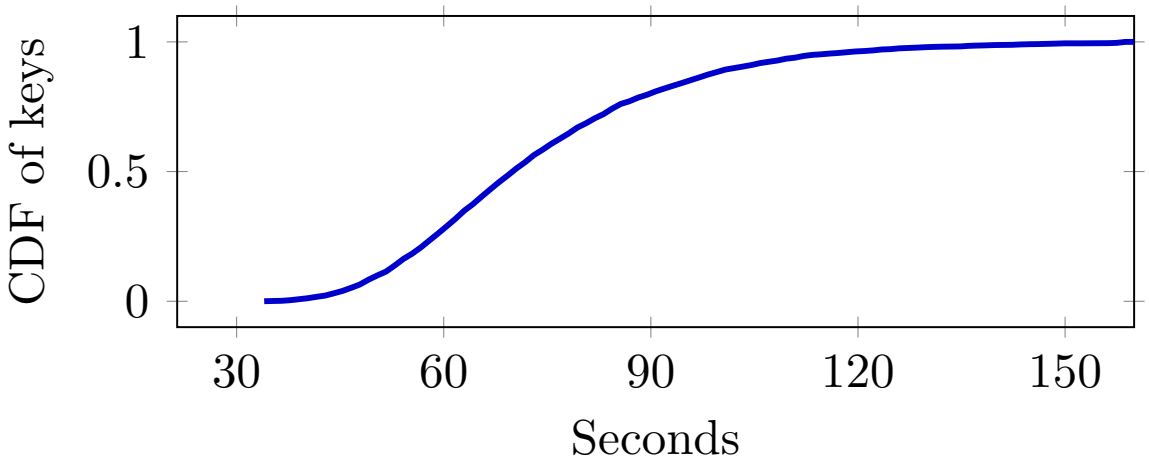

Figure 4.4: **Individual discrete log time for 512-bit DH**—After a week-long precomputation for each of the two top export-grade primes (see Table 4.1), we can quickly break any key exchange that uses them. Here we show times for computing 3,500 individual logs; the median is 70 seconds.

resulting database of known logs for the descent occupies about 2.5 GB in ASCII format.

Descent Once this precomputation was finished, we were able to run the final descent step to compute individual discrete logs in about a minute for targets in each of these groups. In order to save time on individual computations, we implemented a client-server architecture using the ZeroMQ messaging library. The server maintains the precomputed data in RAM and returns logs for values passed to it by clients.

We implemented the descent calculation in a mix of Python and C. The first and second stages are parallelized and run sieving in C, and the final discrete log is deduced in Python. We ran the server on a machine with two 18-core Intel Xeon E5-2699 CPUs and 128 GB of RAM. On average, computing individual logs took about 70 seconds, but the time varied from 34 to 206 seconds (see Figure 4.4). This is divided between about 20 seconds for descent initialization and the remainder on the middle phase. Further optimizations—such as more effective parallelization on the middle phase or additional sieving—should bring the median time well below a minute.

For purposes of comparison, a single 512-bit RSA factorization using the CADO-NFS implementation takes about eight days of wall-clock time on the computer used for the descent, and about three hours parallelized across 1,800 cores of Amazon EC2 `c4.8xlarge` instances.

4.2.4 Active Attack Implementation

We implemented a man-in-the-middle network attacker that sits between a TLS client (web browser) and any server that supports DHE_EXPORT and uses the most common 512-bit Apache group. Our implementation follows the message sequence in Figure 4.3: it downgrades the connection towards the server, computes the session keys, and takes over the connection towards the client by impersonating the server.

The main challenge is to compute the shared secret g^{ab} before the handshake completes in order to forge a Finished message from the server. With our descent implementation, the computation takes an average of 70 seconds, but there are several ways an attacker can work around this delay:

Non-browser clients. Different TLS clients impose different time limits for the handshake, after which they kill the connection. Command-line clients such as `curl` and `git` often run unattended, so they have long or no timeouts, and we can hijack their connections without difficulty.

TLS warning alerts. Web browsers tend to have shorter timeouts, but we can keep their connections alive by sending TLS warning alerts, which are ignored by the browser but reset the handshake timer. For example, this allows us to keep Firefox's TLS connections alive indefinitely. (Other browsers we tested close the connection after a minute.) Although the victim connection still takes much longer than usual, the attacker might choose to compromise a request for a background resource that does not delay rendering the page.

Ephemeral key caching. Many TLS servers do not use a fresh value b for each connection, but instead compute g^b once and reuse it for multiple negotiations. Without enabling the `SSL_OP_SINGLE_DH_USE` option, OpenSSL will reuse g^b for the lifetime of a TLS context. While both Apache and Nginx internally apply this option, certain load balancers, such as stud [395], do not. The F5 BIG-IP load balancers and hardware TLS frontends will reuse g^b unless the "Single DH" option is checked [416]. Microsoft Schannel caches g^b for two hours—this setting is hard-coded. For these servers, an attacker can compute the discrete log of g^b from one connection and use it to attack later handshakes, avoiding the need to do the computation online. By randomly sampling IPv4 hosts serving browser-trusted certificates that support DHE, we found that 17% reused g^b at least once over the course of 20 handshakes, and that 15% only used one value. However, for DHE_EXPORT, only 0.1% reused g^b, likely because Microsoft IIS does not support 512-bit export ciphersuites.

TLS False Start. Even when clients enforce shorter timeouts and servers do not reuse values for b, the attacker can still break the confidentiality of user requests if the client supports the TLS False Start extension [210]. This extension reduces connection latency by

having the client send early application data (such as an HTTP request) without waiting for the server's Finished message to arrive. Recent versions of Chrome, Internet Explorer, and Firefox implement False Start, but their policies on when to enable it vary between versions. Firefox 35, Chrome 41, and Internet Explorer (Windows 10) send False Start data with DHE. In these cases, a man-in-the-middle can record the handshake and decrypt the False Start payload at leisure. We note that this initial data sent by a browser often contains sensitive user authentication information, such as passwords and cookies.

4.2.5 Other Weak and Misconfigured Groups

In our scans, we found several other exploitable security issues in the DHE configurations used by TLS servers.

512-bit primes in non-export DHE We found 2,631 servers with browser-trusted certificates (and 118 in the Top 1M domains) that used 512-bit or weaker primes for non-export DHE. In these instances, active attacks may be unnecessary. If a browser negotiates a DHE ciphersuite with one of these servers, a *passive* eavesdropper can later compute the discrete log and obtain the TLS session keys for the connection. An active attack may still be necessary when the client's ordering of ciphersuites would result in the server not selecting DHE. In this case, as in the DHE_EXPORT downgrade attack, an active attacker can force the server to choose a vulnerable DHE ciphersuite.

As a proof-of-concept, we implemented a passive eavesdropper for regular DHE connections and used it to decrypt test connections to www.fbi.gov. Until April 2015, this server used the default 512-bit DH group from OpenSSL, which was the third group for which we performed the NFS precomputation. The website no longer supports DHE.

Attacks on composite-order subgroups Failure to generate Diffie-Hellman primes according to best practices can result in devastating attacks. Not every TLS server uses "safe" primes. Out of approximately 70,000 distinct primes seen across both export and non-export TLS scans, 4,800 were not safe, meaning that $(p-1)/2$ was composite. (Incidentally, we also found 9 composite p.) These groups are not necessarily vulnerable, as long as g generates a group with at least one sufficiently large subgroup order to rule out the Pohlig-Hellman algorithm as an attack.

In some real-life configurations, however, choosing such primes can lead to an attack. For efficiency reasons, some implementations use ephemeral keys g^x with a short exponent x; commonly suggested sizes for x are as small as 160 or 224 bits, intended to match the

estimated strength of a 1024- or 2048-bit group. For safe p, such exponent lengths are not known to decrease security, as the most efficient attack will be the Pollard lambda algorithm. But if the order of the subgroup generated by g has small factors, they can be used to recover information about exponents. From a subset of factors $\{q_1^{e_1} \dots q_k^{e_k}\}$ with $\prod_i q_i^{e_i} = z$, Pohlig-Hellman can recover $x \mod z$ in time $\sum_i e_i \sqrt{q_i}$. If $x \leq z$, this suffices to recover x. If not, Pollard lambda can use this information to recover x in time $\sqrt{x/z}$. This attack was first described as hypothetical by van Oorschot and Wiener [413].

To see if TLS servers in the wild were vulnerable to this attack, we tested various non-safe primes found in our scans. For each non-safe prime p, we opportunistically factored $p-1$ using Bernstein's batch method [29]. We then ran the GMP-ECM implementations of the Pollard $p-1$ algorithm and the ECM factoring methods [432] for 5 days parallelized across 28 cores and discovered 36,447 prime factors.

We then examined the generators g used with each prime p. We classified a tuple (p,g,y) sent by a server as interesting if the prime factorization of $p-1$ had revealed prime factors of the order of g, and ordered them by the estimated work required using Pohlig-Hellman and Pollard lambda to recover a target private exponent x of length ranging from 64 to 256 bits. There were 753 (p,g) pairs where we knew factors of the subgroup generated by g; these had been used for 40,903 connections across all of our scans.

We implemented the van Oorschot and Wiener algorithm in Sage [392] using a parallel Pollard rho implementation that we wrote in C using the GMP library. We used the distinguished points method for collision detection; for a prime known in advance, this implementation can be arbitrarily sped up by precomputing a table of distinguished points.

We computed partial information about the server secret exponent used in 460 exchanges and were able to recover the whole exponent used by 159 different hosts, 53 of which authenticated with valid browser-trusted certificates. In all cases, the vulnerable hosts used 512-bit prime moduli; three of them used 160-bit exponents and the rest used 128 bits. The order of the largest-order subgroup ranged from 46 bits (which finishes in seconds) to 81 bits (which took between 50 and 176 hours) implementation. The Pollard lambda calculations used interval width varying from 40 to 70 bits.

Our computations would have allowed us to hijack connections to a variety of vulnerable TLS servers, including web interfaces for VPN devices (48 hosts), communications software (21 hosts), web conferencing servers (27 hosts), and FTP servers (6 hosts). As a proof-of-concept, we modified our man-in-the-middle attacker of Section 4.2.3 to impersonate a vulnerable server and capture user credentials. Compared to an attack using NFS, we could compute the discrete log with a delay hardly noticeable for browser users.

Misconfigured groups The Digital Signature Algorithm (DSA) [252] uses primes p such that $p-1$ has a large prime factor q and g generates only a subgroup of order q. When using properly generated DSA parameters, these groups are secure for use in Diffie-Hellman key exchanges. Notably, DSA groups are hard-coded in Java's `sun.security.provider` package and are used by default in many Java-based TLS servers. However, some servers in our scans used Java's DSA primes as p but mistakenly used the DSA group order q in the place of the generator g. We found 5,741 hosts misconfigured this way.

This substitution of q for g is likely due to a usability problem: the canonical ASN.1 representation of Diffie-Hellman key exchange parameters (coming from PKCS#3) is a sequence (p,g), while that of DSA parameters (coming from PKIX) is (p,q,g); we conjecture that the confusion between these formats led to a simple programming error.

In a DSA group, the subgroup generated by q is likely to have many small prime factors in its order, since for p generated according to [252], $(p-1)/q$ is a random integer. For Java's `sun.security.provider` 512-bit prime, using q as a generator leaks 290 bits of information about exponents at a cost of roughly 2^{40} operations. Luckily, since the provider generates exponents of length $\max(n/2, 384)$ for n-bit p, this does not suffice to recover a full exponent. Still, this misconfiguration bug results in a significant loss of security and serves as a cautionary tale for programmers.

4.3 State-Level Threats to DH

The previous sections demonstrate the existence of practical attacks against the Diffie-Hellman key exchange as currently used by TLS. However, these attacks rely on the ability to downgrade connections to export-grade cryptography or on the use of unsafe parameters. In this section we address the following question: how secure is Diffie-Hellman in broader practice, as used in other protocols that do not suffer from a downgrade, and when applied with stronger groups?

To answer this question we must first examine how the number field sieve for discrete log scales to 768- and 1024-bit groups. As we argue below, 768-bit groups, which are still in relatively widespread use, are now within reach for academic computational resources, and performing precomputations for a small number of 1024-bit groups is plausibly within the resources of state-level attackers. The precomputation would likely require special-purpose hardware, but would not require any major algorithmic improvements beyond what is known in the academic literature. We further show that even in the 1024-bit case, the descent time—necessary to solve any specific discrete log instance within a common group—would be fast enough to break individual key exchanges in close to real time.

	Sieving		Linear Algebra		Descent	
	$\log_2 B$	core-years	rows	core-years	core-time	
RSA-512	29	0.5	4.3M	0.33		CADO-NFS defaults
DH-512	27	2.5	2.1M	7.7	10 mins	Conducted computation
RSA-768	37	800	250M	100		[204] with less sieving
DH-768 (*est*)	35	8,000	150M	28,500	2 days	[40, 204] & experimentation
DH-768 [205]	36	4,000	24M	920	43 hours	Data from [205, Table 1].
RSA-1024	42	1,000,000	8.7B	120,000		Fig 4.2
DH-1024	40	10,000,000	5.2B	35,000,000	30 days	Fig 4.2 & experimentation

Table 4.2: **Estimating costs for factoring and discrete log**—For sieving, we give two important parameters: the number of bits of the smoothness bound `B` and the sieving region parameter `I`. For linear algebra, all costs for DH are for safe primes; for DSA primes with q of 160 bits, this should be divided by 6.4 for 1024 bits, 4.8 for 768 bits, and 3.2 for 512 bits.

In light of these results, we examine several standard Internet security protocols—IKE, SSH, and TLS—to determine the vulnerability of their key exchanges to attacks by resourceful attackers. Although the cost of the precomputation for a 1024-bit group is several times higher than for an RSA key of equal size, we observe that a one-time investment could be used to attack millions of hosts, due to widespread reuse of the most common Diffie-Hellman parameters. Unfortunately, our measurements also indicate that it may be very difficult to sunset the use of fixed 1024-bit Diffie-Hellman groups that have long been embedded in standards and implementations.

Finally, we apply this new understanding to a set of recently published documents leaked by Edward Snowden [383] to evaluate the hypothesis that the National Security Agency has *already* implemented such a capability. We show that this hypothesis is consistent with the published details of the intelligence community's cryptanalytic capabilities, and, indeed, matches the known capabilities more closely than other proposed explanations, such as novel breaks on RC4 or AES. We believe that this analysis may help shed light on unanswered questions about how NSA may be gaining access to VPN, SSH, and TLS traffic.

4.3.1 Scaling NFS to 768- and 1024-bit DH

Estimating the cost for discrete log cryptanalysis at longer key sizes is far from straightforward, due in part to the complexity of parameter tuning and to tradeoffs between the sieving and linear algebra steps, which have very different computational characteristics. (Much more attention has gone to understanding 1024-bit factorization, but, even there, many published estimates are crude extrapolations of the asymptotic complexity.) We attempt

estimates for 768- and 1024-bit discrete log based on the existing literature and our own experiments, but further work is needed for greater confidence, particularly for the 1024-bit case. We summarize all the costs, measured or estimated, in Table 4.2.

DH-768: Feasible with academic power For the 768-bit case, we base our estimates on the recent discrete log record at 596 bits [40] and the integer factorization record of 768 bits from 2009 [204]. While the algorithms for factorization and discrete log are similar, the discrete log linear algebra stage is many times more difficult, as the matrix entries are no longer Boolean. We can reduce overall time by sieving more, thus generating a smaller input matrix to the linear algebra step. Since sieving parallelizes better than linear algebra, this tradeoff is desirable for large inputs.

A 596-bit factorization takes about 5 core-years, most of it spent on sieving. In comparison, the record 596-bit discrete log effort tuned parameters such that they spent 50 core-years on sieving. This reduced their linear algebra calculation to 80 core-years. We used this same strategy in our 512-bit experiments in Section 4.2.3.

Similarly, the 768-bit RSA factoring record spent more time on sieving in order to save time on the linear algebra step. The cost of sieving was around 1500 core-years, and the matrix that was produced had 200M rows and columns. As a result, the linear algebra took 150 core-years, but taking algorithmic improvements since 2009 into account and optimizing for the total time,[3] we estimate that factoring an RSA-768 integer would take 900 core-years in total.

For a 768-bit discrete log, we can expect that ten times as much sieving as the RSA case would reduce the matrix to around 150M rows. We extrapolate from experiments with existing software that this linear algebra would take 28,500 core-years, for a total of 36,500 core-years. This is within reach by computing power available to academics.

The descent step takes relatively little time. We experimented with both CADO-NFS and a new implementation with GMP-ECM based on the early-abort strategy described in [31]. Using these techniques, the initial descent phase took an average of around 1 core-day. The remaining phase uses sieving much as in the precomputation; extrapolating from experiments, the rest of the descent should take at most 1 core-day. In total, after precomputation, the cost of a single 768-bit discrete log computation is around 2 core-days and is easily parallelizable.

DH-1024: Plausible with state-level resources Experimentally extrapolating sieving parameters to the 1024-bit case is difficult due to the tradeoffs between the steps of the

[3] We would lower the smoothness bounds compared to the parameters in [204].

algorithm and their relative parallelism. The prior work proposing parameters for factoring a 1024-bit RSA key is thin: [203] proposes smoothness bounds of 42 bits, but the proposed value of the sieving region parameter `I` is clearly too small, giving too few smooth results per sieving subtask. Since no publicly available software can currently deal with values of `I` larger than those proposed, we could not experimentally update the estimates of this paper with more relevant parameter choices.

Without better parameter choices, we resort to extrapolating from asymptotic complexity. For the number field sieve, the complexity is $\exp\left((k+o(1))(\log N)^{1/3}(\log\log N)^{2/3}\right)$, where N is the integer to factor or the prime modulus for discrete log, and k is an algorithm-specific constant. This formula is inherently imprecise, since the $o(1)$ in the exponent can hide polynomial factors. This complexity formula, with $k = 1.923$, describes the overall time for both discrete log and factorization, which are both dominated by sieving and linear algebra in the precomputation. The space complexity (the size of the matrix in memory) is the square root of this function, i.e., the same function, taking $k = 0.9615$. Discrete log descent has a complexity of the same form as well; [24, Chapter 4] gives $k = 1.232$, using an early-abort strategy similar to the one in [31] mentioned above.

Evaluating the formula for 768- and 1024-bit N gives us estimated multiplicative factors by which time and space will increase from the 768- to the 1024-bit case. For precomputation, the total time complexity will increase by a factor of 1220, while space complexity will increase by a factor of 35. These are valid for both factorization and discrete log, since they have the same asymptotic behavior. Hence, for DH-1024, we get a total cost for the precomputation of about 45M core-years. The time complexity for each individual log after the precomputation should be multiplied by 95. This last number does not correspond to what we observed in practice; we attribute that to the fact that the descent step has been far less studied both in theory and in practice compared to the other steps.

For 1024-bit descent, we experimented with our early-abort implementation to inform our estimates for descent initialization, which should dominate the individual discrete log computation. For a random target in Oakley Group 2, initialization took 22 core-days, yielding a few primes of at most 130 bits to be descended further. In twice this time, we reached primes of about 110 bits. At this point, we were certain to have bootstrapped the descent, and could continue down to the smoothness bound in a few more core-days if proper sieving software were available. Thus we estimate that a 1024-bit descent would take about 30 core-days, once again easily parallelizable.

Costs in hardware Although 45M core-years is a huge computational effort, it is not necessarily out of reach for a nation state. Moreover, at this scale, significant cost savings

could be realized by developing application-specific hardware.

Sieving is a natural target for hardware implementation. To our knowledge, the best prior description of an ASIC implementation of 1024-bit sieving is the 2007 work of Geiselmann and Steinwandt [151]. In the following, we update their estimates for modern techniques and adjust parameters for discrete log. We increase their chip count by a factor of ten to sieve more and save on linear algebra as above, giving an estimate of 3M chips to complete sieving in one year. Shrinking the dies from the 130 nm technology node used in the paper to a more modern size reduces costs, as transistors are cheaper at newer technologies. With standard transistor costs and utilization, this would cost about $2 per chip to manufacture, after fixed design and tape-out costs of roughly $2M [218]. This suggests that an $8M investment would buy enough ASICs to complete the DH-1024 sieving precomputation in one year. Since a step of descent uses sieving, the same hardware could likely be reused to speed calculations of individual logs.

Estimating the financial cost for the linear algebra is more difficult, since there has been little work on designing chips that are suitable for the larger fields involved in discrete log. To derive a rough estimate, we can begin with general purpose hardware and the core-year estimate from Table 4.2. The Titan supercomputer [312]—at 300,000 CPU cores, currently the most powerful supercomputer in the U.S.—would take 117 years to complete the 1024-bit linear algebra stage. Titan was constructed in 2012 for $94M, suggesting a cost of $11B in supercomputers to finish this step in a year. In the context of factorization, moving linear algebra from general purpose CPUs to ASICs has been estimated to reduce costs by a factor of 80 [150]. If we optimistically assume that a similar reduction can be achieved for discrete log, the hardware cost to perform the linear algebra for DH-1024 in one year is plausibly on the order of hundreds of millions of dollars.

To put this dollar figure in context, the FY 2012 budget for the U.S. Consolidated Cryptologic Program (which includes the NSA) was $10.5 billion[4] [254]. The agency's classified 2013 budget request, which prioritized investment in "groundbreaking cryptanalytic capabilities to defeat adversarial cryptography and exploit internet traffic," included notable $100M increases in two programs [254]: "cryptanalytic IT services" (to $247M), and a cryptically named "cryptanalysis and exploitation services program C" (to $360M). NSA's leaked strategic plan for the period called for it to "continue to invest in the industrial base and drive the state of the art for high performance computing to maintain pre-eminent cryptanalytic capability for the nation" [292].

[4]The National Science Foundation's budget was $7 billion.

4.3.2 Is NSA Breaking 1024-bit DH?

Our calculations suggest that it is plausibly within NSA's resources to have performed number field sieve precomputations for at least a small number of 1024-bit Diffie-Hellman groups. This would allow them to break any key exchanges made with those groups in close to real time. If true, this would answer one of the major cryptographic questions raised by the Edward Snowden leaks: How is NSA defeating the encryption for widely used VPN protocols?

Classified documents published by Der Spiegel [383] indicate that NSA is passively decrypting IPsec connections at significant scale. The documents do not describe the cryptanalytic techniques used, but they do provide an overview of the attack system architecture. After reviewing how IPsec key establishment works, we will use the published information to evaluate the hypothesis that the NSA is leveraging precomputation to calculate discrete logs at scale.

IKE Internet Key Exchange (IKE) is the main key establishment protocol used for IPsec VPNs. There are two versions, IKEv1 [47] and IKEv2 [195], which differ in message structure but are conceptually similar. For the sake of brevity, we will use IKEv1 terminology.

Each IKE session begins with a Phase 1 handshake, in which the client and server select a Diffie-Hellman group from a small set of standardized parameters and perform a key exchange to establish a shared secret. The shared secret is combined with other cleartext values transmitted by each side, such as nonces and cookies, to derive a value called SKEYID. IKE provides several authentication mechanisms, including symmetric pre-shared keys (PSK); when IKEv1 is authenticated with a PSK, this value is incorporated into the derivation of SKEYID.

The resulting SKEYID is used to encrypt and authenticate a Phase 2 handshake. Phase 2 establishes the parameters and key material, KEYMAT, for a cryptographic transport protocol used to protect subsequent traffic, such as Encapsulating Security Payload (ESP) [199] or Authenticated Header (AH) [198]. In some circumstances, this phase includes an additional round of Diffie-Hellman. Ultimately, KEYMAT is derived from SKEYID, additional nonces, and the result of the optional Phase 2 Diffie-Hellman exchange.

NSA's VPN exploitation process The documents published by Der Spiegel describe a system named TURMOIL that is used to acquire VPN traffic and defeat its encryption. Nothing within the source documents indicate that message injection or a man-in-the-middle attack against IKE or IPsec is required which leads us to believe that it is a passive defeat.

Figure 4.5, an excerpt from one of the documents [302], illustrates the flow of information through the TURMOIL system.

The initial phases of the attack involve acquiring both IKE and ESP payloads as well as determining whether the traffic matches any tasked selector [303]. If so, TURMOIL transmits the complete IKE handshake and may transmit a small amount of ESP ciphertext to NSA's Cryptanalysis and Exploitation Services (CES) [265, 303] via a secure tunnel. Within CES, a specialized VPN Attack Orchestrator (VAO) system manages a collection of high-performance grid computing resources located at NSA Headquarters and in a data center at Oak Ridge National Laboratory, which perform the computation required to generate the ESP session key [272,274,302]. The VAO coordinates with a database CORALREEF, which stores cryptographic values including a set of known PSKs and the resulting "recovered" ESP session keys [272, 302, 308].

The ESP traffic itself is buffered for up to 15 minutes [295], until CES can respond with the recovered ESP keys if they were generated correctly. Once keys have been returned, the ESP traffic is decrypted via hardware accelerators [132] or in software [305, 306]. From this point, decrypted VPN traffic is reinjected into the TURMOIL processing infrastructure and passed to other systems for storage and analysis [305]. The documents indicate that NSA is recovering ESP keys at large scale, with a target of 100,000 per hour [295].

Evidence for a Discrete Log Defeat While the ability to decrypt VPN traffic does not by itself indicate a Diffie-Hellman defeat, there are several features of IKE and the VAO's operation that support this hypothesis. First, the IKE protocol has been extensively analyzed [45, 232], and is not believed to be exploitable in standard configurations by passive attackers. In order to recover the session keys for the ESP or AH protocols, the attacker must at minimum recover the SKEYID generated by the Phase 1 exchange. Absent a vulnerability in the key derivation function or transport encryption, this requires the attacker to recover a shared Diffie-Hellman secret after passively observing an IKE handshake.

Second, while IKE is designed to support a range of Diffie-Hellman groups, our Internet-wide scans (Section 4.3.3) show that the vast majority of IKE systems select one particular 1024-bit DH group (Oakley Group 2) even when offered stronger groups.

Third, given an efficient oracle for solving the discrete logarithm problem, attacks on IKE are possible provided that the attacker can obtain the following: (1) a complete two-sided IKE transcript, including the Diffie-Hellman key shares g^a and g^b as well as the nonces and cookies transmitted by both sides of the connection, and (2) in IKEv1 only, the PSK used in deriving SKEYID. Both of the above requirements are also present in the NSA's VPN attack system. As Figure 4.5 illustrates, a hard requirement of the VAO is the need

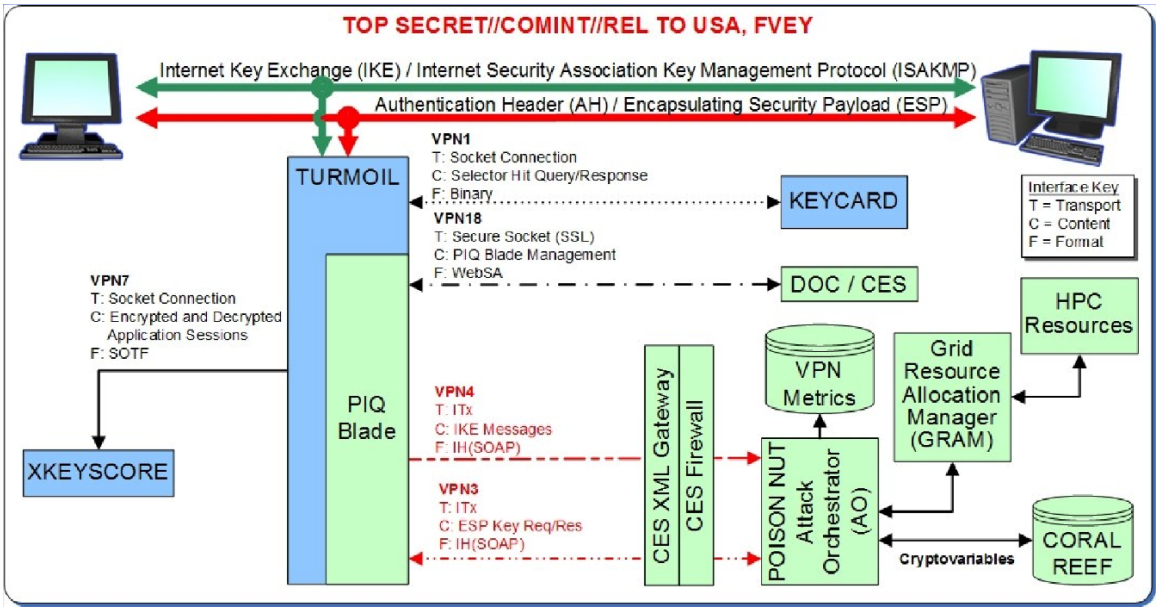

Figure 4.5: **NSA's VPN Decryption Infrastructure**—Acquired IKE handshakes are passed to CES who uses high-performance computing (HPC) resources to generate the symmetric session keys for ESP traffic [302]. For clarity, POISONNUT is a codename sometimes used in place of VAO [274].

to obtain the *complete* two-sided IKE transcript [308]. The published documents indicate that this requirement substantially increases the complexity of the attack execution, since IKE transcripts must be reassembled ("paired") whenever the interaction traverses multiple network paths [258, 265, 266, 301].

The complete attack also requires knowledge of the PSK. Several documents describe techniques for analysts to locate a PSK, including using a database of router configurations [287, 309], the CORALREEF database of known PSKs [308], previously decrypted SSH traffic which could include VPN configuration files [308], or system administrator "chatter" [309]. Additionally, NSA is willing to "[r]un attacks to recover PSK" [308].

This explanation is of course not dispositive and NSA could defeat IPsec using alternative types of attacks. Certain published NSA documents refer to software "implants" on VPN devices, indicating that the use of targeted malware is a piece of the collection strategy [308]; however, the same documents also note that decryption of the resulting traffic *does not* require IKE handshakes, and thus appears to be an alternative to the VAO attack described above. The most compelling argument for a pure cryptographic attack is the generality of the VAO approach, which appears to succeed across a broad swath of non-compromised devices with only the requirements listed above.

	Vulnerable servers, if the attacker can precompute for . . .			
	all 512-bit groups	all 768-bit groups	one 1024-bit group	ten 1024-bit groups
HTTPS Top 1M†	45,100 (8.4%)	45,100 (8.4%)	205,000 (37.1%)	309,000 (56.1%)
HTTPS Top 1M	118 (0.0%)	407 (0.1%)	98,500 (17.9%)	132,000 (24.0%)
HTTPS Trusted†	489,000 (3.4%)	556,000 (3.9%)	1,840,000 (12.8%)	3,410,000 (23.8%)
HTTPS Trusted	1,000 (0.0%)	46,700 (0.3%)	939,000 (6.56%)	1,430,000 (10.0%)
IKEv1 IPv4	–	64,700 (2.6%)	1,690,000 (66.1%)	1,690,000 (66.1%)
IKEv2 IPv4	–	66,000 (5.8%)	726,000 (63.9%)	726,000 (63.9%)
SSH IPv4	–	–	3,600,000 (25.7%)	3,600,000 (25.7%)

Table 4.3: **Estimated impact of Diffie-Hellman attacks**—We use Internet-wide scanning to estimate the number of real-world servers for which typical connections could be compromised by attackers with various levels of computational resources. For HTTPS, we provide figures with active downgrade (denoted by "†") and without. All others are passive attacks.

4.3.3 Effects of a 1024-bit Break

In this section, we use Internet-wide scanning to assess the impact of a hypothetical DH-1024 break on three popular protocols: IKE, SSH, and HTTPS. Our measurements indicate that these protocols, as they are commonly used, would be subject to widespread compromise by a state-level attacker who had the resources to invest in precomputation for a small number of common 1024-bit groups.

IKE We measured how IPsec VPNs use Diffie-Hellman in practice by scanning a 1% random sample of the public IPv4 address space for IKEv1 and IKEv2 (the protocols used to initiate an IPsec VPN connection) in May 2015. We used the ZMap UDP probe module to measure support for Oakley Groups 1 and 2 (two popular 768- and 1024-bit, built-in groups) and which group servers prefer. To test support for individual groups, we offered only the single group in question. To detect default behavior, we offered servers a variety of DH groups, with the lowest priority groups being Oakley Groups 1 and 2. When measuring server preference, we scanned with the 3DES symmetric cipher—the most commonly supported symmetric cipher in our single group scans. Because of this, the percentages we present for IKEv1 and IKEv2 are a lower bound for the number of servers that prefer Oakley Groups 1 and 2.

Of the 80K hosts that responded with a valid IKE packet, 44.2% were willing to accept an offered proposal from at least one scan. The majority of the remaining hosts responded with a `NO-PROPOSAL-CHOSEN` message regardless of our proposal. Many of these may be site-to-site VPNs that reject our source address. We consider these hosts "unprofiled" and

omit them from the results here.

We found that 31.8% of IKEv1 and 19.7% of IKEv2 servers support Oakley Group 1 (768-bit) while 86.1% and 91.0% respectively supported Oakley Group 2 (1024-bit). In our sample of IKEv1 servers, 2.6% of profiled servers preferred the 768-bit Oakley Group 1 — which is within cryptanalytic reach today for moderately resourced attackers — and 66.1% preferred the 1024-bit Oakley Group 2. For IKEv2, 5.8% of profiled servers chose Oakley Group 1, and 63.9% chose Oakley Group 2. This coincides with our anecdotal findings that most VPN clients only offer Oakley Group 2 by default.

SSH All SSH handshakes complete either a finite field Diffie-Hellman or elliptic curve Diffie-Hellman exchange as part of the SSH key exchange. The SSH protocol explicitly defines support for Oakley Group 2 (1024-bit) and Oakley Group 14 (2048-bit) but also allows a server-defined group, which can be negotiated through an auxiliary Diffie-Hellman Group Exchange (DH-GEX) handshake [126].

In order to measure how SSH uses DH in practice, we implemented the SSH protocol in the ZMap toolchain and scanned 1% random samples of the public IPv4 address space in April 2015. We find that 98.9% of SSH servers support the 1024-bit Oakley Group 2, 77.6% support the 2048-bit Oakley Group 14, and 68.7% support DH-GEX.

During the SSH handshake, the client and server select the client's highest priority mutually supported key exchange algorithm. Therefore, we cannot directly measure what algorithm servers will prefer in practice. In order to estimate this, we performed a scan in which we mimicked the algorithms offered by OpenSSH 6.6.1p1, the latest version of OpenSSH. In this scan, 21.8% of servers preferred the 1024-bit Oakley Group 2, and 37.4% preferred a server-defined group. 10% of the server-defined groups were 1024-bit, but, of those, near all provided Oakley Group 2 rather than a custom group.

Combining these equivalent choices, we find that a state-level attacker who performed NFS precomputations for the 1024-bit Oakley Group 2 (which has been in standards for almost two decades) could passively eavesdrop on connections to 3.6M (25.7%) publicly accessible SSH servers.

HTTPS DHE is commonly deployed on web servers. 68.3% of Alexa Top 1M sites support DHE, as do 23.9% of sites with browser-trusted certificates. Of the Top 1M sites that support DHE, 84% use a 1024-bit or smaller group, with 94% of these using one of five groups.

Despite widespread support for DHE, a passive eavesdropper can only decrypt connections that organically agree to use Diffie-Hellman. We can estimate the number of sites

for which this will occur by offering the same sets of ciphersuites as Chrome, Firefox, and Safari. While the offered ciphers differ slightly between browsers, this turns out to result in negligible differences in whether DHE is chosen.

Approximately 24.0% of browser connections with HTTPS-enabled Top 1M sites (and 10% with browser-trusted sites) will negotiate DHE with one of the ten most popular 1024-bit primes; 17.9% of connections with Top 1M sites could be passively eavesdropped given the precomputation for a single 1024-bit prime. The most popular site that negotiates a DHE ciphersuite using one of the two most common 1024-bit primes is sohu.com (ranked 31st globally).

Mail TLS is also used to secure email transport. SMTP, the protocol used to relay messages between mail servers, allows a connection to be upgraded to TLS by issuing the `STARTTLS` command. POP3S and IMAPS, used by end users to fetch received mail, wrap the entire connection in TLS.

We studied 1% samples of the public IPv4 address space for IMAPS, POP3S, and SMTP+StartTLS. We found that 50.7% of SMTP servers supported `STARTTLS`, 41.4% supported DHE, and 14.8% supported DHE_EXPORT ciphers. 15.5% of SMTP servers used one of the ten most common 1024-bit groups.

For IMAPS, 8.4% of servers supported DHE_EXPORT and 75% supported DHE. However, the ten most common 1024-bit primes account for only 5.4% of servers. POP3S deployment is similar, with 8.9% of servers supporting DHE_EXPORT and 74.9% supporting DHE, but with the ten most common 1024-bit primes accounting for only 4.8% of servers.

If each of the top ten 1024-bit primes used by each protocol were compromised, this would affect approximately 1.7M SMTP, 276K IMAPS, and 245K POP3S servers. Using our downgrade attack of Section 4.2.3, an attacker with modest resources can hijack connections to approximately 1.6M SMTP, 429K IMAPS, and 454K POP3S servers.

4.4 Recommendations

Our findings indicate that one of the key recommendations from security experts in response to the threat of mass surveillance — promotion of DHE-based TLS ciphersuites offering "perfect forward secrecy" over RSA-based ciphersuites — may have actually reduced security for many hosts. In this section, we present concrete recommendations to recover the expected security of Diffie-Hellman as it is used in mainstream Internet protocols.

Transition to elliptic curves. Transitioning to elliptic curve Diffie-Hellman (ECDH) key exchanges with appropriate parameters avoids all known feasible cryptanalytic attacks. Current elliptic curve discrete log algorithms for strong curves do not gain as much of an advantage from precomputation. In addition, ECDH keys are shorter than in "mod p" Diffie-Hellman, and shared-secret computations are faster. Unfortunately, the most widely supported ECDH parameters (those specified by NIST) are now viewed with suspicion due to NSA influence on their design despite no known or suspected weaknesses. These curves are undergoing scrutiny, and new curves, such as Curve25519, are being standardized by the IRTF for use in Internet protocols. We recommend transitioning to elliptic curves where possible as the most effective long-term solution to the vulnerabilities we describe.

Increase minimum key strengths. Server operators should disable DHE_EXPORT and configure DHE ciphersuites to use primes of 2048 bits or larger. Browsers and clients should raise the minimum accepted size for Diffie-Hellman groups to at least 1024 bits in order to avoid downgrade attacks when communicating with servers that still use smaller groups. Primes of less than 1024 bits should not be considered secure, even against an attacker with moderate resources.

Our analysis suggests that 1024-bit discrete log may be within reach for state-level actors. As such, 1024-bit DHE (and 1024-bit RSA) must be phased out in the near term. NIST has recommended such a transition since 2010 [26]. We recommend that clients raise the minimum DHE group size to 2048 bits as soon as server configurations allow. Server operators should move to 2048-bit or larger groups to facilitate this transition. Precomputation for a 2048-bit non-trapdoored group is around 10^9 times harder than for a 1024-bit group, so 2048-bit Diffie-Hellman will remain secure barring a major algorithmic improvement.

Avoid fixed-prime 1024-bit groups. For implementations that must continue to use or support 1024-bit groups for compatibility reasons, generating fresh groups may help mitigate some of the damage caused by NFS-style precomputation for very common fixed groups. However, we note that it is possible to create trapdoored primes [162, 363] that are computationally difficult to detect. At minimum, clients should check that servers' parameters use safe primes or a verifiable generation process, such as that proposed in FIPS 186 [252]. Ideally, the process for generating and validating parameters in TLS should be standardized so as to thwart the risk of trapdoors.

Don't deliberately weaken crypto. Our downgrade attack on export-grade 512-bit Diffie-Hellman groups in TLS illustrates the fragility of cryptographic "front doors". Although the key sizes originally used in DHE_EXPORT were intended to be tractable only to NSA, two decades of algorithmic and computational improvements have significantly lowered the bar to attacks on such key sizes. Despite the eventual relaxation of crypto export restrictions and subsequent attempts to remove support for DHE_EXPORT, the technical debt induced by the additional complexity has left implementations vulnerable for decades. Like FREAK [33], our attacks warn of the long-term debilitating effects of deliberately weakening cryptography.

4.5 Disclosure and Response

We notified major client and server developers about the vulnerabilities discussed in the original version before we made our findings public. Prior to our work, Internet Explorer, Chrome, Firefox, and Opera all accepted 512-bit primes, whereas Safari allowed groups as small as 16 bits. As a result of our disclosures, Internet Explorer [240], Firefox, and Chrome are transitioning the minimum size of the DHE groups they accept to 1024 bits, and OpenSSL and Safari are expected to follow suit. On the server side, we notified Apache, Oracle, IBM, Cisco, and various hosting providers. Akamai has removed all support for export ciphersuites. Many TLS developers plan to support a new extension that allows clients and servers to negotiate a few well-known groups of 2048-bits and higher and to gracefully reject weak ones [154].

4.6 Conclusion

Diffie-Hellman key exchange is a cornerstone of applied cryptography, but we find that, as used in practice, it is often less secure than widely believed. The problems stem from the fact that the number field sieve for discrete log allows an attacker to perform a single precomputation that depends only on the group, after which computing individual logs in that group has a far lower cost. Although this fact is well known to cryptographers, it apparently has not been widely understood by system builders. Likewise, many cryptographers did not appreciate that the security of a large fraction of Internet communication depends on Diffie-Hellman key exchanges that use a few small, widely shared groups.

A key lesson from this state of affairs is that cryptographers and creators of practical systems need to work together more effectively. System builders should take responsibility for being aware of applicable cryptanalytic attacks. Cryptographers, for their part, should

involve themselves in how crypto is actually being applied, such as through engagement with standards efforts and software review. Bridging the perilous gap that separates these communities will be essential for keeping future systems secure.

2017 Update When the ACM CCS version of this research was published, the latest discrete log record was a 596-bit computation. Based on that work, and on prior experience with the 768-bit factorization record in 2009 [204], we made the conservative prediction that it was possible, as explained in Section 4.1, to put more computational effort into sieving for the discrete log case than for factoring, so that the linear algebra step would run on a slightly smaller matrix. This led to a runtime estimate of around 35,000 core-years, most of which was spent on linear algebra.

This estimate turned out be overly conservative, for several reasons. First, there have been significant improvements in our software implementation (Section 4.2.3). In addition, our estimate did not use the Joux-Lercier alternative polynomial selection method [191, Section 2.1], which is specific to discrete logarithms. For 768-bit discrete logarithms, this polynomial selection method leads to a significantly smaller computational cost.

In 2016, Kleinjung *et al.* completed a 768-bit discrete logarithm computation [205]. While this is a massive computation on the academic scale, a it has likely been within reach of NS-Attackers for more than a decade. Kleinjung *et al.*'s findings are shown in Table 4.2.

Nation-State Attacker Model When we view this research in the context of our NS-Attacker model, we see many fitting points between the vulnerability attributes and our model. The most obvious of these is that NS-Attackers' advantage of Non-Symmetric Defeats. The index calculus algorithm's property of only needing the group parameters for precomputation and not an instance key-share allows this asymmetry to be exploited. Once the computation is complete and the attacker possess the log database, the per-instance cost of computing the discrete log of a key-share is significantly smaller than the standard discrete log attacks.

NS-Attackers' characteristic of Money provides the ability to conduct the precomputation in a moderately feasible time-frame. Currently, the best estimate for 1024-bit Diffie-Hellman is on the order of tens of millions of dollars. While significantly less than our original estimate of hundreds of millions of dollars, it is still out of reach for other types of attackers. For the largest NS-Attackers, this a more than acceptable financial cost for the amount of information it would make available. The characteristic of Money is also not limited to those that *have* money, but also extends to those whose allies have money. The simplest way to leverage this is an API where allies send IKE transcripts and receive the decryption key

(effectively adding new rules to the "CES Firewall" in Figure 4.5). Another option would be for the more financially capable NS-Attacker to complete the precomputation and then transfer the log table used for per-connection descent to less financially capable allies.

NS-Attackers' advantage of Distant Return Horizons also fits well to this type of attack. As described in Section 2.3.4, NS-Attackers are able to obtain useful information from data acquired a decade or more before. While non-forward secret connection would likely be decryptable long before, the use of Diffie-Hellman would have nullified the usefulness of possible special-purpose devices described in Section 2.3.3. The index-calculus attack against Diffie-Hellman would be applicable in defeating the protections of that historical data as well as current data.

Lastly, NS-Attackers' attribute of Specialization is applicable in multiple ways. First, NS-Attackers employ mathematicians who specialize in cryptography [140, 394, 418] as well as track parameter/algorithm usage (Section 2.4.1). This gives credence to the idea that our discovery was not the first. Second, NS-Attackers are known to work closely with foundries [224] and super-computer companies [263, 394] for intelligence purposes which makes the development process of the ASIC implementations significantly more attainable.

CHAPTER 5

Identifying and Analyzing Vulnerabilities

While the ability to plausibly explain and add technical depth to known NS-Attacker operations, it still relies on an oracle to a vulnerability's existence and connecting the dots or reverse-engineering to find it. A more useful technique would be to find vulnerabilities without such an oracle. In this chapter, we present our 2016 research into common TLS configurations which use "cryptographic shortcuts" that allow an asymmetric defeat well suited for use by NS-Attackers. In the worst-case scenario, we find that the use of TLS Session Tickets combined with popular default configurations and the ubiquity of shared-hosting services allow attackers to retrospectively decrypt a large number of passively acquired TLS connections, regardless of perfect forward secrect cipher usage, after obtaining a server-side secret of order 16-bytes.

TLS is designed with support for perfect forward secrecy (PFS) in order to provide resistance against *future* compromises of endpoints [78]. A TLS connection that uses a *non*-PFS cipher suite can be recorded and later decrypted if the attacker eventually gains access to the server's long-term private key. In contrast, a forward-secret cipher suite prevents this by conducting an ephemeral finite field Diffie-Hellman (DHE) or ephemeral elliptic curve Diffie-Hellman (ECDHE) key exchange. These key exchange methods use the server's long-term private key only for authentication; any attack after the TLS connection has ended will not help the attacker recover the session key. For this reason, the security community strongly recommends configuring TLS servers to use forward-secret ciphers [173, 344]. PFS deployment has increased substantially in the wake of the OpenSSL Heartbleed vulnerability — which potentially exposed the private keys for 24–55% of popular websites [85] — and of Edward Snowden's disclosure of NS-Attacker operations to monitor Internet communications en masse [147, 226].

Despite the recognized importance of forward secrecy, many TLS implementations that use it also take various cryptographic shortcuts that weaken its intended benefits in exchange for better performance. Ephemeral value reuse, session ID resumption [11], and session ticket resumption [92] are all commonly deployed performance enhancements that work by

maintaining secret cryptographic state for periods longer than the lifetime of a connection. While these mechanisms reduce computational overhead for the server and latency for clients, they also create important caveats to the security of forward-secret ciphers.

TLS performance enhancements' reduction of forward secrecy guarantees has been pointed out before [209, 399], but their real-world security impact has never been systematically measured. To address this, we conducted a nine-week study of the Alexa Top Million domains. We report on the prevalence of each performance enhancement and attempt to characterize each domain's *vulnerability window*—the length of time surrounding a forward-secret connection during which an adversary can trivially decrypt the content if they obtain the server's secret cryptographic state. Alarmingly, we find that this window is over 24 hours for 38% of Top Million domains and over 30 days for 10%, including prominent Internet companies such as Yahoo, Netflix, and Yandex.

In addition to these protocol-level shortcuts, many providers employ SSL terminators for load balancing or other operational reasons [236]. SSL terminators perform cryptographic operations on behalf of a destination server, translating clients' HTTPS connections into unencrypted HTTP requests to an internal server. We find that many SSL terminators share cryptographic state between multiple domains. Sibling domains' ability to affect the security of each other's connections also adds caveats to forward secrecy. We observed widespread state sharing across thousands of groups of domains, including tens of thousands of sites that use CloudFlare and thousands operated by Google.

The widespread use of TLS performance enhancements may make them an attractive target for nation-state adversaries. Our findings show that a relatively small attack against an SSL terminator (to recover cryptographic state) could be leveraged to trivially decrypt up to months worth of connections to many different web sites. The cryptographic state could conceivably also be obtained by legal compulsion, such as a warrant or subpoena.

To our knowledge, we are the first to quantify this attack surface and its dangers, and the first to show that real-world TLS security benefits far less from forward secrecy than statistics about support for PFS ciphers would suggest.

5.1 Background

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that operate below the application layer and provide end-to-end encrypted channels for diverse applications, including HTTPS, IMAPS, and SMTP. This section explains how TLS provides forward secrecy and facilitates session resumption. We refer readers to RFC 5280 [38] for a detailed description of the protocol.

5.1.1 Forward Secrecy in TLS

In TLS, perfect forward security [78] protects the confidentiality of connections in the event that the server is *later* compromised by an attacker. Its threat model is an adversary who passively observes and records the TLS handshakes and encrypted traffic between a victim client and server. At some point after the connection has ended, the attacker gains access to the server's secret internal state — perhaps by exploiting a memory leak like Heartbleed [61], by seizing the hardware and performing live-memory forensics, or by computing the server's private key by factoring its public RSA modulus [170]. If the server correctly provides forward secrecy, the attacker will not be able to decrypt connections recorded in the past.

In order to achieve forward secrecy, TLS supports using Diffie-Hellman key exchange to negotiate temporary symmetric keys for the session. The protocol supports two main flavors of Diffie-Hellman: finite-field ephemeral Diffie-Hellman (DHE) and elliptic curve ephemeral Diffie-Hellman (ECDHE). In DHE handshakes, the server selects a finite cyclic group G and a generator g. It picks a random value a and sends $g^a \bmod G$ to the client, while the client picks a random b and sends $g^b \bmod G$ to the server. Both sides then compute g^{ab} and use it to derive the session keys. Per RFC 5246 [342], both the client and server should generate a fresh a and b for each handshake. ECDHE functions similarly but over an elliptic curve group. The client generates a random d_A and sends $d_A G$ to the client, while the client generates d_B and sends $d_B G$ to the server. Both then derive session keys from $d_A d_B G$.

Whether the handshake uses DHE or ECDHE, the server still needs to *authenticate* itself to the client in order to prevent man-in-the-middle attacks, and it does so using its long-term private key and certificate. However, a successful attack on the authentication would require compromising the private key before the TLS handshake completes. After that, as long as the client and server both discard the session state, the connection data should be infeasible to decrypt.

Using forward secret TLS handshakes is considered a security best practice [344], and all modern browsers support them. However, many server implementations, including Apache and Nginx, must be manually configured to use them.

5.1.2 Session Resumption

In order to reduce connection overhead, TLS allows subsequent sessions to resume a prior session without completing a full handshake. The protocol provides two mutually exclusive mechanisms to do this: session ID resumption and session tickets. Both mechanisms allow the server to skip a costly public-key operation on later connections, and they save one network round trip of latency. As we will show, server support for these resumption methods

is pervasive — 50% of Mozilla Firefox TLS sessions are resumptions[1] — and of the Alexa Top Million websites that support HTTPS, 83% support session ID resumption and 76% support session tickets.

Session ID Resumption Session ID resumption was introduced in SSL 2.0 [172] and allows a client and server to quickly resume an existing session. During the initial handshake, the server provides a random session ID, which both the client and server maintain in a table that maps IDs to session keys and connection states from recent connections. Upon reconnection, the client provides this session ID in its first protocol message, Client Hello. If the server recognizes the session, it will respond with a Server Hello message containing the same session ID, after which both sides immediately resume an encrypted connection using the original session keys. RFC 5246 suggests a maximum 24-hour session lifetime, after which the server should discard the cached key and state.

Session Ticket Resumption TLS session tickets were introduced in RFC 4507 [352] and redefined in RFC 5077 [92]. They allow session resumption without requiring the server to maintain per-connection state. Instead, the server provides the client with an opaque encrypted "ticket" containing the session keys and other data necessary to resume the session. The client includes this ticket in later connections as an offer to resume without the full handshake. More precisely, when the client first connects, it includes an empty session ticket extension in its Client Hello. The server includes a corresponding extension in the Server Hello message and, after the key exchange completes, sends the client an opaque ticket and a lifetime "hint" in a New Session Ticket message. The client then stores a mapping of the server's identity to the session ticket and cryptographic state required for the client to resume the connection. On subsequent connections, the client includes the ticket in its Client Hello. If the server accepts the ticket, the pair completes an abbreviated handshake, like in session ID resumption. During this process, the server can reissue the client a fresh session ticket, but the cipher and session keys remain constant.

The ticket can contain arbitrary data, but RFC 5077 recommends a structure consisting of a randomly generated key name (identifying the symmetric keys used to encrypt the ticket), an IV, the encrypted state, and a MAC. The RFC recommends that the server encrypt the state using AES-CBC and a 128-bit key and construct the MAC using HMAC-SHA-256 with a 256-bit key. (Note that these keys are never revealed to the client, which merely stores the encrypted ticket and returns it in later connections.) Throughout this work, we refer to the symmetric encryption key as the "Session Ticket Encryption Key" (STEK). Common

[1] As seen by Mozilla Firefox Telemetry [244] from March 3 to March 15, 2016.

server implementations, including Nginx and Apache, support both loading pregenerated STEKs from the filesystem and generating random STEKs upon server initialization.

Impacts on Forward Secrecy Both of these performance enhancements degrade the protection achieved by forward-secret TLS handshakes [209, 399]. The client and server will store the same symmetric key for use in future sessions, extending the lifetime of the ephemeral handshake. More importantly, for session tickets, compromising the server's STEK would allow decryption of all prior connections for which that STEK was used. If a server's STEK never changes, the site provides no effective forward secrecy to connections that use TLS session tickets, regardless of the key exchange mechanism used.

5.1.3 Reusing Ephemeral Values

While not a session resumption technique, servers will oftentimes reuse DHE and ECDHE values to reduce computation for each initial handshake. For instance, with DHE, a server might repeatedly use the same value a so that it does not have to keep computing g^a. As we will discuss later, we empirically find that at least 7.2% of HTTPS domains in the Alexa Top Million reuse DHE values and 15.5% reuse ECDHE values.

Since the client will generate its own unique values (b, g^b), the session keys derived from g^{ab} will differ for every connection. However, an attacker who obtains the server's a can compute the session keys for any observed connection that uses it. Thus, forward secrecy is not actually achieved until the server stops reusing this value and securely erases it. If the server's a never changes, then a PFS key exchange does not provide any effective forward secrecy.

We discuss how session resumption and ephemeral value reuse affect the TLS ecosystem's attack surface — and attacker incentives — in Section 5.5.

5.1.4 Changes in TLS 1.3

Although still in the draft stage, TLS 1.3 [341] makes many changes to session resumption and other security properties. Session IDs and session tickets are nominally obsoleted, but the mechanisms persist via the pre-shared keys (PSKs).

A PSK identifier is issued by the server in a New Session Ticket message after the first handshake is complete and then included in the second connection's Client Hello. The identifier itself may contain a database lookup key (analogous to a session ID resumption) or an encrypted and authenticated copy of the TLS resumption state (analogous to a session

DHE	Alexa 1M domains (14Apr2016)	957,116
	Non-blacklisted domains	952,991
	Browser-trusted TLS domains	427,313
	Support DHE ciphers	252,340
	$\geq$ 2x same server KEX value	18,113
	All same server KEX value	12,461
ECDHE	Alexa 1M domains (15Apr2016)	958,470
	Non-blacklisted domains	954,338
	Browser-trusted TLS domains	438,383
	Support ECDHE ciphers	390,120
	$\geq$ 2x same server KEX value	60,370
	All same server KEX value	41,683
Session Tickets	Alexa 1M domains (17Apr2016)	956,094
	Non-blacklisted domains	951,978
	Browser-trusted TLS domains	435,150
	Issue session tickets	354,697
	$\geq$ 2x same STEK ID	353,124
	All same STEK ID	334,404

Table 5.1: **Support for Forward Secrecy and Resumption**

ticket resumption). Unlike the current TLS versions, version 1.3 explicitly derives a separate resumption secret.

This resumption secret can be used in two ways for session resumption. The first is for a direct resumption for a secondary session via the "`psk_ke`" mechanism. The second is to be used as authentication for resumed connection that conducts a second (EC)DHE key exchange via the "`psk_dhe_ke`" mechanism.

In addition to these two, the resumption secret can also be used for QUIC-like [339] 0-RTT communication. In this case, "early data" is sent by the client while awaiting the completion of a resumed or new TLS handshake. The data is encrypted to the resumption secret and can stream until the client receives the server's Finished message.

5.2 Data Collection

To assess the impacts of session resumption and ephemeral value reuse, we measured HTTPS behavior of Alexa Top Million domains [9] over a 9-week period in the Spring of 2016. We repeatedly connected to each server on TCP/443 using a version of the ZMap tool chain [82, 86] that we modified to support session ID and ticket resumption. In all cases, we restricted our analysis to websites that presented browser-trusted certificates that chain to

the NSS root store[2]. Table 5.1 gives high level metrics from conducting 10 TLS connections in quick succession to each Alexa Top Million domain on the days given.

As with any active scanning research, there are many ethical considerations at play. We followed the best practices defined by Durumeric et al. [86] and refer to their work for more detailed discussion of the ethics of active scanning. All scans were completed from the University of Michigan campus and followed the institutional blacklist. For experiments that required multiple connections in a single day, we restricted our measurements to popular sites in the Alexa Top Million for which this load should be negligible. When possible we used existing data from the Censys Project [82] instead of running redundant scans. We are publishing all of the data we independently collected on Scans.io [83], and our modifications have been merged into the main ZMap project.

Alexa Top Million Dataset Our measurements occurred within a 9-week period from March 2, 2016 to May 4, 2016 and used the Alexa Top Million as the target domains. We saw a surprising amount of churn within the Top Million domains from day to day. In total, we scanned 1,527,644 unique domains including over 155K which were in ≤ 7 polls of the Top Million. Only 539,546 domains remained in the Top Million for the whole 9 weeks. Of these, 369,034 (68%) ever supported HTTPS, 291,643 (54%) ever presented a browser-trusted certificate, and 288,252 (53%) ever issued a session ticket, completed a DHE or ECDHE key exchange, or resumed a session. To prevent churn in the Top Million from biasing our results, we restrict measurements over multiple days to domains that remained in the list for the entire period.

5.3 TLS Secret State Longevity

In this section, we describe HTTPS domains' behavior in practice with regard to the lifetime of cryptographic state, including how long session ID and session ticket resumption is allowed, the lifetime of session ticket encryption keys, and the reuse of key exchange values. We find that while session IDs and session tickets are generally only honored for under an hour (82% and 76%, respectively), session ticket encryption keys (STEKs) persist much longer.

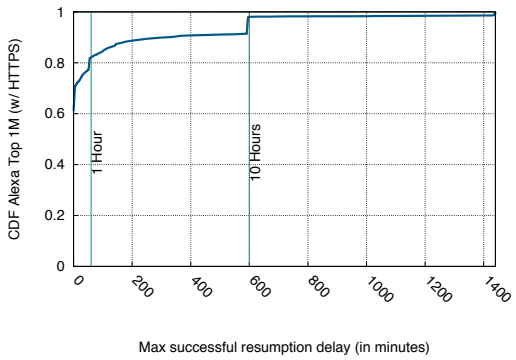

Figure 5.1: **Session ID Lifetime** — We measured how long Session IDs were honored by HTTPS websites in the Alexa Top Million.

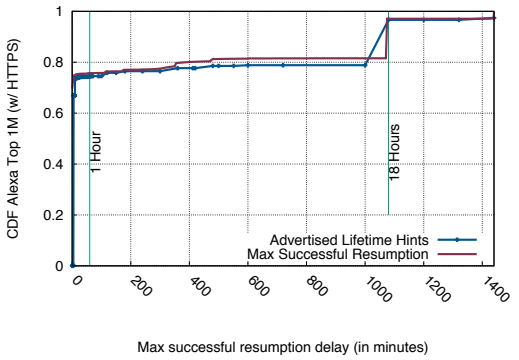

Figure 5.2: **Session Ticket Lifetime** — We measured advertised session ticket lifetime and how long tickets were honored by Alexa Top Million websites.

5.3.1 Session ID Lifetime

To measure how long session IDs are accepted, we initiated a TLS handshake with each of the Alexa Top Million domains on April 27, 2016. We attempted to resume each session one second later and then every five minutes until either the site failed to resume the session or 24 hours had elapsed. Of the 433,220 domains that supported HTTPS and presented a browser trusted certificate, 419,302 (97%) indicated support for session ID resumption by setting a session ID value in the Server Hello message, and 357,536 (83%) resumed the session after a one second delay.

As shown in Figure 5.1, the distribution of lifetimes is somewhat discrete: 82% of domains that supported session ID resumption allowed resumption for one hour or less, and 61% did for less than five minutes. Only 2,845 domains (0.8%) resumed sessions for 24 hours or longer; 86% of those domains belong to or are hosted by Google. We also note that Facebook's CDN honored session IDs for more than 24 hours.

These empirical results align with the default configuration of population web server implementations. Apache enables session ID resumption by default and sets the lifetime to five minutes [242]. Nginx issues session IDs but does not allow resumption unless it is explicitly configured; session IDs expire after five minutes when enabled unless the administrator sets a different lifetime [243]. Microsoft IIS expires session IDs after ten hours [239], corresponding to the jump seen in Figure 5.1.

[2]Durumeric et al. find that 99.5% of certificates trusted by NSS are valid in all major browsers and can be used to estimate browser trusted websites [84].

5.3.2 Session Ticket Lifetime

We use a similar experiment to measure how long domains allowed session tickets to resume TLS connections and the hinted lifetime. We initiated a TLS handshake with each site in the Alexa Top Million on April 29, 2016. We attempted to resume each connection one second later, then every five minutes until either the domain failed to resume the session or 24 hours had elapsed. If the domain reissued a session ticket during any of the connections, we continued to attempt resumption with the ticket issued from the first connection. We found that 366,178 out of the 461,475 domains with a browser-trusted certificate (79%) issued a session ticket and 351,603 (76%) resumed the session after one second.

Similar to session ID resumption, 67% of domains accepted a session ticket for less than five minutes and 76% for one hour or less as seen in Figure 5.2. The indicated ticket lifetime closely follows the advertised lifetime hint, with the exception of 14,663 domains that leave it unspecified and up to the client's policy [92]. At the extreme end, we found that two domains specified a lifetime hint longer than ten days: `fantabobworld.com` and `fantabobshow.com`, both of which specified a 90 day hint. 54,522 unique domains hosted by CloudFlare resumed for 18 hours, causing the steep increase in Figure 5.2. As with session ID resumption, 8,969 domains accepted tickets for 24 hours, of which 8,535 were hosted by Google (95%), which specified a 28 hour lifetime hint.

This behavior also agrees with the known defaults for popular web server implementations. Apache and Nginx both enable session ticket resumption by default with a three minute lifetime.

5.3.3 STEK Lifetime

While the time span that domains will accept previously issued session tickets is an important metric, it reflects only the ticket's lifetime (set by policy) and not necessarily the time period for which the associated STEK exists and is used to issue new session tickets. As discussed in Section 5.1, the content of a historical session can be decrypted using a site's STEK regardless of whether a PFS handshake occurs and regardless of whether the ticket's lifetime has expired or not. In other words, a "forward secret" session is not actually forward secret while the STEK that encrypted the associated ticket persists.

While it is not possible to directly detect that the key used to encrypt the session state has changed, popular server implementations include a 16-byte STEK identifier in the ticket, as prescribed in RFC 5077 [92]. We reviewed popular open-source TLS implementations, including OpenSSL, LibreSSL, GNUTLS, mbedTLS, and NSS, and found that all follow this recommendation except for mbedTLS, which uses a 4-byte STEK identifier. We also

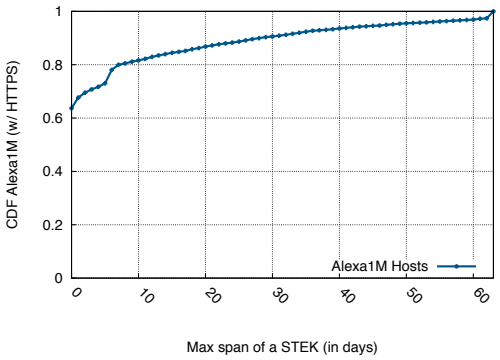

Figure 5.3: **STEK Lifetime** — TLS connections cannot achieve forward secrecy until the STEK (the key used by the server to encrypt the session ticket) is discarded.

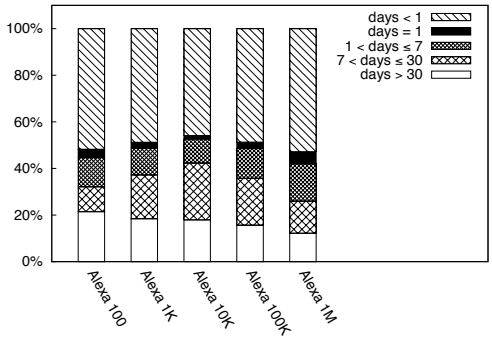

Figure 5.4: **STEK Lifetime by Alexa Rank** — We found 12 Alexa Top 100 sites that persisted STEKs for at least 30 days.

tested Microsoft's SChannel implementation and found it to use an ASN.1 encoded object containing a DPAPI object [42]. For the measurements below, we parsed this object and extracted the Master Key GUID to use as the STEK identifier.

Between March 2, 2016 and May 4, 2016, we connected to the Alexa Top Million domains daily and recorded the session ticket that was issued by the server, if one was issued. We were able to determine the lifetime of each STEK by looking for the first and last time that the (STEK identifier, domain) pair was seen. As opposed to measuring the number of sequential days that a domain issues tickets with identical STEK identifiers, this metric accounts for much of the real-world jitter seen in Internet scanning. This could be due to the ZMap tool-chain's choice of A-record entries between days, a poorly configured load balancer which does not maintain client-server affinity, or simply the server failing to respond to one our connections. It is highly unlikely that an administrator would switch static STEKs only to switch back or that a randomly generated STEK identifier would collide within the bounds of our study. Therefore, we can safely assume that a STEK was in use between the first and last time that its identifier was seen and that any intermediate STEK identifiers seen were the result of fluctuations connecting to different servers.

Of the 291,643 browser-trusted sites always in the Alexa Top Million, 66,941 (23%) never issued a session ticket. 118,835 (41%) used different issuing STEKs for each day. 63,976 domains (22%) reused the same STEK for at least 7 days, and 28,210 domains (10%) reused for at least 30 days. We show the CDF of these lifetimes in Figure 5.3.

We found a surprising collection of websites, including those of major Internet companies, that fall within the 30+ day reuse. Table 5.2 shows the ten most popular domains according to their average Alexa ranking that reused a STEK for at least 7 days. While there

Rank	Domain	# Days	Rank	Domain	# Days
5	yahoo.com	63	31	netflix.com	54
19	qq.com	56	35	imgur.com	63
20	taobao.com	63	41	tmall.com	63
21	pinterest.com	63	53	fc2.com	18
28	yandex.ru	63	55	pornhub.com	29

Table 5.2: **Top Domains with Prolonged STEK Reuse**—We show the most popular domains (by average Alexa rank) that reused a STEK for at least 7 days.

are many other notable domains, we note that there are a total of eight `yandex.[tld]` domains, each of which showed 63 days of reuse, `slack.com` (a popular team communication service) showed 18 days of reuse, and `mail.ru` showed 63 days of reuse. 63 days indicates that it was seen on both the first and last day of our study and was likely in use both before and after our study.

Figure 5.4 depicts how STEK lifetimes varied with Alexa rank tiers according to the average rank of each domain over the 9-week period. We observed 56 domains which issued session tickets in the Alexa Top 100, 494 in the Top 1K, 4,154 in the Top 10K, 37,224 in the Top 100K, and 224,702 in the Alexa Top Million. Again, these are only domains which remained within the Alexa Top Million for the entire span of our study.

The longevity of STEK lifetimes can be largely explained by the the popular implementations. Apache 2.4.0 and Nginx 1.5.7 and later allow an administrator to configure the server to read 48 bytes of randomness from a file path on disk. This file contains the STEK identifier, encryption key, and MAC key in order to synchronize STEKs across servers. This configuration can only be changed via direct interaction from the administrator and restarting the server process. If this option is not available, or if a key file is not configured, the server randomly generates a STEK on startup and uses it for the lifetime of the process.

While there is a worrying set of websites that appeared to never rotate STEKs, we note that many have more reasonable configurations. Google, Twitter, YouTube, Baidu, and many others never reused an issuing STEK across days. However, as we will discuss in Section 5.6, that is not always the sole indicator of a secure configuration.

5.3.4 EC(DHE) Value Lifetime

As described in Section 5.1, TLS servers can cache and reuse ephemeral handshake values (a, g^a in a finite-field Diffie-Hellman exchange or d_A, $d_A G$ in elliptic curve Diffie-Hellman) to reduce the computational cost of public key cryptography. Table 5.1 shows that 7.2% of domains in a single Alexa Top Million list reuse a DHE value for some amount of time and

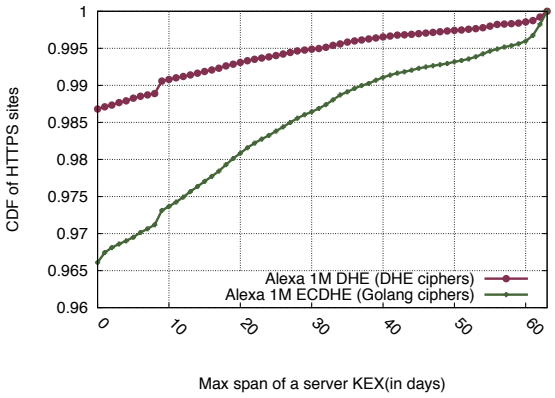

Figure 5.5: **Ephemeral Exchange Value Reuse**—We measured how long Alexa Top Million websites served identical DHE and ECDHE values (note vertical scale is cropped).

15.5% reuse an ECDHE value for amount of time.

To determine how long these ephemeral values persist, we analyzed two sets of daily scans for the Top Million Domains. One set, obtained from the Censys project [82], offered only DHE ciphers and the other offered ECDHE and RSA ciphers, with ECDHE as the first priority.

DHE Of the 291,643 domains consistently in the Alexa Top Million and who support HTTPS with a valid certificate, only 166,608 (57%) ever connected successfully when the client offered only DHE ciphers. 12,824 domains (4.4%) reused a DHE value for some amount of time in the 10 connection scans referenced in Table 5.1. The Censys project scans show that 3,849 (1.3%) reused a DHE value for at least one day, 3,347 (1.2%) for at least 7 days, and 1,527 (0.52%) for 30 or more days. Figure 5.5 shows this visually.

Table 5.3 shows the top ten domains which reused a DHE value for more than 7 days as determined by their average Alexa rank. We also find `commsec.com.au` (an Australian brokerage firm) with 36 days of reuse and 32 `kayak.[tld]` domains with between 6 and 18 days of reuse.

ECDHE 234,302 domains 80% of those consistently in the Alexa Top Million who support HTTPS with a valid certificate, completed an ECDHE handshake. 42,029 domains (14.4%) reused an ECDHE value for some amount of time in our 10 connection scans referenced in Table 5.1. In our daily scans, we saw 9,886 domains (3.4%) that reused an ECDHE value for at least one day, 8,710 (3.0%) reused for at least 7 days, and 4,071 (1.4%) reused for 30 or more days. This is shown visually in Figure 5.5.

Table 5.4 shows the top ten domains that reused an ECDHE value for more than

Rank	Domain	# Days	Rank	Domain	# Days
31	netflix.com	59	580	kayak.com	13
53	fc2.com	18	592	cbssports.com	60
392	ebay.in	7	626	gamefaqs.com	12
456	ebay.it	8	633	overstock.com	17
528	bleacherreport.com	24	730	cookpad.com	63

Table 5.3: **Top Domains with Prolonged DHE Reuse** — We show the most popular domains (by average Alexa rank) that reused a DHE value for at least 7 days.

Rank	Domain	# Days	Rank	Domain	# Days
31	netflix.com	59	353	paytm.com	27
74	whatsapp.com	62	464	playstation.com	11
158	vice.com	26	527	woot.com	62
221	9gag.com	31	528	bleacherreport.com	24
322	liputan6.com	28	615	leagueoflegends.com	27

Table 5.4: **Top Domains with Prolonged ECDHE Reuse** — We show the most popular domains (by average Alexa rank) that reused an ECDHE value for at least 7 days.

7 days. Notable domains beyond the top ten include `betterment.com` (an online investing service) with 62 days of reuse, `mint.com` (a budgeting website that connects to banks and investment services) with 62 days of reuse, and `symantec.com`, `symanteccloud.com`, and `norton.com` with 41, 16, and 19 days of reuse respectively.

As seen in Figure 5.5, the ephemeral value longevity metrics are fairly consistent with one another, but are substantially different from the STEK longevity rates seen in Figure 5.3.

5.4 TLS Secret State Sharing

When measuring the increased attack surface resulting from stored TLS secrets, it is also important to consider cases where secrets are shared across domains, servers, or data centers. If a shared TLS secret is extracted from a single site, it can be used to compromise connections to all the other sites regardless of whether they use different long-term SSL certificates.

We found many "service groups" in which multiple domains shared a session cache, STEK, or Diffie-Hellman value, making these secrets particularly valuable targets for attack. While it would be logical for a single domain to use this technique to allow sessions to be resumed across multiple servers, the magnitude of sharing across domains was surprising. The root cause of this behavior is likely that domains share an SSL terminator, whether it is

Operator	# domains	Operator	# domains
CloudFlare #1	30,163	Blogspot #2	743
CloudFlare #2	15,241	Blogspot #3	732
Automattic #1	2,247	Blogspot #4	648
Automattic #2	1,552	Shopify	593
Blogspot #1	849	Blogspot #5	561

Table 5.5: **Largest Session Cache Service Groups**

a separate device such as a Cavium card [49] or multiple domains running on the same web server.

5.4.1 Shared Session ID Caches

To establish a lower bound on how many websites share session ID caches, we conducted a cross-domain probing experiment where we attempted to resume a TLS connection to domain b with a session that originated from domain a. If performed exhaustively, this would require hundreds of thousands of connections to each domain. However, we made the experiment tractable by limiting groups to a small number of domains from each AS and by transitively growing the graph. That is, if we observed that id_a was valid on domain b and id_b was valid on domain c, we conclude that id_a would have also been valid on domain c and group domains a, b, and c together.

For each site, we randomly selected up to five other sites in its AS and up to five sites that shared its IP address and tested whether its session ID allowed connection to these other sites. We note that because servers can expire session IDs at any time, there is no harm to the server to provide an invalid session ID; the server will simply complete a typical TLS handshake as if no session ID had been presented.

Of the 357,536 domains that supported session ID resumption in Section 5.3.1, we found 212,491 service groups, of which 183,261 (86%) contained only a single domain. The largest service group we found belonged to CloudFlare and contained 30,163 domains (66% of the 45,520 Alexa Top Million domains in their AS). We show the ten largest session cache service groups in Table 5.5.

As shown in the table, we observed cases where a single logical provider (such as a CDN or cloud services company) had multiple service groups even within the same /24 CIDR block. We manually confirmed that this was not an artifact of our grouping methodology and in fact reflected the remote configuration. While we believe that this measurement technique is effective, it provides only a lower bound on the true number of domains that share session caches. Our ability to provide a tighter estimate is limited, since TLS does not provide the

Operator	# domains	Operator	# domains
CloudFlare	62,176	GoDaddy	1,875
Google	8,973	Amazon	1,495
Automattic	4,182	Tumblr #1	975
TMall	3,305	Tumblr #2	959
Shopify	3,247	Tumblr #3	956

Table 5.6: **Largest STEK Service Groups**

Operator	# domains	Operator	# domains
SquareSpace	1,627	Atypon	167
LiveJournal	1,330	Affinity Internet	146
Jimdo #1	179	Line Corp.	114
Jimdo #2	178	Digital Insight	98
Distil Networks	174	EdgeCast CDN	75

Table 5.7: **Largest Diffie-Hellman Service Groups**

client any information about the session cache or saved session state other than the random session ID.

5.4.2 Shared STEKs

To track how STEKs are shared across servers, we connected to each April 17, 2016 Alexa Top Million domain ten times over a six hour window and grouped sites together that shared at least one STEK identifier during the scans. Since some providers rotate session tickets at smaller intervals than six hours, we repeated the experiment with one connection over a 30 minute window, similarly grouped domains, and then joined the two groups.

Of the 354,697 sites that supported session tickets, we found 170,634 STEK service groups, of which 140,715 (83%) contained only a single domain. As with session IDs, the largest group belonged to CloudFlare; it contained 62,176 domains. The next largest belonged to Alphabet (Google's parent company) and contained 8,973 hosts sharing a STEK. We show the top ten largest STEK service groups in Table 5.6.

5.4.3 Shared (EC)DHE Values

Lastly, we looked for Alexa Top Million domains that shared DHE or ECDHE key-exchange values. To do this, we completed 10 TLS handshakes with each Alexa Top Million domain over a five-hour window. As with the shared STEK experiment, we also performed a scan that made a single connection to every domain during a 30 minute window. Both scans were

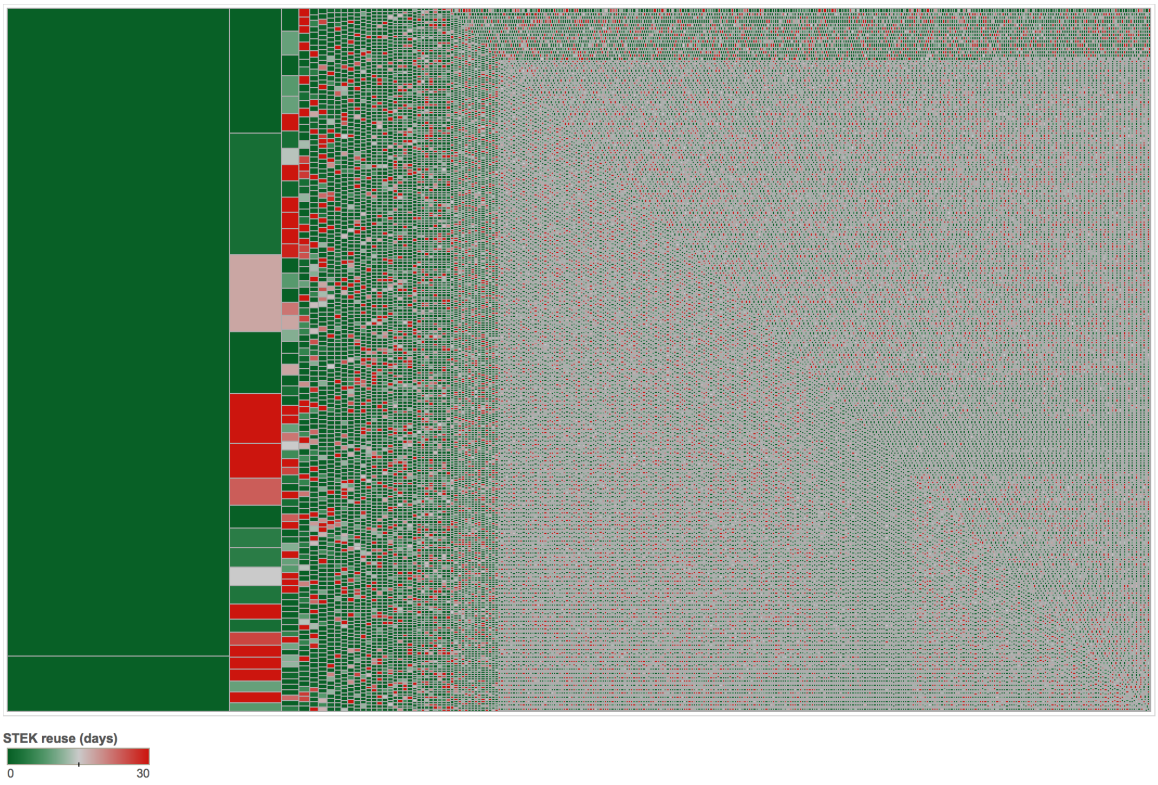

Figure 5.6: **STEK Sharing and Longevity Visualization**—Each box in this illustration is sized proportionally to the number of domains in that service group and colored according to the observed longevity of the key. Solid red boxes represent groups of domains that shared a key that persisted for at least 30 days.

conducted twice, once with only DHE ciphers and once with only ECDHE ciphers, for a total of four scans.

We found that Diffie-Hellman values were shared in fewer instances and by somewhat smaller groups than either session caches or STEKs. The most widely shared DHE value was one we saw 1,368 times across 137 domains and 119 IP addresses, all within AS 20401 (Hostway Corporation). We also found a single ECDHE value shared 1,790 times across 179 domains on a single IP, which appeared to be a Jimdo hosting server [182] on Amazon EC2.

We labeled servers that ever presented the same DHE or ECDHE key-exchange value to be part of the same service group. We found 421,492 Diffie-Hellman service groups, of which 417,397 (99%) contained only a single domain. The largest group belonged to SquareSpace and contained 1,627 domains. We identify the largest ephemeral value service groups in Table 5.7.

5.5 Crypto Shortcut Dangers

As of May 2016, we find that 90.2% of Top Million domains with trusted HTTPS use forward secret key exchanges for connections from modern browsers. Prior to our study, we — the authors — would have assumed from this that connections would be forward secret shortly after the connection has ended. However, when we consider the interaction of crypto shortcuts and cross-domain secret sharing, we see that this is not the case and that many popular domains remain susceptible to retrospective decryption.

As opposed to the naive understanding, forward secrecy is not a binary concept being either forward secret or not forward secret. Forward secrecy is a gradient where the confidentiality of the data is forward secret after some passage of time dependent on many different factors. At one extreme, an arbitrarily complex key-ratcheting mechanism could protect data confidentiality even if an endpoint is compromised while the connection is in progress. The attacker would be able to decrypt the connection's content after the compromise, but not before. At the other extreme, a TLS connection that uses RSA key exchange is effectively never forward secure. Due to the long-term nature of most SSL certificates as well as the likelihood that they are stored on disk, recovery is likely possible even long after the certificate has expired.

To quantify the amount of forward secrecy, we can attempt to establish lower bounds for each site's *vulnerability window*. This is the span of time during which an attacker could recover the session keys for an observed TLS connection by compromising secret values stored by the server. Our measurements from the previous sections allow us to estimate lower bounds for this window, but the true exposure may be much greater. While we can detect that a server refuses to resume older sessions, we cannot tell whether it has securely erased the corresponding secrets or whether the secrets may be vulnerable to forensic recovery.

In addition to quantifying the amount of forward secrecy, we also wish to account for the concentration of the secrets themselves. In a secure world, a compromise of one server would affect as few connections on as few domains as possible. But as shown in Section 5.4, this is far from the case and that the compromise of a small number of SSL endpoints could endanger an out-sized number of domains' content.

The interaction of these two factors presents an enticing target for an attacker who wishes to decrypt large numbers of connections for a comparatively small amount of work.

5.5.1 Exposure from Session Tickets

The long-term usage of session ticket encryption keys (STEKs) is the most worrisome practice we observed. Since the session ticket contains the session keys encrypted with the

STEK, and since it is sent as part of each TLS connection outside of the TLS tunnel (initially by the server and subsequently by the client), an attacker who obtains the associated STEK can decrypt the ticket, recover the session keys, and decrypt the connection contents.

The vulnerability window begins when the STEK is generated (potentially before the victim connection) and ends when it is securely erased from all servers. As reported in Section 5.3.3, 36% of the ticket-issuing domains we considered reused the same STEK for at least a day, 22% for more than a week, and 10% for more than a month.

In Figure 5.6, we visualize the interaction of session ticket service groups and the median STEK reuse for each service group. The two largest service groups (CloudFlare and Google) account for 20% of Top Million HTTPS sites and are shown in the far-left column, and both reused STEKs for less than 24 hours. On the opposite end of the longevity spectrum were TMall (a Chinese online retailer) and Fastly (a CDN), which are represented by the largest red elements in the second column of Figure 5.6. Together, they accounted for 1,208 domains. Fastly, which controlled domains such as `foursquare.com`, `www.gov.uk`, and `aclu.org`, always issued session tickets with the same STEK throughout our 9-week study.

While not one of the largest service groups, we note a concerning cluster of sites controlled by Jack Henry & Associates. This service group contains 79 bank and credit union domains which issued session tickets for 59 days using a single STEK and then all rotated to a different — but still shared — STEK for the final 4 days of our study.

While we are pleased that many of the largest service groups rotate their STEKs at least daily, the magnitude of reliance on a small number of secret values is disconcerting. Current versions of Chrome, Firefox, IE, and Microsoft Edge all offer the session ticket extension by default and an attacker who could collect the traffic as well as obtain the STEK within the vulnerability window would be able to decrypt and access the millions of victims' connection content with ease.

5.5.2 Exposure from Session Caches

When a server supports session ID resumption, an attacker can potentially recover keys for past sessions as long as they reside in the server's session cache. As such, the vulnerability window begins when the victim connection completes its handshake and ends when the server implementation securely discards the session state.

Our experiments in Section 5.3.1 show that at least 83% of Top Million sites employ session caching and retain state for some amount of time after a connection, and at least 18% do so for more than 60 minutes. Section 5.4.1 shows that session cache sharing is

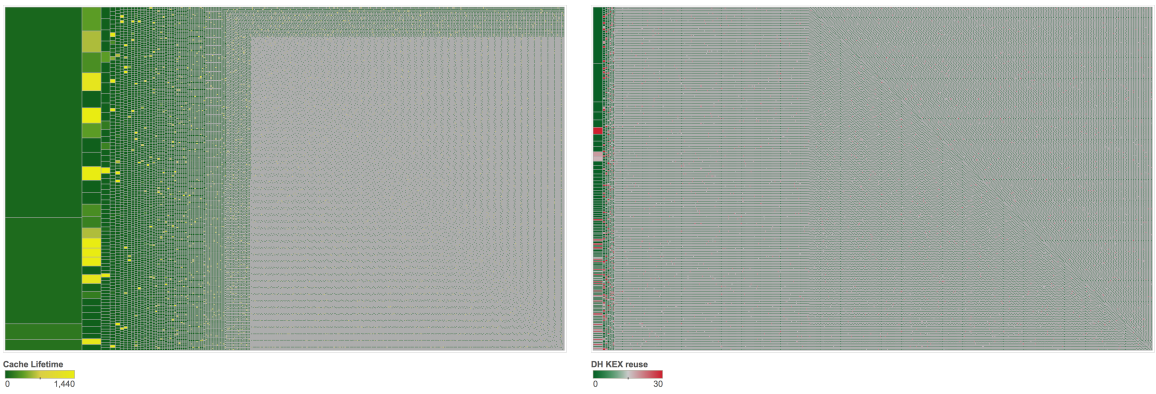

Figure 5.7: **Visualizing Session Caches and Diffie-Hellman Reuse**—For comparison with Fig. 5.6, we show similar illustrations of the longevity and cross-domain sharing exhibited by session caches (*left*) and repeated Diffie-Hellman values (*right*).

widespread, with 49% of Top Million domains sharing a cache with at least one other popular domain. Figure 5.7 shows the interaction of these measurements.

The combined effect of session caching and cache sharing makes large interdomain session caches a particularly attractive target for attackers. The ten largest shared caches (Table 5.5) account for 15% of Top Million domains and exhibited median vulnerability windows of 5 and 1,440 minutes (24 hours). Of these, the five longest-lived all belonged to Google Blogspot and exhibited median cache lifetimes ranging from 4.5 hours to 24 hours (the maximum we tested). An attacker who could access the contents of any one of these caches would be able to decrypt hours' worth of TLS traffic for hundreds of popular sites.

Compared with Figure 5.6, Figure 5.7 shows a similar distribution within the largest service groups. Although the maximum vulnerability windows are orders of magnitude different, the proportional distribution is similar.

5.5.3 Exposure from Diffie-Hellman Reuse

When a server reuses Diffie-Hellman ephemeral values (contrary to the advice of RFC 5246 [342]), this also leads to an extended vulnerability window. The window last from the time the server generates its random Diffie-Hellman value (a or d_A) until that value is securely erased. Like session tickets, an attacker who leaks the server's Diffie-Hellman value can also decrypt *future* TLS connections until the server ceases using that value as well as any previous connections using that value.

Figure 5.7 shows combined effect of longevity and inter-domain sharing was significantly smaller for Diffie-Hellman reuse than for session resumption, but it still resulted in a few notable high-value targets. Affinity Internet shared a single Diffie-Hellman value across

91 domains for 62 days, and Jimdo shared one value for 19 days across 64 domains and another value for 17 days across a different 60 domains (seen as the red blocks in the far left column).

5.5.4 Combined Exposure

Since session tickets, session caches, and Diffie-Hellman reuse all lead to an extended vulnerability window, an attacker with some way of accessing the server's internal state could choose to exploit any of them to compromise forward secrecy. A domain's overall exposure is determined by the longest vulnerability window it exhibits for any of these mechanisms.

Of the 291,643 domains that were in the Alexa Top Million for the duration of our measurements and supported HTTPS with a browser-trusted certificate, 288,252 (99%) issued a session ticket, resumed a session, or conducted a DHE or ECDHE key exchange. Figure 5.8 shows the distribution of the maximum vulnerability window found for every domain.

About 90% of browser-trusted Top Million domains with browser-trusted certificates are configured to support forward-secrecy with modern browsers, which, as commonly thought of, would result in a vulnerability window that lasts no longer than the connection. Due to combined effects of the TLS crypto shortcuts we have discussed, we find that 110,788 domains (38%) have a maximum vulnerability window of more than 24 hours, 65,028 (22%) of more than 7 days, and 28,880 (10%) of more than 30 days.

5.6 Nation-state Perspective

As seen above, our results indicate that TLS crypto shortcuts leave popular HTTPS sites significantly less well protected than we thought in the face of server-side information leaks such as Heartbleed. However, the risks of these mechanisms appear even more severe if we consider threats from nation-state attackers such as the NSA. In particular, the "shape" of the vulnerability windows created by session tickets is ideally suited for exploitation by intelligence agencies for surveillance purposes. In this section, we consider how a nation-state attacker might seek to exploit TLS crypto shortcuts and we assess the potential impact on Internet security of such a compromise against one particular high-value target, Google. Due to the availability of information regarding the NSA and other "Five Eyes" agencies, we focus on the *modi operandi* of these groups.

Recent TLS vulnerabilities — such as FREAK [34], Logjam [8], and DROWN [20] —

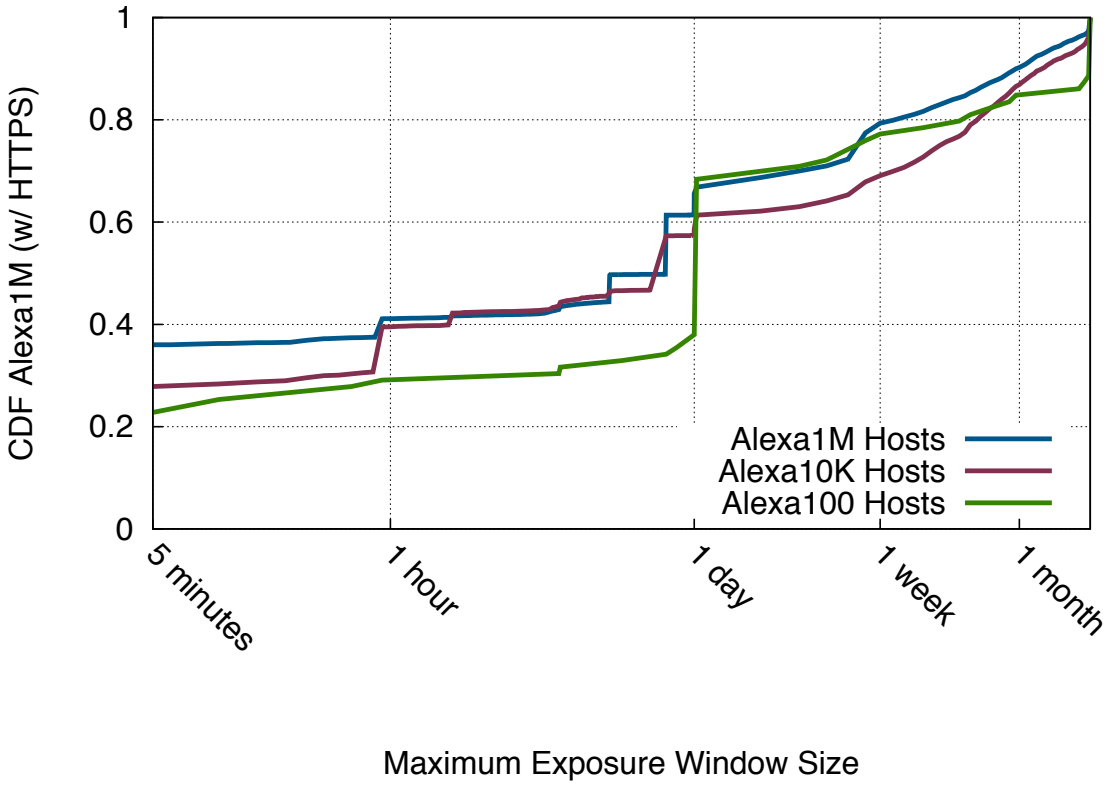

Figure 5.8: **Overall Vulnerability Windows**—This CDF depicts the combined effects of exposure from session tickets, session caches, and Diffie-Hellman reuse.

require active interference with each connection, making them unsuitable for stealthy, retroactive, or wide-scale surveillance. Some researchers believe that NSA can currently defeat TLS encryption when used with 1024-bit RSA [222] or DHE [8]. In either case, specific non-standard configurations would be required in cipher selection (preferring RSA client write and DHE ciphers with specific DH constants respectively) to enable passive decryption. However, there is no credible evidence that they can break the higher-strength cryptography now used by most popular sites.

5.6.1 The STEK as an Enabling Vector

It is well known that the NSA and other intelligence agencies have the ability to passively collect vast amounts of Internet traffic. Some collection is "targeted" at a specific person, website, or IP address, but other collection involves indiscriminately storing all network traffic in large circular buffers, such as XKEYSCORE [226] and TEMPORA [147], for *ex post facto* analysis [417].

These capabilities are almost certainly challenged by the growth of TLS, which has

accelerated following increased public awareness of surveillance [231] and the availability of free browser-trusted certificates [180]. Faced with these constraints, nation-state adversaries might find that session tickets provide an appealing mode of attack. Exfiltrating one 16-byte STEK from a server would allow the adversary to decrypt every passively collected connection which uses the TLS session ticket extension during the vulnerability window, including connections within the window but before the STEK was leaked. As seen in Figures 5.6, stealing a small number of STEKs would enable decryption of content from a large number of domains.

Although obtaining a STEK may require attacking the provider and not the end-target, intelligence agencies have been known to conduct sophisticated intrusions in order to facilitate later passive surveillance. GCHQ infiltrated SIM card manufacturer Gemalto to steal the encryption keys used by millions of cellphones [60, 328, 356]. They also attacked engineers at Belgacom, the largest ISP in Belgium, in order to gain access to traffic from its core routers [128]. An unknown adversary—thought to be a nation state—infiltrated Juniper Networks' code repository and inserted a cryptographic backdoor into the company's VPN products [53]. Similar operations could be used to access STEKs from high-value targets.

It is likely that some domains synchronize STEKs across servers in many network locations and jurisdictions. A nation-state attacker could attempt to compromise the synchronization mechanism, or they could convince a hosting facility to grant them access to the equipment for physical attacks [13]. Within its national borders, such an attacker might use the court system to compel an organization to turn over the STEKs, as Lavabit was ordered to do with its TLS private key [336]. However obtained, the STEK would provide global decryption capabilities.

5.6.2 Target Analysis: Google

To provide a concrete example, we simulate a nation-state attacker's possible analysis of an attack against Google—a large tech company with experience being attacked by [152, 430] and defending against [129] nation-state adversaries. As the attacker, our goal is to leverage our existing passive collection systems—which currently only see TLS ciphertext—in order to gain insight into a large swath of network communication.

As seen in Table 5.6, a single STEK is shared by nearly all Google web services, including Search, GMail, Drive, Docs, Hangouts, and many more. We find that Google also uses the same STEK for other TLS-based protocols, including SMTP+STARTTLS, SMTPS, IMAPS, and POP3S. We experimentally determined that Google's STEK is rolled over

every 14 hours, but issued tickets are accepted for up to 28 hours, indicating that each key is maintained at least that long. This implies that only two 16-byte keys must be stolen every 28 hours in order for the attacker to be able to decrypt all Google TLS connections that use the session ticket extension.

By requesting the MX records for the Alexa Top Million domains, we find that over 90,000 domains (9.1%) point to Google's SMTP servers. This is likely a reflection of the Google for Work program in which more than 2 million businesses (including 60% of Fortune 500 companies) use Google's service for their internal and external e-mail [158]. So in addition to the e-mail communications and web-app data from `@gmail.com` addresses, the content of any company which relies on Google's cloud service for intracompany e-mail or web-apps would be decryptable.

The intelligence value from the resulting decryption ability would extend far beyond Google's own properties. Google supplies analytics, ads, and APIs to many websites whose requests would likely send the user's Google cookies. We have confirmed that browser connections to these Google dependencies use the same STEK as other Google sites. Obtaining the Google STEK would allow tracking users even when they are not directly accessing Google sites.

As this analysis shows, Google's STEK would be an immensely valuable target, as it would enable the decryption of a huge amount of encrypted traffic and provide intelligence on targeted and untargeted individuals. Even if the exploitation required the use of sophisticated, persistent hardware or software implants, the trade off between the possibility of their discovery and the rich intelligence that would be gained likely falls within the acceptable risks category for many nation-state adversaries.

Google's is the case of a well protected organization with a highly talented security team. As shown in Section 5.3.3, many other organizations—including large tech and cloud service companies—appear to be far less cognizant of the risks of TLS performance enhancements. As an example, Yandex is a Russian Internet company that mirrors Google's offerings in search, e-mail, and cloud storage and enjoys a 57% domestic market share [207]. Like Google, Yandex appears to use a single STEK for almost all of its properties, but unlike Google, this STEK has been in use continuously since at least January 10, 2016—eight months prior to this writing. A single operation to recover this STEK would immediately allow decryption of months' worth of connections.

5.7 Discussion

While we've notified the domains and companies that we explicitly point out above, there are other ways to address the ecosystem-wide issues we found. In this section we step back and view the problems found with (EC)DHE values, session caches, and session tickets from a community level. We draw lessons from our measurements and make recommendations for server operators.

5.7.1 Security Community Lessons

The security community's advocacy for the adoption of TLS forward secrecy has shown clear gains, with over 90% of Top Million HTTPS sites now using forward secret key exchanges for modern browsers. And the use of forward secret key exchanges is undoubtedly a vast security improvement from non-forward secret exchanges. However, our results are a reminder that cipher selection is only one part of the story.

Forward secrecy comes with many critical caveats and nuances of implementation [343]. The security community needs to do a better job of monitoring implementation behavior—through measurements like the ones we present here—in order to have a realistic understanding of the threats we need to guard against.

The security community also needs to more clearly communicate such caveats to TLS server operators and implementers so that they can make informed choices about security/performance tradeoffs. Absent such knowledge, there is a risk that forward-secret TLS handshakes can create a false sense of security. In the aftermath of the Heartbleed vulnerability, security experts urged administrators to enable PFS ciphers in order to guard against retrospective decryption as a result of future server-side memory leaks [85, 431]. However, only a few experts ever noted that performance enhancements like session resumption undermine that protection [343], and the fact seems to have been largely overlooked. The next time there there is such a vulnerability, administrators who enabled PFS as a defense might mistakenly believe they are safe.

One opportunity to begin such education is protocol standards. As described in Section 5.1.4, the TLS 1.3 draft proposes changes that have direct consequences for the protocol's vulnerability window. Draft 15 briefly addresses the changes to forward secrecy cased by PSK connections and 0-RTT, but simply sets a 7 day maximum for PSK lifetimes without discussion. As shown above, PSKs honored for 7 days (whether database lookups or encrypted state) require TLS secrets to exist for the same amount of time and may be a significant risk for high-value domains.

5.7.2 Server Operators Recommendations

For maximum security, server operators should disable all session resumption and Diffie-Hellman reuse. And while we are aware that many operators will be understandably unwilling to do so due to the bandwidth, computation, and latency advantages, there is a middle-ground between the two that limits vulnerability windows as well as allows the performance enhancements.

Use HTTP/2 Using HTTP/2 [27] drastically reduces the computation, bandwidth, and latency of loading a website without requiring any crypto shortcuts. An entire domain's contents (base page and all dependencies) can be loaded over a single TLS connection. This results in the time-to-first byte on the first request being identical to standard HTTP over TLS, but all follow-on requests are significantly faster without expanding the PFS vulnerability window.

Rotate STEKs frequently Reducing the time period that a STEK is used to encrypt session tickets is the simplest way to reduce the vulnerability window when using session ticket resumption. While Figure 5.3 shows that many domains are already doing this, it also shows that many are not. Twitter, CloudFlare, and Google have all created their own custom key rotation solutions [173, 209, 216], but, to our knowledge, no popular server software does this, with the exception of the most recent release of Caddy [176].

Use different STEKs for different regions Rather than sharing a single session ticket key globally, large sites should seek geographical diversity by using different keys in different regions. In addition to limiting exposure if a single server is compromised or physically attacked, this practice would help constrain the effects of legally mandated STEK disclosure to connections within a particular jurisdiction.

Reduce session cache lifetimes Specific to session ID resumption, quickly expiring cached session state is also useful. The number of connections that are at risk of decryption at any time grows proportionally with the lifetime of the server-side state. By measuring the duration of a typical user visit, operators can use that to ensure that a user only has to conduct one full handshake per visit but also refrain from retaining the session state longer than necessary.

Store, distribute, and erase secrets securely TLS implementations need to ensure that TLS secrets handled securely before, during, and after their use. For a small site, these details

should be handled by the TLS implementation. But for more complicated deployments that involve synchronizing caches or STEKs across multiple servers, operators need to be more directly involved. Whatever mechanism they design to synchronize STEKs needs to ensure that these keys are transmitted securely and maintained only in memory (rather than persistent storage), so that they can be reliably discarded.

5.8 Conclusion

We conducted a 9-week study of HTTPS within the Alexa Top Million with a focus on understanding both the prevalence and characteristics of TLS performance enhancements such as (EC)DHE value reuse, session ID resumption, and session ticket resumption. Through this study, we were able to characterize the effects of cryptographic shortcuts on the promises associated with the use of forward-secret ciphers. Our findings show that the TLS ecosystem achieves much weaker protection from forward secrecy than statistics about support for forward-secret handshakes would suggest. They also emphasize the need for the security community to clearly communicate the relevant tradeoffs between security and performance to server operators.

Nation-State Attacker Model When we view this research in the context of our NS-Attacker model, we see multiple places where the attributes of our NS-Attacker model apply. The first and most obvious is NS-Attackers' advantage of Non-Symmetric Defeats. The resources investment required to exploit an SSL terminator to acquire the STEK or session cache as well as the investment to compute the discrete log a long-term DH key share is assumed to be large. But if successful, this resource investment allows content decryption at near zero cost per connection. This fits perfectly with the large one-time cost, trivial marginal cost description type of defeat described in Section 2.3.3.

NS-Attackers' characteristic of Sovereignty can drastically reduce the one-time resource investment required for the asymmetric defeat. As described in Section 2.2.1, NS-Attackers can use legal writs to obtain data from a provider. While it may be technically infeasible, or at least difficult, to extract any of the shortcut secrets when in default configurations, a STEK or session cache shared across multiple terminators is likely to be recoverable with little to moderate effort.

With regard to the constraint of Required Hard Victims, this type of attack could contribute to both the collect-it-all and retrospective mitigations. Research found that a small number of domains are dependencies of an amazingly large number of the Alexa Top Million domains. Of the Top 10 domains loaded by the Alexa Top Million, 8 are

Google properties including `google-analytics.com` which is loaded by 67.8%. Acquiring Google's STEK would allow the attacker to not only defeat HTTPS protection when victims are on Google properties, but also enable victim tracking when not on Google properties via Google cookies.

CHAPTER 6

Future Work and Conclusion

In only the last decade, Nation-State Attackers have emerged as a recognized threat to not only the security and privacy of other state actors, but also as a threat to corporate entities and citizens of all countries as a whole. And while some factions of the security community were already cognizant of the potential impact, the security community as a whole and the larger technical community were largely caught off guard by the disclosure of NS-Attackers' behavior.

In the years since, large and sweeping efforts have been made to curb these activities as well as construct barriers to other potential NS-Attacker endeavors. But as we discussed in Chapter 2, NS-Attackers are an agile attacker who can not be expected to concede the loss of intelligence capabilities nor overlook vulnerabilities which may present themselves in the future.

In this chapter, we first identify and briefly discuss avenues of future research and exploration which we believe provide appealing improvements to defenses against NS-Attackers. Finally, we conclude with a brief summary of our research efforts and our previous discussion.

6.1 Going Forward

When we examine Nation-State Attackers in the context of our previously presented model and case studies, we see that Nation-State Attacker have a potentially immense impact on computer security. In order to ameliorate their potential effects, the security community must re-examine many of the fundamental assumptions and constructions of the Internet as it currently exists. In this section, we identify and briefly discuss avenues of research which we believe many prove to be useful in addressing the dangers of NS-Attackers.

Personal Clouds As we discussed in Section 2.6, cloud services present an attractive objective for NS-Attackers. The concentrations of users and data increase the likelihood that an NS-Attacker would target them for exploitation whether for watering-hole style attacks or to exploit the veritable "data lake" of information. An appealing avenue of future research and investment is in breaking up these large concentrations by way of isolating the users and their data from each other.

The obvious disadvantage of this type of partitioning is that it also splits the efforts to secure these concentrations as described in Section 2.6. One approach to addressing this is to abstract the security and privacy decisions away from the individual user in favor of secure-and-private-by-default configurations. Project such as ownCloud [322] and Mail-in-a-Box [223] provide drop-in replacements for popular cloud use cases such as file sharing and e-mail hosting. Lowering the barrier to entry and improving the default security characteristics of systems such as these has shown to improve the security of the ecosystems and users as a whole [194].

Security Bastions Where as personal clouds are envisioned to improve the security and privacy of users' data, organizations typically deal with substantially more data and interactions. While some organizations have the financial and technical resources to migrate away from shared cloud services, it is a substantial undertaking. Instead, we see promise in the use of on-site security bastions as a drop-in security improvement.

With these bastions, organizations could still take advantage of cloud services for storing and retrieving large amounts of data, but maintain control of the security of that data. As described in Section 2.6, many cloud providers control the at-rest encryption keys which substantially reduces the protection of encrypting that data. These security bastions allow each organization to control their own key material on-site. As envisioned, these bastions would operate similar to an HTTP proxy which would act as a middle-man encrypting or decrypting data on its way to the cloud service.

Assumption Identification and Verification Lastly, we believe that it is important for the security and research community to continue identifying and investigating assumptions made about how users and implementers realize standards. As seen in Chapter 4 and Chapter 5, expectations about the use of cryptographic parameters do not always match reality. Additionally, research into how people use security and privacy is on going and still evolving.

6.2 Conclusion

Since 2013, Nation-State Attackers have been thrust into the public's perception by a series of leaked documents, declassified reports, and attributed attacks. This new information shows a level of feasibility, scale, and maliciousness that had previously been largely dismissed by the security community as a whole. Additionally, many documents included oracles of effective attacks without technical detail or indication of still strong alternatives to the now weak systems and protocols. Through wide-ranging and intensive efforts, the security community was able to reverse engineer and/or rediscover many of these attacks and adjust the "Best Practices" to account for them. One substantial drawback of this type of effort is that it inherently relies on an oracle of what is possible. Without proactively studying and accounting for the fundamental attributes of Nation-State Attackers, the security community must await the next disclosure and the next oracle to mitigate the next attack.

In Chapter 2, we presented our high-level model of Nation-State Attackers. In addition to defining a direct and concise lexicon, we also identified the defining characteristics of, advantages of being, and constraints faced by Nation-State Attackers. We then proposed ways in which Nation-State Attackers' operations and behaviors differ from each other as well as aspects of Nation-State Attackers that individuals, organizations, and researchers should be cognizant of when designing or implementing systems and protocols.

In Chapter 3, we presented our 2014 publication "Security Analysis of the Estonian Internet Voting System" which showed the impact that Nation-State Attackers described by our model can have on real-world threat models. Through a combination of their characteristic of Access and advantages of Near Superset Attacker and Specialization, we showed that the Estonian Internet Voting System could not withstand attacks from real-world adversaries and was susceptible to interference by Nation-State Attackers.

In Chapter 4, we discussed our 2015 publication "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice" which showed how our Nation-State Attacker model can be used to explain and provide concrete details for known but ambiguous Nation-State Attacker operations. The asymmetric properties of the Index Calculus algorithm corresponded with Nation-State Attackers' advantage of Non-Symmetric Defeats in highly beneficial ways. Additionally, Nation-State Attackers' characteristic of Money as well as the advantages of Specialization and Distant Return Horizons also conformed to the requirements of the described attack. All of these factors, along with our Internet measurements to determine the impact of our attack if realized, supported our assertion that the NSA is likely capable of decrypting large numbers of VPN connections by efficiently defeating the Diffie-Hellman key exchange using Index Calculus.

In Chapter 5, we discussed our 2016 publication "Measuring the Security Harm of TLS Crypto Shortcuts" which showed how our Nation-State Attacker model can be used to find vulnerabilities without the aid of an existence oracle. Much like the Index Calculus algorithm, the real-world usage of TLS Session Tickets presented asymmetric properties that correspond with Nation-State Attackers' advantage of Non-Symmetric Defeats. TLS Session Tickets also provided a fitting mitigation to Nation-State Attackers' constraint of Required Hard Victims as they can be combined with the Collect-it-All mentality efficiently defeating large numbers of encrypted communications. When viewed with the characteristic of Sovereignty, alternatives for the asymmetric defeat's one-time effort provide additional forms of this defeat.

In conclusion, we believe that our proposed Nation-State Attacker model is a needed and useful foundation to security researchers, practitioners, and the community as a whole when confronting Nation-State Attackers. By stepping back from the intricate and nuanced details of known operations, our identification and discussion of Nation-State Attackers' characteristics, advantages, and constraints provides a much more manageable and general view of these highly privileged attackers. This perspective allows us to not only identify current vulnerabilities that Nation-State Attackers may exploit, but also to proactively identify types of system constructions which are beneficial to Nation-State Attackers and their attributes. Going forward, we believe that our model can be used by the wider security community to account for Nation-State Attackers and improve the security of users across the Internet.

BIBLIOGRAPHY

[1] Networkminer. http://www.netresec.com/?page=NetworkMiner.

[2] Selector types. Media leak. https://theintercept.com/document/2014/03/12/selector-types/.

[3] Untangling the web: A guide to Internet research. Declassified Document, Feb. 2007. https://www.nsa.gov/news-features/declassified-documents/assets/files/Untangling-the-Web.pdf.

[4] Intelligence and security committee of parliament annual report 2015-2016, July 2016. http://isc.independent.gov.uk/files/2015-2016_ISC_AR.pdf.

[5] L. Ablon and A. Bogart. Zero days, thousands of nights. Technical report, RAND Corporation, 2017.

[6] ACLU v. Clapper - challenge to NSA mass call-tracking program, oct 2015. https://www.aclu.org/cases/aclu-v-clapper-challenge-nsa-mass-call-tracking-program.

[7] B. Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th USENIX Security Symposium*, Aug. 2008.

[8] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[9] Alexa Internet, Inc. Alexa Top 1,000,000 Sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[10] K. Alexander. Opening statement of Gen keith b. alexander, Director, NSA before the Senate Committee on the Judiciary, Oct. 2013. https://www.judiciary.senate.gov/imo/media/doc/10-2-13AlexanderTestimony.pdf.

[11] C. Allen and T. Dierks. The TLS Protocol Version 1.0. RFC 2246, Jan. 1999.

[12] M. Ambinder. What the NSA's massive org chart (probably) looks like. Defense One, Aug. 2013. http://www.defenseone.com/ideas/2013/08/what-nsas-massive-org-chart-probably-looks/68642/.

[13] J. Angwin, C. Savage, J. Larson, H. Moltke, L. Poitras, and J. Risen. AT&T helped U.S. spy on Internet on a vast scale. The New York Times, Aug. 16, 2015. http://www.nytimes.com/2015/08/16/us/politics/att-helped-nsa-spy-on-an-array-of-internet-traffic.html.

[14] A. Ansper, A. Buldas, M. Oruaas, J. Priisalu, A. Veldre, J. Willemson, and K. Virunurm. E-voting concept security: analysis and measures. Technical Report EH-02-01, Estonian National Electoral Committee, 2003.

[15] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding the Mirai botnet. Aug. 2017.

[16] A. W. Appel. Security seals on voting machines: A case study. *ACM Trans. Inf. Syst. Secur.*, 14(2):18:1–18:29, Sept. 2011.

[17] J. Applebaum, J. Horchet, and C. Stöcker. Shopping for spy gear: Catalog advertises NSA toolbox. Der Spiegel, Dec. 2013. http://www.spiegel.de/international/world/catalog-reveals-nsa-has-back-doors-for-numerous-devices-a-940994.html.

[18] J. Ashcroft. Procedures for the dissemination by NSA to foreign governments of information from FISA electronic surveillance or physical search conducted by the FBI, Aug. 2002. https://assets.documentcloud.org/documents/1061347/ashcroft-new-dissemination-procedures.pdf.

[19] D. Auerbach, J. Mayer, and P. Eckersley. What we need to know about PRISM. Electronic Frontier Foundation, June 2013. https://www.eff.org/deeplinks/2013/06/what-we-need-to-know-about-prism.

[20] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, E. Käsper, S. Cohney, S. Engels, C. Paar, and Y. Shavitt. DROWN: Breaking TLS with SSLv2. In *25th USENIX Security Symposium*, Aug. 2016. https://drownattack.com.

[21] Introduction to BADDECISION. Media leak, Dec. 2010. https://assets.documentcloud.org/documents/3031639/07-Introduction-to-BADDECISION-Redacted.pdf.

[22] S. Bai, C. Bouvier, A. Filbois, P. Gaudry, L. Imbert, A. Kruppa, F. Morain, E. Thomé, and P. Zimmermann. `cado-nfs`, an implementation of the number field sieve algorithm, 2014. Release 2.1.1.

[23] J. Ball. NSA's Prism surveillance program: how it works and what it can do. *The Guardian*, June 2013. https://www.theguardian.com/world/2013/jun/08/nsa-prism-server-collection-facebook-google.

[24] R. Barbulescu. *Algorithmes de logarithmes discrets dans les corps finis*. PhD thesis, Université de Lorraine, France, 2013.

[25] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Eurocrypt*, 2014.

[26] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. *NIST Special Publication 800-57: Recommendation for Key Management*, 2007.

[27] M. Belshe, R. Peon, and M. Thomson. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, May 2015.

[28] J. Benaloh, M. Byrne, P. T. Kortum, N. McBurnett, O. Pereira, P. B. Stark, and D. S. Wallach. STAR-Vote: A secure, transparent, auditable, and reliable voting system. *CoRR*, abs/1211.1904, 2012.

[29] D. J. Bernstein. How to find smooth parts of integers, 2004. http://cr.yp.to/factorization/smoothparts-20040510.pdf.

[30] D. J. Bernstein. Break a dozen secret keys, get a million more for free, Nov. 2015. https://blog.cr.yp.to/20151120-batchattacks.html.

[31] D. J. Bernstein and T. Lange. Batch NFS. In *Selected Areas in Cryptography*, 2014.

[32] D. J. Bernstein, T. Lange, and R. Niederhagen. Dual EC: A standardized back door. In *The New Codebreakers*, pages 256–281. Springer, 2016.

[33] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *IEEE Symposium on Security and Privacy*, 2015.

[34] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *36th IEEE Symposium on Security and Privacy*, May 2015.

[35] B. Binde, R. McRee, and T. J. OConnor. Assessing outbound traffic to uncover advanced persistent threat. *SANS Institute. Whitepaper*, 2011.

[36] Structure of the BND. Media leak. http://www.spiegel.de/media/media-34050.pdf. Archived Nov 24, 2016: https://web.archive.org/web/20161124013106/http://www.spiegel.de/media/media-34050.pdf.

[37] Bundesnachrichtendienst - information collection. http://www.bnd.bund.de/EN/Scope_of_Work/Information_Collection/Information_Collection_node.html.

[38] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008.

[39] A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique cryptanalysis of the full AES. In *International Conference on the Theory and Application of Cryptology and Information Security*, 2011.

[40] C. Bouvier, P. Gaudry, L. Imbert, H. Jeljeli, and E. Thomé. New record for discrete logarithm in a prime finite field of 180 decimal digits, 2014. http://caramel.loria.fr/p180.txt.

[41] J. Bretschneider, S. Flaherty, S. Goodman, M. Halvorson, R. Johnston, M. Lindeman, R. L. Rivest, P. Smith, and P. B. Stark. Risk-limiting post-election audits: Why and how, Oct. 2012. http://www.stat.berkeley.edu/~stark/Preprints/RLAwhitepaper12.pdf.

[42] E. Burzstein and J. M. Picod. Recovering Windows secrets and EFS certificates offline. In *4th USENIX Workshop on Offensive Technologies*, Aug. 2010.

[43] S. Buttar. Apple, Americans, and security vs. FBI, Feb. 2016. https://www.eff.org/deeplinks/2016/02/apple-americans-and-security-vs-fbi.

[44] J. A. Calandrino, J. A. Halderman, and E. W. Felten. Machine-assisted election auditing. In *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*, 2007.

[45] R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *Crypto*, 2002.

[46] Computer file backup software & data protection. https://www.carbonite.com/.

[47] D. Carrel and D. Harkins. The Internet Key Exchange (IKE). RFC 2409, Nov. 1998.

[48] K. R. Carter. Cloudflare's Transparency Report for second half of 2016 and an additional disclosure for 2013, Jan. 2017. https://blog.cloudflare.com/cloudflares-transparency-report-for-second-half-2016-and-an-additional-disclosure-for-2013-2/.

[49] Cavium. Intelligent network adapters. http://www.cavium.com/Intelligent_Network_Adapters_NIC4E.html.

[50] Certificate transparency. https://www.certificate-transparency.org/.

[51] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, Jan 2004.

[52] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. Rivest, P. Y. A. Ryan, E. Shen, A. Sherman, and P. Vora. Scantegrity II: End-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE Transactions on Information Forensics and Security*, 4(4):611–627, Dec. 2009.

[53] S. Checkoway, J. Maskiewicz, C. Garman, J. Fried, S. Cohney, M. Green, N. Heninger, R.-P. Weinmann, E. Rescorla, and H. Shacham. A systematic analysis of the Juniper Dual EC incident. In *23rd ACM Conference on Computer and Communications Security*, Oct. 2016.

[54] S. Checkoway, R. Niederhagen, A. Everspaugh, M. Green, T. Lange, T. Ristenpart, D. J. Bernstein, J. Maskiewicz, H. Shacham, M. Fredrikson, et al. On the practical exploitability of Dual EC in TLS implementations. In *USENIX security symposium*, pages 319–335, 2014.

[55] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig. HORNET: High-speed onion routing at the network layer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1441–1454. ACM, 2015.

[56] F. Church, J. Tower, P. Hart, H. Baker, W. Mondale, B. Goldwater, W. Huddleston, C. Mathias, R. Morgan, R. Schweiker, G. Hart, W. Miller, F. Schwarz, C. Smothers, and A. Hatry. Intelligence activities and the rights of Americans: Book ii. U.S. Government Printing Office, Apr. 1976. https://www.intelligence.senate.gov/sites/default/files/94755_II.pdf.

[57] Cisco SAFE for small enterprice networks, July 2010. https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Security/SAFE_RG/safesmallentnetworks.html.

[58] M. Clayton. Ukraine election narrowly avoided 'wanton destruction' from hackers. Christian Science Monitor, June 2014. http://www.csmonitor.com/World/Security-Watch/Cyber-Conflict-Monitor/2014/0617/Ukraine-election-narrowly-avoided-wanton-destruction-from-hackers-video.

[59] Suitablity and security processes review: Report to the President, Feb. 2014. https://www.archives.gov/files/isoo/oversight-groups/nisp/2014-suitability-and-processes-report.pdf.

[60] CNE access to core mobile networks. Media leak. https://theintercept.com/document/2015/02/19/cne-access-core-mobile-networks-2/.

[61] Codenomicon. The Heartbleed bug. http://heartbleed.com/.

[62] M. C. T. Command. Rifle platoon in the defense – b3j3778 – student handout. http://www.trngcmd.marines.mil/Portals/207/Docs/TBS/B3J3778%20Rifle%20Platoon%20in%20the%20Defense.pdf.

[63] A. Commeine and I. Semaev. An algorithm to solve the discrete logarithm problem with the number field sieve. In *PKC*, 2006.

[64] New contact-chaining procedures to allow better, faster analysis. Media leak, Jan. 2011. https://www.aclu.org/files/assets/Contact_Chaining_Memo_2011_01_03.pdf.

[65] A. Cooper, H. Tschofenig, D. B. D. A. Ph.D., J. Peterson, J. B. Morris, M. Hansen, and R. Smith. Privacy Considerations for Internet Protocols. RFC 6973, July 2013.

[66] J. Cooper. The funding of the power agencies of the Russian state. *The Journal of Power Institutions in Post-Soviet Societies. Pipss. org*, (6/7), 2007.

[67] D. Coppersmith. Solving linear equations over GF(2) via block Wiedemann algorithm. *Math. Comp.*, 62(205), 1994.

[68] R. Crandall and C. B. Pomerance. *Prime Numbers: A Computational Perspective*. Springer, 2001.

[69] C. Cremers. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. *Computer Security–ESORICS 2011*, 2011.

[70] J. M. Crewdson. C.I.A. men opened 3 Senators' mail and note to Nixon. The New York Times, 1975. http://www.nytimes.com/1975/09/25/archives/cia-men-opened-3-senators-mail-and-note-to-nixon-panel-says-aides.html.

[71] TLS trends : A roundtable discussion on current usage and future directions. Media leak. http://www.spiegel.de/media/media-35510.pdf. Archived Apr 7, 2017: https://web.archive.org/web/20170407201656/http://www.spiegel.de/media/media-35510.pdf.

[72] Cybernetica AS. Internet voting solution, 2013. Accessed: May 13, 2014, http://cyber.ee/uploads/2013/03/cyber_ivoting_NEW2_A4_web.pdf.

[73] D. Danchev. Study finds the average price for renting a botnet. ZDNet, May 2010. http://www.zdnet.com/blog/security/study-finds-the-average-price-for-renting-a-botnet/6528.

[74] DEFIANTWARRIOR and the NSA's use of bots. Media leak. http://www.spiegel.de/media/media-35689.pdf. Archived Oct 12, 2016: https://web.archive.org/web/20161012140326/http://www.spiegel.de/media/media-35689.pdf.

[75] B. den Boer. Diffie-Hellman is as strong as discrete log for certain primes. In *Crypto*, 1988.

[76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22(6):644–654, 1976.

[77] W. Diffie and M. E. Hellman. Special feature exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10(6):74–84, 1977.

[78] W. Diffie, P. C. Van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and cryptography*, 2(2):107–125, 1992.

[79] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[80] DoD mobility classified capability - secret 2.0.5, Mar. 2016. http://www.disa.mil/~/media/files/disa/fact-sheets/dmcc-s.pdf.

[81] Digital Reciever Technology. https://www.drti.com/.

[82] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. Censys: A search engine backed by Internet-wide scanning. In *22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[83] Z. Durumeric, J. A. Halderman, et al. Internet-wide scan data repository. https://scans.io.

[84] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS certificate ecosystem. In *13th ACM Internet Measurement Conference*, IMC '13, pages 291–304, 2013.

[85] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman. The matter of Heartbleed. In *14th ACM Internet Measurement Conference*, IMC '14, pages 475–488, 2014.

[86] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *22nd USENIX Security Symposium*, Aug. 2013.

[87] National Security Letters. https://www.eff.org/issues/national-security-letters.

[88] EFF sues DOJ for records on procedures for ending NSL gag orders, June 2017. https://www.eff.org/press/releases/eff-sues-fbi-records-procedures-ending-nsl-gag-orders.

[89] The government's word games when talking about NSA domestic spying. https://www.eff.org/nsa-spying/wordgames.

[90] R. Ensafi, D. Fifield, P. Winter, N. Feamster, N. Weaver, and V. Paxson. Examining how the great firewall discovers hidden circumvention servers. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 445–458. ACM, 2015.

[91] R. Ensafi, P. Winter, A. Mueen, and J. R. Crandall. Analyzing the great firewall of china over space and time. *Proceedings on privacy enhancing technologies*, 2015(1):61–76, 2015.

[92] P. Eronen, H. Tschofenig, H. Zhou, and J. A. Salowey. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077, Jan. 2008.

[93] For the user of ID-card and mobile ID, Nov. 2017. https://www.politsei.ee/en/nouanded/isikut-toendavad-dokumendid/id-kaardi-ja-mobiil-id-kasutajale.dot.

[94] National Electoral Committee: e-voting will take place, Oct. 2017. https://www.valimised.ee/en/news/national-electoral-committee-e-voting-will-take-place.

[95] Estonian Certification Authority. Avaleht. In Estonian. http://www.id.ee/.

[96] Estonian Certification Authority. ID-tarkvara. In Estonian. https://installer.id.ee/.

[97] Estonian Certification Authority. Kasulik tugiinfo mobiil-id kohta. In Estonian. http://mobiil.id.ee/kasulik-tugiinfo/.

[98] Estonian Information System's Authority. Public key infrastructure PKI, May 2012. https://www.ria.ee/public-key-infrastructure/.

[99] Estonian Internet Voting Committee. Dokumendid, 2013. In Estonian. Accessed: March 2014, http://vvk.ee/valijale/e-haaletamine/e-dokumendid/.

[100] Estonian Internet Voting Committee. Ehk videos, 2013. In Estonian. Accessed: March 2014, https://www.youtube.com/channel/UCTv2y5BPOo-ZSVdTg0CDIbQ/videos.

[101] Estonian Internet Voting Committee. Statistics about Internet voting in Estonia, May 2014. http://www.vvk.ee/voting-methods-in-estonia/engindex/statistics.

[102] Estonian Internet Voting Committee. Using ID-card and mobil-ID, May 2014. https://www.valimised.ee/eng/kkk.

[103] Estonian Ministry of Foreign Affairs. Estonia today, 2012. http://www.euc.illinois.edu/estonia/documents/E-Estonia.pdf.

[104] Estonian National Electoral Committee. Kohaliku omavalitsuse volikogu valimised 2013. In Estonian. http://www.vvk.ee/kohalikud-valimised-2013/.

[105] Estonian National Electoral Committee. Vabariigi valimiskomisjon. In Estonian. Accessed: October 2013, http://www.vvk.ee/.

[106] Estonian National Electoral Committee. Elektroonilise hääletamise süsteemi üldkirjeldus, 2013. In Estonian. http://vvk.ee/public/dok/elektroonilise-haaletamise-systeemi-yldkirjeldus-EH-03-03-1_2013.pdf.

[107] Estonian National Electoral Committee. Valimised: Android Apps on Google Play, Oct. 2013. In Estonian. Accessed: May 13, 2014, https://play.google.com/store/apps/details?id=ee.vvk.ivotingverification.

[108] Estonian National Electoral Committee. Comment on the article published in The Guardian, May 2014. http://vvk.ee/valimiste-korraldamine/vvk-uudised/vabariigi-valimiskomisjoni-vastulause-the-guardianis-ilmunud-artiklile/.

[109] Estonian National Electoral Committee. Valimised on the App Store on iTunes, Apr. 2014. In Estonian. Accessed: May 15, 2014, https://itunes.apple.com/ee/app/valimised/id871129256.

[110] Estonian National Electoral Committee. Valimised: Windows Phone'i rakenduste+mängude pood (Eesti), Apr. 2014. In Estonian. Accessed: May 15, 2014, https://www.windowsphone.com/et-ee/store/app/valimised/11c10268-343f-461a-9c73-630940d8234b.

[111] Estonian National Electoral Committee, Estonian Internet Voting Committee, and Cybernetica AS. Android based vote verification application for Estonian i-voting system, Sept. 2013. https://github.com/vvk-ehk/ivotingverification.

[112] Estonian National Electoral Committee, Estonian Internet Voting Committee, and Cybernetica AS. e-hääletamise tarkvara, Sept. 2013. Accessed: March 2014, https://github.com/vvk-ehk/evalimine.

[113] Estonian Public Broadcasting. Center Party petitions European human rights court over e-voting, Sept. 2013. Accessed: May 14, 2014, http://news.err.ee/v/politics/4ee0c8a2-b9c2-4d28-8ae4-061e7d9386a4.

[114] Exploit database by Offensive Security. https://www.exploit-db.com/.

[115] Government request report. https://govtrequests.facebook.com/.

[116] N. Falliere, L. O. Murchu, and E. Chien. W32.Stuxnet dossier, Feb. 2011. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

[117] Verbatim transcript - fbi and reality winner. Politico. https://www.politico.com/f/?id=0000015e-c5e7-ddab-a57f-cfe7d5b50002.

[118] A. P. Felt. Twitter, Jan. 2016. https://twitter.com/__apf__/status/684865174373707777.

[119] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz. Measuring HTTPS adoption on the web. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1323–1338, Vancouver, BC, 2017. USENIX Association.

[120] In re production of tangible things from [redacted], Mar. 2009. https://www.dni.gov/files/documents/section/pub_March%202%202009%20Order%20from%20FISC.pdf.

[121] J. Fleming. EU nations developing cyber 'capabilities' to infiltrate government, private targets. EurActiv, Dec. 2013. http://www.euractiv.com/infosociety/eu-nations-lack-common-approach-news-532294.

[122] A. C. for Country of Origin & Asylum Research and Documentation. Iran: Capacity and methods of authorities to monitor online activities and religious activities of Iranians living abroad, June 2017. https://www.justice.gov/eoir/page/file/975076/download.

[123] E. F. Foundation. The legal FAQ for Tor relay operators, Apr. 2014. https://www.torproject.org/eff/tor-legal-faq.html.en.

[124] J. Franke, T. Kleinjung, C. Paar, J. Pelzl, C. Priplata, and C. Stahlke. SHARK: a realizable special hardware sieving device for factoring 1024-bit integers. *Cryptographic Hardware and Embedded Systems–CHES 2005*, pages 119–130, 2005.

[125] L. Freeh. Statement of Louis J. Freeh, director Federal Bureau of Investigation before the Senate Judiciary Committee, July 1997. https://www.epic.org/crypto/legislation/freeh_797.html.

[126] M. Friedl, N. Provos, and W. A. Simpson. Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4419, Mar. 2006.

[127] P. Fuhrman. Government cyber-surveillance is the norm in China and its popular. The Washington Post, Jan. 2016. https://www.washingtonpost.com/opinions/cyber-surveillance-is-a-way-of-life-in-china/2016/01/29/e4e856dc-c476-11e5-a4aa-f25866ba0dc6_story.html.

[128] R. Gallagher. Operation Socialist. The Intercept, Dec. 13, 2014. https://theintercept.com/2014/12/13/belgacom-hack-gchq-inside-story/.

[129] S. Gallagher. Googlers say "f*** you" to NSA, company encrypts internal network. Ars Technica, Nov. 2013. http://arstechnica.com/information-technology/2013/11/googlers-say-f-you-to-nsa-company-encrypts-internal-network/.

[130] E. Galperin and D. O'Brien. Russia asks for the impossible with its new surveillance laws, July 2016. https://www.eff.org/deeplinks/2016/07/russia-asks-impossible-its-new-surveillance-laws.

[131] L. Gaspar, L. Beslay, and I. Coisel. FPGA performances in cryptography. 2014.

[132] Innov8 experiment profile. Media leak. http://www.spiegel.de/media/media-35509.pdf. Archived Apr 7, 2017: https://web.archive.org/web/20170407200510/http://www.spiegel.de/media/media-35509.pdf.

[133] GCHQ analytic cloud challenges. Media leak, May 2012. https://assets.documentcloud.org/documents/2432286/gchq-analytic-cloud-challenges.pdf.

[134] Mobile networks in my NOC world. Media leak. https://assets.documentcloud.org/documents/1379028/gchq-mobile-networks-in-my-noc-world.pdf. Archived Jan 28, 2015: https://web.archive.org/web/20150128051544/https://s3.amazonaws.com/s3.documentcloud.org/documents/1379028/gchq-mobile-networks-in-my-noc-world.pdf.

[135] Bullrun coi - briefing sheet. Media leak. http://www.spiegel.de/media/media-35531.pdf. Archived Sept 30, 2017: https://web.archive.org/web/20170930090009/http://www.spiegel.de/media/media-35531.pdf.

[136] Content or metadata? Media leak, Jan. 2010. https://assets.documentcloud.org/documents/2432272/content-metadata-matrix.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121215712/https://s3.amazonaws.com/s3.documentcloud.org/documents/2432272/content-metadata-matrix.pdf.

[137] HIMR data mining research problem book. Media leak, Sept. 2011. https://assets.documentcloud.org/documents/2702948/Problem-Book-Redacted.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121215359/https://s3.amazonaws.com/s3.documentcloud.org/documents/2702948/Problem-Book-Redacted.pdf.

[138] Events product centre. Media leak. https://assets.documentcloud.org/documents/2432293/operational-engineering-nov-2010.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121220742/https://s3.amazonaws.com/s3.documentcloud.org/documents/2432293/operational-engineering-nov-2010.pdf.

[139] Legal authorisation flowchart: Collection. Media leak, nov 2006. https://theintercept.com/document/2015/06/22/tsi-legal-flowcharts/.

[140] Mathematics & cryptography. https://www.gchq-careers.co.uk/departments/mathematics-and-cryptography.html.

[141] What's the worst that could happen? Media leak. https://assets.documentcloud.org/documents/2699620/What-Is-the-Worst-That-Can-Happen-March-2010.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121224813/https://s3.amazonaws.com/s3.documentcloud.org/documents/2699620/What-Is-the-Worst-That-Can-Happen-March-2010.pdf.

[142] Operational legalities. Media leak. https://theintercept.com/document/2015/06/22/operational-legalities-gchq-powerpoint-presentation/.

[143] Full-spectrum cyber effects. Media leak, 2010. http://msnbcmedia.msn.com/i/msnbc/sections/news/snowden_cyber_offensive2_nbc_document.pdf. Archived Oct 14, 2017: https://web.archive.org/web/20171014124447/http://msnbcmedia.msn.com/i/msnbc/sections/news/snowden_cyber_offensive2_nbc_document.pdf.

[144] Technical capability. Media leak. https://assets.documentcloud.org/documents/2189963/cyprus-gchq.pdf. Archived Aug 3, 2015: https://web.archive.org/web/20150803201725/https://s3.amazonaws.com/s3.documentcloud.org/documents/2189963/cyprus-gchq.pdf.

[145] SIRDCC speech technology WG assessment of current STT technology. Media leak, Dec. 2009. https://assets.documentcloud.org/documents/2072016/wg-advice-to-security-service.pdf. Archived May 5, 2015: https://web.archive.org/web/20150505214651/https://s3.amazonaws.com/s3.documentcloud.org/documents/2072016/wg-advice-to-security-service.pdf.

[146] Supporting Internet operations. Media leak. https://assets.documentcloud.org/documents/2432222/200g-iris-access.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121221019/https://s3.amazonaws.com/s3.documentcloud.org/documents/2432222/200g-iris-access.pdf.

[147] media-34103. Media leak. http://www.spiegel.de/media/media-34103.pdf. Archived Sept 30, 2017: https://web.archive.org/web/20170930084837/http://www.spiegel.de/media/media-34103.pdf.

[148] News. Media leak. http://www.spiegel.de/media/media-34103.pdf. Archived Sept 30, 2017: https://web.archive.org/web/20170930084837/http://www.spiegel.de/media/media-34103.pdf.

[149] ISA-94: Application for renewal of warrant GPW/1160 in respect of activities which involve the modification of commercial software, June 2008. https://assets.documentcloud.org/documents/2106826/gchq-application-for-renewal-of-warrant-gpw-1160.pdf. Archived Jun 25, 2016: https://web.archive.org/web/20150625215229/https://s3.amazonaws.com/s3.documentcloud.org/documents/2106826/gchq-application-for-renewal-of-warrant-gpw-1160.pdf.

[150] W. Geiselmann, H. Kopfer, R. Steinwandt, and E. Tromer. Improved routing-based linear algebra for the number field sieve. In *Information Technology: Coding and Computing*, 2005.

[151] W. Geiselmann and R. Steinwandt. Non-wafer-scale sieving hardware for the NFS: Another attempt to cope with 1024-bit. In *Eurocrypt*, 2007.

[152] B. Gellman and A. Soltani. NSA infiltrates links to Yahoo, Google data centers worldwide, Snowden documents say. The Washington Post, Oct. 30, 2013. https://www.washingtonpost.com/world/national-security/nsa-infiltrates-links-to-yahoo-google-data-centers-worldwide-snowden-documents-say/2013/10/30/e51d661e-4166-11e3-8b74-d89d714ca4dd_story.html.

[153] T. Gibbons-Neff. As Russia scopes undersea cables, a shadow of the United States' Cold War past. *The Washington Post – Checkpoint Blog*, Oct. 2015. https://www.washingtonpost.com/news/checkpoint/wp/2015/10/26/as-russia-scopes-undersea-cables-a-shadow-of-the-united-states-cold-war-past/.

[154] D. Gillmor. Negotiated finite field Diffie-Hellman ephemeral parameters for TLS. IETF Internet Draft, May 2015.

[155] J. Gilmore. Cracking DES: Secrets of encryption research, wiretap politics & chip design, July 1998.

[156] D. Goodin. How "omnipotent" hackers tied to NSA hid for 14 yearsand were found at last, Feb. 2015. https://arstechnica.com/information-technology/2015/02/how-omnipotent-hackers-tied-to-the-nsa-hid-for-14-years-and-were-found-at-last/.

[157] D. Goodin. Mysterious Microsoft patch killed 0-days released by NSA-leaking Shadow Brokers, Apr. 2017. https://arstechnica.com/security/2017/04/purported-shadow-brokers-0days-were-in-fact-killed-by-mysterious-patch/.

[158] Google. Google for work: Enterprise solutions to work the way you live. https://www.google.com/work/.

[159] Detecting logos — Google cloud vision API documentation — Google cloud platform. https://cloud.google.com/vision/docs/detecting-logos.

[160] Transparency report. https://www.google.com/transparencyreport/.

[161] Gopacket. https://github.com/google/gopacket.

[162] D. M. Gordon. Designing and detecting trapdoors for discrete log cryptosystems. In *Crypto*, 1992.

[163] D. M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math.*, 6(1), 1993.

[164] A. Greenberg. Shopping for zero-days: A price list for hackers' secret software exploits. Forbes, Mar. 2012. http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/.

[165] A. Greenberg. Major leak suggests NSA was deep in Middle East banking system. Wired, Apr. 2017. https://www.wired.com/2017/04/major-leak-suggests-nsa-deep-middle-east-banking-system/.

[166] G. Greenwald. How covert agents infiltrate the Internet to manipulate, deceive, and destroy reputations. The Intercept, Feb. 2014. https://theintercept.com/2014/02/24/jtrig-manipulation/.

[167] M. R. Group. XKEYSCORE, cipher detection, and you!, Aug. 2008. https://assets.documentcloud.org/documents/2157058/xks-cipher-detection-and-you.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121221126/https://s3.amazonaws.com/s3.documentcloud.org/documents/2157058/xks-cipher-detection-and-you.pdf.

[168] M. V. Hayden. *Playing to the Edge*. Penguin Press, 2016.

[169] S. Heiberg, P. Laud, and J. Willemson. The application of i-voting for Estonian parliamentary elections of 2011. In *VOTE-ID*, pages 208–223, 2011.

[170] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Security Symposium*, Aug. 2012.

[171] C. Herley and P. van Oorschot. SoK: Science, security and the elusive goal of security as a scientific pursuit. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017.

[172] K. E. Hickman. The SSL protocol, Apr. 1995. https://tools.ietf.org/html/draft-hickman-netscape-ssl-00.

[173] J. Hoffman-Andrews. Forward secrecy at Twitter, Nov. 2013. https://blog.twitter.com/2013/forward-secrecy-at-twitter.

[174] G. Hoglund and J. Butler. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, 2005.

[175] E. Holder. Minimization procedures used by the National Security Agency in connection with acquisitions of foreign intelligence information puruant to Section 702 of the Foreign Intelligence Surveillance Act of 1978, as amended. Declassified document. https://www.dni.gov/files/documents/Minimization%20Procedures%20used%20by%20NSA%20in%20Connection%20with%20FISA%20SECT%20702.pdf.

[176] M. Holt. Caddy 0.8.3 released, Apr. 2016. https://caddyserver.com/blog/caddy-0_8_3-released.

[177] H. Hoogstraaten, R. Prins, D. Niggebrugge, D. Heppener, F. Groenewegen, J. Wettinck, K. Strooy, P. Arends, P. Pols, R. Kouprie, S. Moorrees, X. van Pelt, and Y. Z. Hu. Black Tulip report of the investigation into the DigiNotar Certificate Authority breach. Technical report, Fox-IT, Aug. 2012.

[178] HTTP Public Key Pinning (HPKP) - HTTP — MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning.

[179] ICT Export Cluster. e-estonia.com: The digital society, Aug. 2014. http://e-estonia.com/.

[180] Internet Security Research Group. Let's Encrypt certificate authority. https://letsencrypt.org/.

[181] T. Jager, K. G. Paterson, and J. Somorovsky. One bad apple: Backwards compatibility attacks on state-of-the-art cryptography. In *NDSS*, 2013.

[182] Jimdo. Website builder: Create a free website. http://www.jimdo.com/.

[183] M. Johanson. How burglars use Facebook to target vacationing homeowners. International Business Times, July 2013. http://www.ibtimes.com/how-burglars-use-facebook-target-vacationing-homeowners-1341325.

[184] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 337–348. ACM, 2013.

[185] R. Johnston. The real deal on seals: Improving tamper detection. *Security Management*, 41:93–100, Sept. 1997.

[186] R. Johnston. Some comments on choosing seals and on PSA label seals. In *Proceedings of the 7th Security Seals Symposium*, 2006. http://www.ne.anl.gov/capabilities/vat/pdfs/choosing-seals-and-using-PSA-seals-2006.pdf.

[187] R. Johnston. Insecurity of New Jersey's seal protocols for voting machines, Oct. 2010. http://www.cs.princeton.edu/~appel/voting/Johnston-AnalysisOfNJSeals.pdf.

[188] R. Johnston and A. R. Garcia. Vulnerability assessment of security seals. *Journal of Security Administration*, 20:15–27, 1997.

[189] Joint and national intelligence support to military operations, Jan. 2012. http://www.dtic.mil/doctrine/new_pubs/jp2_01.pdf.

[190] D. W. Jones and B. Simons. *Broken Ballots: Will Your Vote Count?* Stanford University Center for the Study of Language and Information, 2012.

[191] A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. *Math. Comp.*, 72(242):953–967, 2003.

[192] R. Joyce. Disrupting nation state hackers. USENIX Enigma Conference 2016, Jan. 2016. https://www.youtube.com/watch?v=bDJb8WOJYdA.

[193] E. Kain. Report: NSA intercepting laptops ordered online, installing spyware. Forbes, Dec. 2013. Accessed: May 14, 2014, http://www.forbes.com/sites/erikkain/2013/12/29/report-nsa-intercepting-laptops-ordered-online-installing-spyware/.

[194] J. Kasten. Server authentication on the past, present, and future Internet, 2015. https://deepblue.lib.umich.edu/bitstream/handle/2027.42/116632/jdkasten_1.pdf?sequence=1&isAllowed=y.

[195] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296, Oct. 2014.

[196] Kazakhtelecom JSC notifies on introduction of national security certificate from 1 January 2016. WayBack Machine, Dec. 2016. https://web.archive.org/web/20151202203337/http://telecom.kz/en/news/view/18729.

[197] T. Kean. *The 9/11 commission report: Final report of the national commission on terrorist attacks upon the United States*. Government Printing Office, 2011.

[198] S. Kent. IP Authentication Header. RFC 4302, Dec. 2005.

[199] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, Dec. 2005.

[200] J. Kirsch, H. Moltke, J. Appelbaum, L. Poitras, M. Ermert, and C. Grothoff. NSA/GCHQ: The HACIENDA program for Internet colonization. Heise online, Aug. 2014. https://www.heise.de/ct/artikel/NSA-GCHQ-The-HACIENDA-Program-for-Internet-Colonization-2292681.html?hg=1&hgi=16&hgf=false. Archived Sept 15, 2017: https://web.archive.org/web/20170915004908/https://www.heise.de/ct/artikel/NSA-GCHQ-The-HACIENDA-Program-for-Internet-Colonization-2292681.html.

[201] J. Kitcat. Source availability and e-voting: An advocate recants. *Commun. ACM*, 47(10):65–67, Oct. 2004.

[202] EPIC - Klayman v. Obama. https://epic.org/amicus/fisa/215/klayman/.

[203] T. Kleinjung. Cofactorisation strategies for the number field sieve and an estimate for the sieving step for factoring 1024 bit integers, 2006. http://www.hyperelliptic.org/tanja/SHARCS/talks06/thorsten.pdf.

[204] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann. Factorization of a 768-bit RSA modulus. In *Crypto*, 2010.

[205] T. Kleinjung, C. Diem, A. K. Lenstra, C. Priplata, and C. Stahlke. Computation of a 768-bit prime field discrete logarithm. In *EUROCRYPT*, 2017.

[206] B. Knight. Germany to pour cash into mass surveillance. Deutsche Welle, Aug. 2016. http://www.dw.com/en/germany-to-pour-cash-into-mass-surveillance/a-19537549.

[207] D. Korobov. Yandex worker stole search engine source code, tried selling for just $28k. Ars Technica, Dec. 2015. http://arstechnica.com/business/2015/12/yandex-employee-stole-search-engine-source-code-tried-to-sell-it-for-just-27000-2/.

[208] B. Krebs. Congressional report slams OPM on data breach, Sept. 2016. https://krebsonsecurity.com/2016/09/congressional-report-slams-opm-on-data-breach/.

[209] A. Langley. How to botch TLS forward secrecy, June 2013. https://www.imperialviolet.org/2013/06/27/botchingpfs.html.

[210] A. Langley, N. Modadugu, and B. Moeller. Transport layer security (TLS) false start. IETF Internet Draft, 2010.

[211] B. Laxton, K. Wang, and S. Savage. Reconsidering physical key secrecy: Teleduplication via optical decoding. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, pages 469–478, 2008.

[212] H. K. Lee, T. Malkin, and E. Nahum. Cryptographic strength of SSL/TLS servers: current and recent practices. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 83–92. ACM, 2007.

[213] A. K. Lenstra and H. W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*. Springer, 1993.

[214] C++ packet sniffing and crafting library. https://libtins.github.io/.

[215] N. Lidzborsky. Staying at the forefront of email security and reliability: HTTPS-only and 99.978 https://gmail.googleblog.com/2014/03/staying-at-forefront-of-email-security.html.

[216] Z. Lin. TLS session resumption: Full-speed and secure, Feb. 2015. https://blog.cloudflare.com/tls-session-resumption-full-speed-and-secure/.

[217] M. Lindeman and P. B. Stark. A gentle introduction to risk-limiting audits. *IEEE Security & Privacy*, 10(5):42–49, 2012.

[218] M. Lipacis. Semiconductors: Moore stress = structural industry shift. Technical report, Jefferies, Sept. 2012. https://javatar.bluematrix.com/docs/pdf/10087cbf-e43a-4a33-8aa0-05ec885ac04c.pdf.

[219] H. Lipmaa. Paper-voted (and why I did so). Blog post, Mar. 2011. http://helger.wordpress.com/2011/03/05/paper-voted-and-why-i-did-so/.

[220] H. Lipmaa. A simple cast-as-intended e-voting protocol by using secure smart cards, May 2014. http://eprint.iacr.org/2014/348.

[221] J. Liu. So what does the USA Freedom Act do anyway?, June 2015. https://www.lawfareblog.com/so-what-does-usa-freedom-act-do-anyway.

[222] I. Lovecruft. Twitter, Dec. 2015. https://twitter.com/isislovecruft/status/681590393385914368.

[223] Mail-in-a-Box. https://mailinabox.email/.

[224] M. Mak. Trusted defense microelectronics: Future access and capabilities are uncertain, Oct. 2015. http://www.gao.gov/assets/680/673401.pdf.

[225] Mandiant. APT1: Exposing one of China's cyber espionage units, Feb. 2013. http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf.

[226] M. Marquis-Boire, G. Greenwald, and M. Lee. XKEYSCORE: NSA's Google for the world's private communications. The Intercept, July 2015. https://theintercept.com/2015/07/01/nsas-google-worlds-private-communications/.

[227] U. M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Crypto*, 1994.

[228] U. M. Maurer and S. Wolf. Diffie-Hellman oracles. In *Crypto*, 1996.

[229] N. Mavrogiannopoulos, F. Vercauteren, V. Velichkov, and B. Preneel. A cross-protocol attack on the TLS protocol. In *ACM CCS*, pages 62–72, 2012.

[230] M. Mayer. Our commitment to protecting your information, Nov. 2013. https://yahoo.tumblr.com/post/67373852814/our-commitment-to-protecting-your-information.

[231] J. McLaughlin. Spy chief complains that Edward Snowden sped up spread of encryption by 7 years, Apr. 2016. https://theintercept.com/2016/04/25/spy-chief-complains-that-edward-snowden-sped-up-spread-of-encryption-by-7-years/.

[232] C. Meadows. Analysis of the Internet key exchange protocol using the NRL protocol analyzer. In *IEEE Symposium on Security and Privacy*, 1999.

[233] N. Mediati. How to remotely install apps on your smartphone. TechHive, Nov. 2013. http://www.techhive.com/article/2067005/how-to-remotely-install-apps-on-your-smartphone.html.

[234] D. Medine, R. Brand, E. C. Cook, J. Dempsey, and P. Wald. Report on the surveillance program operated pursuant to Section 702 of the Foreign Intelligence Surveillance Act. Privacy and Civil Liberties Oversight Board, July 2014.

[235] D. Medine, R. Brand, E. C. Cook, J. Dempsey, and P. Wald. Report on the Telephone Records Program conducted under Section 215 of the USA PATRIOT Act and on the operations of the Foreign Intelligence Surveillance Court. Privacy and Civil Liberties Oversight Board, Jan. 2014.

[236] P. Membrey, D. Hows, and E. Plugge. SSL load balancing. In *Practical Load Balancing*, pages 175–192. Springer, 2012.

[237] J. Menn. Exclusive: Secret contract tied NSA and security industry pioneer. Reuters, Dec. 2013. http://www.reuters.com/article/us-usa-security-rsa/exclusive-secret-contract-tied-nsa-and-security-industry-pioneer-idUSBRE9BJ1C220131220.

[238] J. Mickens. This world of ours. USENIX ;login: logout, Jan. 2014. https://www.usenix.org/system/files/1401_08-12_mickens.pdf.

[239] Microsoft. TLS/SSL settings, Nov. 2015. https://technet.microsoft.com/en-us/library/dn786418.aspx.

[240] Microsoft Security Bulletin MS15-055. Vulnerability in Schannel could allow information disclosure, May 2015.

[241] CSR reports hub. https://www.microsoft.com/en-us/about/corporate-responsibility/reports-hub.

[242] mod_ssl: Apache HTTP server version 2.4. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html.

[243] Module ngx_http_ssl_module. http://nginx.org/en/docs/http/ngx_http_ssl_module.html.

[244] Mozilla Telemetry. https://telemetry.mozilla.org/.

[245] Fireeye contracts with NSA, may 2017. https://www.muckrock.com/foi/united-states-of-america-10/fireeye-contracts-with-nsa-18358/#file-133635.

[246] E. Nakashima. The NSA has linked the WannaCry computer worm to north korea. The Washington Post, June 2017. https://www.washingtonpost.com/world/national-security/the-nsa-has-linked-the-wannacry-computer-worm-to-north-korea/2017/06/14/101395a2-508e-11e7-be25-3a519335381c_story.html.

[247] P. W. Nan Tian, Aude Fleurant and S. Wezeman. Trends in world military expenditure, 2016. Stockholm International Peace Research Institute, Apr. 2017. https://www.sipri.org/sites/default/files/Trends-world-military-expenditure-2016.pdf.

[248] M. Nemec, M. Sys, P. Svenda, D. Klinec, and V. Matyas. The return of Coppersmith's attack: Practical factorization of widely used RSA moduli. In *24nd ACM Conference on Computer and Communications Security*, Oct. 2017.

[249] NetMiner - social network analysis software. http://www.netminer.com/main/main-read.do.

[250] Nsa doc-1.pdf. Media Leak. https://firstlook.org/wp-uploads/sites/1/2014/09/NSA-Doc-1.pdf.

[251] L. H. newman. The attack on global privacy leaves few places to turn. Wired, Aug. 2017. https://www.wired.com/story/china-russia-vpn-crackdown/.

[252] NIST. FIPS PUB 186-4: Digital signature standard, 2013.

[253] Documents from no place to hide. Media leak. http://us.macmillan.com/static/holt/greenwald/NoPlaceToHide-Documents-Compressed.pdf. Archived Sept 29, 2017: https://web.archive.org/web/20170929153922/https://us.macmillan.com/static/holt/greenwald/NoPlaceToHide-Documents-Compressed.pdf.

[254] FY 2013 congressional budget justification. Media leak, Feb. 2012. http://cryptome.org/2013/08/spy-budget-fy13.pdf. Archived Dec 8, 2017: https://web.archive.org/web/20171208163819/http://cryptome.org/2013/08/spy-budget-fy13.pdf.

[255] FY 2013 congressional budget justification (Osama bin Laden). Media leak, Feb. 2012. https://assets.documentcloud.org/documents/2082989/fy13-black-budget-on-bin-laden-raid.pdf. Archived May 18, 2015: https://web.archive.org/web/20150518192659/https://s3.amazonaws.com/s3.documentcloud.org/documents/2082989/fy13-black-budget-on-bin-laden-raid.pdf.

[256] NIOC Maryland advanced computer network operations course. Media leak, Apr. 2012. http://www.spiegel.de/media/media-35657.pdf. Archived Oct 19, 2016: https://web.archive.org/web/20161019202649/http://www.spiegel.de/media/media-35657.pdf.

[257] Advanced HTTP activity analysis. Media leak. https://theintercept.com/document/2015/07/01/advanced-http-activity-analysis-2/.

[258] APEX active/passive exfiltration. Media leak, Aug. 2009. http://www.spiegel.de/media/media-35671.pdf. Archived Sept 29, 2017: https://web.archive.org/web/20170929164529/http://www.spiegel.de/media/media-35671.pdf.

[259] VPN and VOIP exploitation with HAMMERCHANT and HAMMERSTEIN. Media leak. https://assets.documentcloud.org/documents/1076868/vpn-and-voip-exploitation-with-hammerchant-and.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121220058/https://s3.amazonaws.com/s3.documentcloud.org/documents/1076868/vpn-and-voip-exploitation-with-hammerchant-and.pdf.

[260] German, NSA SIGINTers share DNI processing knowledge. Media leak. http://www.spiegel.de/media/media-34087.pdf. Archived Dec 16, 2016: https://web.archive.org/web/20161216190518/http://www.spiegel.de/media/media-34087.pdf.

[261] NSA intelligence relationship with Germany - Bundesnachrichtendienst (BND). Media leak, Jan. 2013. http://www.spiegel.de/media/media-34053.pdf. Archived Dec 11, 2016: https://web.archive.org/web/20161211174124/http://www.spiegel.de/media/media-34053.pdf.

[262] Classification guide for SIGINT material dating from 16 August 1945 - 31 December 1967. Media leak, Apr. 2012. http://www.spiegel.de/media/media-34093.pdf. Archived Nov 24, 2016: https://web.archive.org/web/20161124080323/http://www.spiegel.de/media/media-34093.pdf.

[263] NSA and the supercomputer industry, 2012. https://www.nsa.gov/news-features/declassified-documents/cryptologic-quarterly/assets/files/NSA_and_the_Supercomputer.pdf.

[264] DEEPDIVE configuration read me. Media leak. https://assets.documentcloud.org/documents/2116009/deepdive-readme.pdf. Archived Jul 2, 2015: https://web.archive.org/web/20150702025240/https://s3.amazonaws.com/s3.documentcloud.org/documents/2116009/deepdive-readme.pdf.

[265] Fielded capability: End-to-end VPN SPIN 9 design review. Media leak. http://www.spiegel.de/media/media-35529.pdf. Archived Mar 31, 2017: https://web.archive.org/web/20170331033516/http://www.spiegel.de/media/media-35529.pdf.

[266] GALLANTWAVE@scale. Media leak. http://www.spiegel.de/media/media-35514.pdf. Archived Oct 19, 2016: https://web.archive.org/web/20161019184752/http://www.spiegel.de:80/media/media-35514.pdf.

[267] Huawei powerpoint slides. Media leak. https://assets.documentcloud.org/documents/1094880/huawei-powerpoint-slides.pdf. Archived Dec 11, 2016: https://web.archive.org/web/20161211092251/https://s3.amazonaws.com/s3.documentcloud.org/documents/1094880/huawei-powerpoint-slides.pdf.

[268] How is Human Language Technology (HLT) progressing? Media leak, Sept. 2011. https://www.eff.org/files/2015/05/26/20150505-intercept-sidtoday-is-hlt-progressing-final.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121220631/https://www.eff.org/files/2015/05/26/20150505-intercept-sidtoday-is-hlt-progressing-final.pdf.

[269] I hunt admins that use telnet (part 3). Media leak, Dec. 2012. https://assets.documentcloud.org/documents/1094387/i-hunt-sys-admins.pdf. Archived Feb 13, 2017: https://web.archive.org/web/20170213215614/https://s3.amazonaws.com/s3.documentcloud.org/documents/1094387/i-hunt-sys-admins.pdf.

[270] Industrial-scale exploitation. Media leak. https://assets.documentcloud.org/documents/1077724/industry-scale-exploitation.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121220430/https://s3.amazonaws.com/s3.documentcloud.org/documents/1077724/industry-scale-exploitation.pdf.

[271] Memorandum of understanding (MOU) between the National Security Agency/Central Security Service (NSA/CSS) and the Israeli SIGINT National Unit (ISNU) pertaining to the protection of U.S. persons. Media leak. http://s3.documentcloud.org/documents/785495/doc1.pdf. Archived Mar 19, 2017: https://web.archive.org/web/20170319071609/https://s3.amazonaws.com/s3.documentcloud.org/documents/785495/doc1.pdf.

[272] LONGHAUL – WikiInfo. Media leak. http://www.spiegel.de/media/media-35533.pdf. Archived Apr 7, 2017: https://web.archive.org/web/20170407201624/http://www.spiegel.de/media/media-35533.pdf.

[273] Bad guys are everywhere, good guys are somewhere! Media leak. https://s3.amazonaws.com/s3.documentcloud.org/documents/1301057/tm-m-402.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121214748/https://s3.amazonaws.com/s3.documentcloud.org/documents/1301057/tm-m-402.pdf.

[274] POISONNUT – WikiInfo. Media leak. http://www.spiegel.de/media/media-35519.pdf. Archived Mar 31, 2017: https://web.archive.org/web/20170331033423/http://www.spiegel.de/media/media-35519.pdf.

[275] An easy win: Using SIGINT to learn about new viruses. Media leak. https://assets.documentcloud.org/documents/2106783/project-camberdada.pdf. Archived Jun 26, 2015: https://web.archive.org/web/20150626020220/https://s3.amazonaws.com/s3.documentcloud.org/documents/2106783/project-camberdada.pdf.

[276] Technology to help transcribers. Media leak, Oct. 2003. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2003/2003-10-14_SIDToday_-_Technology_to_Help_Transcribers.pdf.

[277] FAIRVIEW and STORMBREW: 'live' - on the net. Media leak, Nov. 2003. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2003/2003-11-19_SIDToday_-_FAIRVIEW_and_STORMBREW_Live_-_On_the_Net.pdf.

[278] Getting leads: GEO mining. Media leak, Aug. 2004. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2004/2004-08-17_SIDToday_-_Getting_Leads_GEO_Mining.pdf.

[279] Machines translating Arabic. Media leak, Jan. 2004. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2004/2004-01-21_SIDToday_-_Machines_Translating_Arabic.pdf.

[280] Shift to software demodulation in Misawa expands collection, saves money. Media leak, Mar. 2009. https://assets.documentcloud.org/

documents/3675589/SIDToday-Shift-to-Software-Demodulation-in.pdf. Archieved Jan 21, 2018: https://web.archive.org/web/20180121232149/https://s3.amazonaws.com/s3.documentcloud.org/documents/3675589/SIDToday-Shift-to-Software-Demodulation-in.pdf.

[281] NSA's oldest third party SIGINT partnership. Media leak, Oct. 2005. https://assets.documentcloud.org/documents/1282654/nsa-tr-607.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121225035/https://s3.amazonaws.com/s3.documentcloud.org/documents/1282654/nsa-tr-607.pdf.

[282] Is there a sustainable ops tempo in S2? how can analysts deal with the flood of collection? – an interview with [redacted] (conclusion). Media leak, Apr. 2011. https://assets.documentcloud.org/documents/2089125/analytic-modernization.pdf. Archived May 28, 2015: https://web.archive.org/web/20150528214825/https://s3.amazonaws.com/s3.documentcloud.org/documents/2089125/analytic-modernization.pdf.

[283] Silent success: SIGINT synergy helps shape US forign policy. Media leak, Aug. 2010. https://www.aclu.org/files/natsec/nsa/20140722/Silent%20Success%20SIGINT%20Synergy%20Helps%20Shape%20US%20Foreign%20Policy.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121224923/https://www.aclu.org/files/natsec/nsa/20140722/Silent%20Success%20SIGINT%20Synergy%20Helps%20Shape%20US%20Foreign%20Policy.pdf.

[284] Targeting terrorist Internet traffic. Media leak, Mar. 2004. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2004/2004-03-30_SIDToday_-_Targeting_Terrorist_Internet_Traffic.pdf.

[285] The rewards of metadata. Media leak, Jan. 2004. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2004/2004-01-23_SIDToday_-_The_Rewards_of_Metadata.pdf.

[286] A visit to the USUN. Media leak, Jan. 2004. https://github.com/firstlookmedia/sidtoday/blob/master/documents/2004/2004-01-09_SIDToday_-_A_Visit_to_the_USUN.pdf.

[287] What your mother never told you about SIGDEV analysis. Media leak. http://www.spiegel.de/media/media-35551.pdf. Archived Mar 30, 2017: https://web.archive.org/web/20170330044927/http://www.spiegel.de/media/media-35551.pdf.

[288] Making things measurable: Technology trending challenges and approaches. Media leak, June 2012. http://www.spiegel.de/media/media-35535.pdf. Archived Jun 22, 2017: https://web.archive.org/web/20170622025513/http://www.spiegel.de:80/media/media-35535.pdf.

[289] Secret documents reveal N.S.A. campaign against encryption. Media leak. http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html. Archived Jan 3, 2018:

https://web.archive.org/web/20180103181552/http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html.

[290] SIGINT mission strategic plan fy2008 - 2013. Media leak, Oct. 2007. https://www.eff.org/files/2013/11/15/20131104-nyt-sigint_strategic_plan.pdf. Archived Jan 3, 2015: https://web.archive.org/web/20150103031255/https://www.eff.org/files/2013/11/15/20131104-nyt-sigint_strategic_plan.pdf.

[291] The SIGINT philosopher: Cognitive overflow? Media leak, Apr. 2011. https://assets.documentcloud.org/documents/2088972/cognitive-overflow.pdf. Archived May 28, 2015: https://web.archive.org/web/20150528203837/https://s3.amazonaws.com/s3.documentcloud.org/documents/2088972/cognitive-overflow.pdf.

[292] SIGINT strategy. Media leak. http://www.nytimes.com/interactive/2013/11/23/us/politics/23nsa-sigint-strategy-document.html. Archived Jan 21, 2018: https://web.archive.org/save/http://www.nytimes.com/interactive/2013/11/23/us/politics/23nsa-sigint-strategy-document.html?mtrref=undefined&mtrref=undefined&mtrref=undefined&mtrref=undefined&mtrref=undefined&mtrref=undefined.

[293] SKYNET: Applying advanced cloud-based behavior analytics. Media leak. https://assets.documentcloud.org/documents/2074625/skynet-applying-advanced-cloud-based-behavior.pdf. Archived May 9, 2015: https://web.archive.org/web/20150509131943/https://s3.amazonaws.com/s3.documentcloud.org/documents/2074625/skynet-applying-advanced-cloud-based-behavior.pdf.

[294] SKYNET: Courier detection via machine learning. Media leak, June 2012. https://assets.documentcloud.org/documents/2074620/skynet-courier-detection-via-machine-learning.pdf. Archived May 8, 2015: https://web.archive.org/web/20150508110226/https://s3.amazonaws.com/s3.documentcloud.org/documents/2074620/skynet-courier-detection-via-machine-learning.pdf.

[295] SPIN 15 VPN story. Media leak. http://www.spiegel.de/media/media-35522.pdf. Archived Mar 30, 2017: https://web.archive.org/web/20170330212419/http://www.spiegel.de/media/media-35522.pdf.

[296] United States SIGINT system January 2007 strategic mission list. Media leak. https://cryptome.org/2014/09/nsa-strategic-mission-list.pdf. Archived Dec 6, 2017: https://web.archive.org/web/20171206143902/https://cryptome.org/2014/09/nsa-strategic-mission-list.pdf.

[297] NSA Suite B cryptography, Dec. 2012. https://web.archive.org/web/20130201165547/http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

[298] Groundbreaking ceremony held for computing center, May 2013. https://www.nsa.gov/news-features/press-room/press-releases/2013/computing-center.shtml.

[299] Stealthy techniques can crack some of SIGINT's hardest targets. Media Leak. http://www.spiegel.de/media/media-35669.pdf. Archived Nov 24, 2017: https://web.archive.org/web/20171124115053/http://www.spiegel.de/media/media-35669.pdf.

[300] Targeting rationale (TAR). Media leak. https://assets.documentcloud.org/documents/750577/target-analyst-rationale-instructions-final.pdf. Archived Jan 21, 2018: https://web.archive.org/web/20180121225923/https://s3.amazonaws.com/s3.documentcloud.org/documents/750577/target-analyst-rationale-instructions-final.pdf.

[301] TURMOIL IPsec VPN sessionization. Media leak, Aug. 2009. http://www.spiegel.de/media/media-35528.pdf. Archived Mar 31, 2017: https://web.archive.org/web/20170331100513/http://www.spiegel.de/media/media-35528.pdf.

[302] TURMOIL VPN processing. Media leak, Oct. 2009. http://www.spiegel.de/media/media-35526.pdf. Archived Sept 30, 2017: https://web.archive.org/web/20170930091943/http://www.spiegel.de/media/media-35526.pdf.

[303] TURMOIL/APEX/APEX high level description document. Media leak. http://www.spiegel.de/media/media-35513.pdf. Archived Mar 29, 2017: https://web.archive.org/web/20170329235112/http://www.spiegel.de/media/media-35513.pdf.

[304] The unofficial XKEYSCORE user guide. Media leak. https://assets.documentcloud.org/documents/2116191/unofficial-xks-user-guide.pdf. Archived Jul 2, 2015: https://web.archive.org/web/20150702230507/https://s3.amazonaws.com/s3.documentcloud.org/documents/2116191/unofficial-xks-user-guide.pdf.

[305] VALIANTSURF – WikiInfo. Media leak. http://www.spiegel.de/media/media-35527.pdf. Archived Mar 31, 2017: https://web.archive.org/web/20170331091428/http://www.spiegel.de/media/media-35527.pdf.

[306] VALIANTSURF (VS): Capability levels. Media leak. http://www.spiegel.de/media/media-35517.pdf. Archived Mar 31, 2017: https://web.archive.org/web/20170331093002/http://www.spiegel.de/media/media-35517.pdf.

[307] VoIP configuration and forwarding read me. Media leak. https://assets.documentcloud.org/documents/2116266/voip-readme.pdf. Archived July 3, 2015: https://web.archive.org/web/20150703021656/https://s3.amazonaws.com/s3.documentcloud.org/documents/2116266/voip-readme.pdf.

[308] Intro to the VPN exploitation process. Media leak, Sept. 2010. http://www.spiegel.de/media/media-35515.pdf. Archived Aug 29, 2017: https://web.archive.org/web/20170829162736/http://www.spiegel.de/media/media-35515.pdf.

[309] VPN SigDev basics. Media leak. http://www.spiegel.de/media/media-35520.pdf. Archived Apr 1, 2017: https://web.archive.org/web/20170401102232/http://www.spiegel.de/media/media-35520.pdf.

[310] Document excerpt on the sharing of the NSA spy tool XKeyscore with the Federal Office for the Protection of the Constitution (BfV), Germanys domestic intelligence agency. Media leak. http://www.spiegel.de/media/media-34045.pdf. Archived Nov 24, 2016: https://web.archive.org/web/20161124044811/http://www.spiegel.de/media/media-34045.pdf.

[311] XKEYSCORE finding and querying on document metadata, Apr. 2009. https://assets.documentcloud.org/documents/2157080/finding-and-querying-document-metadata.pdf. Archived Jul 2, 2015: https://web.archive.org/web/20150702032823/https://s3.amazonaws.com/s3.documentcloud.org/documents/2157080/finding-and-querying-document-metadata.pdf.

[312] Oak Ridge National Laboratory. Introducing Titan, 2012. https://www.olcf.ornl.gov/titan.

[313] Statistical transparency report regarding the use of national security authorities for calendar year 2016. IC on the Record, May 2017. https://icontherecord.tumblr.com/transparency/odni_transparencyreport_cy2016.

[314] Report on the President's Surveillance Program, July 2009. https://assets.documentcloud.org/documents/2427921/savage-nyt-foia-stellarwind-ig-report.pdf.

[315] Report on the President's Surveillance Program – NSA Inspector General – working draft. Media leak, Mar. 2009. https://www.aclu.org/files/natsec/nsa/20130816/NSA%20IG%20Report.pdf. Archived: Dec 29, 2017: https://web.archive.org/web/20171229024857/https://www.aclu.org/files/natsec/nsa/20130816/NSA%20IG%20Report.pdf.

[316] U. Oja. Paavo Pihelgas: Elektroonilise hääletamise vaatlemine on lihtsalt võimatu, Mar. 2011. In Estonian. http://forte.delfi.ee/news/digi/paavo-pihelgas-elektroonilise-haaletamise-vaatlemine-on-lihtsalt-voimatu.d?id=41933409.

[317] A. Olofsson. An introduction to semiconductor economics, Oct. 2014. http://www.adapteva.com/andreas-blog/semiconductor-economics-101/.

[318] K. Opsahl. Deputy Attorney General Rosensteins responsible encryption demand is bad and he should feel bad, Oct. 2017. https://www.eff.org/deeplinks/2017/10/deputy-attorney-general-rosensteins-responsible-encryption-demand-bad-and-he.

[319] H. Orman. The OAKLEY Key Determination Protocol. RFC 2412, Nov. 1998.

[320] OSINT fusion project. Media leak. https://assets.documentcloud.org/documents/2153970/osint-fusion-project.pdf. Archived Jul 2, 2015: https://web.archive.org/web/20150702222336/https://s3.amazonaws.com/s3.documentcloud.org/documents/2153970/osint-fusion-project.pdf.

[321] M. Ossmann. Hackrf. https://github.com/mossmann/hackrf.

[322] ownCloud - the last cloud collaboration platform you'll ever need. https://owncloud.org/.

[323] N. Ozer. Police use of social media surveillance software is escalating, and activists are in the digital crosshairs, Sept. 2016. https://www.aclu.org/blog/privacy-technology/surveillance-technologies/police-use-social-media-surveillance-software.

[324] F. Paget. Hacking summit names nations with cyberwarfare capabilities. McAfee Blog Central, Oct. 2013. http://blogs.mcafee.com/mcafee-labs/hacking-summit-names-nations-with-cyberwarfare-capabilities.

[325] The cyber-crime black market uncovered. Technical report, Panda Security, 2010. http://www.pandasecurity.com/mediacenter/src/uploads/2014/07/The-Cyber-Crime-Black-Market.pdf.

[326] Papilio FPGA platform. http://papilio.cc/.

[327] A. Parsovs. Practical issues with TLS client certificate authentication, Feb. 2014. https://www.internetsociety.org/sites/default/files/12_4_1.pdf.

[328] PCS harvesting at scale. Media leak. https://theintercept.com/document/2015/02/19/pcs-harvesting-scale/.

[329] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson. Global measurement of dns manipulation. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, 2017.

[330] J. Penney. Chilling effects: Online surveillance and Wikipedia use. *Berkeley Technology Law Journal*, 31.1, 2016.

[331] N. Perlroth. All 3 billion Yahoo accounts were affected by 2013 attack. The New York Times, Oct. 2017. https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html.

[332] N. Perlroth, J. Larson, and S. Shane. N.S.A. able to foil basic safeguards of privacy on web. The New York Times, Sept. 2013. http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html?pagewanted=2&_r=1.

[333] B. Plumer. Estonia gets to vote online. Why can't America? Wonkblog. The Washington Post, Nov. 2012. http://www.washingtonpost.com/blogs/wonkblog/wp/2012/11/06/estonians-get-to-vote-online-why-cant-america/.

[334] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (corresp.). *Trans. Inform. Theory*, 24(1), 1978.

[335] J. M. Pollard. A Monte Carlo method for factorization. *BIT Numerical Mathematics*, 15(3):331–334, 1975.

[336] K. Poulsen. Snowden's email provider loses appeal over encryption keys. Wired, Apr. 2014. https://www.wired.com/2014/04/lavabit-ruling/.

[337] A. Press. New nuclear sub is said to have special eavesdropping ability. *The New York Times*, Feb. 2005. http://www.nytimes.com/2005/02/20/politics/new-nuclear-sub-is-said-to-have-special-eavesdropping-ability.html.

[338] ptrace(2): process trace. Linux Programmer's Manual.

[339] QUIC, a multiplexed stream transport over UDP. https://www.chromium.org/quic.

[340] T. Raidma and J. Kase. Kohaliku omavalitsuse volikogu valimiste e-hääletamise protseduuride hindamise lõpparuanne, Jan. 2014. In Estonian. http://vvk.ee/public/KOV13/lopparuanne_2013.ddoc.

[341] E. Rescorla. The Transport Layer Security (TLS) protocol version 1.3 draft-ietf-tls-tls13-15, Aug. 2016. https://tools.ietf.org/html/draft-ietf-tls-tls13-15.

[342] E. Rescorla and T. Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Aug. 2008.

[343] I. Ristic. Twitter, Apr. 2014. https://twitter.com/ivanristic/status/453280081897467905.

[344] I. Ristic. SSL/TLS deployment best practices, Dec. 2014. https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices.pdf.

[345] J. Robertson and M. Riley. Kaspersky Lab has been working with Russian intelligence. Bloomberg Businessweek, July 2017. https://www.bloomberg.com/news/articles/2017-07-11/kaspersky-lab-has-been-working-with-russian-intelligence.

[346] D. G. Robinson and J. A. Halderman. Ethical issues in e-voting security analysis. In *Proceedings of the 2nd Workshop on Ethics in Computer Security Research (WECSR)*, March 2011.

[347] A. Roth. Putin signs new anti-terror law in Russia. Edward Snowden is upset. *The Washington Post*, July 2016. https://www.washingtonpost.com/world/europe/putin-signs-law-to-bolster-russian-surveillance-angering-edward-snowden/2016/07/07/4d307aca-443e-11e6-a76d-3550dba926ac_story.html.

[348] rsyslog: The rocket-fast system for log processing, Apr. 2014. http://www.rsyslog.com/.

[349] SDR (Software Defined Radio). http://osmocom.org/projects/sdr/wiki/rtl-sdr.

[350] D. Rushe. Yahoo $250,000 daily fine over NSA data refusal was set to double 'every week'. *The Guardian*, Sept. 2014. https://www.theguardian.com/world/2014/sep/11/yahoo-nsa-lawsuit-documents-fine-user-data-refusal.

[351] P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. Prêt à voter: A voter-verifiable voting system. *Trans. Info. For. Sec.*, 4(4):662–673, Dec. 2009.

[352] J. A. Salowey, H. Zhou, H. Tschofenig, and P. Eronen. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 4507, May 2006.

[353] D. Sanger and N. Perlroth. U.S. said to find North Korea ordered cyberattack on Sony. The New York Times, Dec. 2014. https://www.nytimes.com/2014/12/18/world/asia/us-links-north-korea-to-sony-hacking.html.

[354] D. E. Sanger. Obama order sped up wave of cyberattacks against Iran. The New York Times, June 2012. http://www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html.

[355] D. Savignac. SID and RAD to develop lexicons in five languages. Media leak, Apr. 2004. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2004/2004-04-09_SIDToday_-_SID_and_RAD_to_Develop_Lexicons_in_Five_Languages.pdf.

[356] J. Schahill and J. Begley. The great SIM heist. The Intercept, Feb. 19, 2015. https://theintercept.com/2015/02/19/great-sim-heist/.

[357] O. Schirokauer. Virtual logarithms. *J. Algorithms*, 57(2):140–147, 2005.

[358] B. Schneier. Computer crime hype. Schneier on Security, Dec. 2005. https://www.schneier.com/blog/archives/2005/12/computer_crime_1.html.

[359] B. Schneier. Attacking Tor: how the NSA targets users' online anonymity. The Guardian, Oct. 2013. https://www.theguardian.com/world/2013/oct/04/tor-attacks-nsa-users-online-anonymity.

[360] B. Schneier. The NSA is not made of magic, May 2014. https://www.schneier.com/blog/archives/2014/05/the_nsa_is_not_.html.

[361] B. Schneier. Attack attribution and cyber conflict. Schneier on Security, Mar. 2015. https://www.schneier.com/blog/archives/2015/03/attack_attribut_1.html.

[362] A. Selyukh and M. Nayak. U.S. phone companies escape new data storage mandate in surveillance bill. *Reuters*, June 2015. http://www.reuters.com/article/us-usa-security-surveillance-telecommuni-idUSKBN0OI2TR20150602.

[363] I. A. Semaev. Special prime numbers and discrete logs in finite prime fields. *Math. Comp.*, 71(237):363–377, 2002.

[364] P. Sensburg, N. Warken, and C. Flisek. Beschlussfassung und bericht des 1. untersuchungsausschusses nach artikel 44 des grundgesetzes, June 2017. https://cdn.netzpolitik.org/wp-upload/2017/06/1812850-ungeschwaerzt.pdf.

[365] N. Shachtman. The secret history of Iraq's invisible war. Wired, June 2011. https://www.wired.com/2011/06/iraqs-invisible-war/.

[366] A. Shamir. *Factoring Large Numbers with the TWINKLE Device*, pages 2–12. 1999.

[367] A. Shamir and E. Tromer. Factoring large numbers with the TWIRL device. In *Crypto*, volume 3, pages 1–26. Springer, 2003.

[368] D. Shanks. Class number, a theory of factorization, and genera. In *Proc. Sympos. Pure Math.*, volume 20. 1971.

[369] Shodan. https://www.shodan.io/.

[370] Submarine cable monitoring system (SCL-SCMS). http://www.shoghicom.com/submarine-cable-monitoring-system.php.

[371] CANYONDUST goes global. Media leak, Sept. 2003. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2003/2003-09-25_SIDToday_-_CANYONDUST_Goes_Global.pdf.

[372] How SIGINT's done down under. Media leak, Aug. 2003. https://github.com/firstlookmedia/sidtoday/raw/master/documents/2003/2003-08-28_SIDToday_-_How_SIGINTs_Done_Down_Under.pdf.

[373] Open whisper systems. https://whispersystems.org/.

[374] Grand jury subpoena for Signal user data, eastern district of Virginia, Oct. 2016. https://signal.org/bigbrother/eastern-virginia-grand-jury/.

[375] R. Silverman. An analysis of Shamirs factoring device. RSA Laboratories' Bulletin, May 1999. http://www.networkdls.com/articles/bulletn12.pdf.

[376] B. Simons. Report on the Estonian Internet voting system. Verified Voting Blog, Sept. 2011. https://www.verifiedvoting.org/report-on-the-estonian-internet-voting-system-2/.

[377] R. Singel. AT&T 'spy room' documents unsealed; you've already seen them. Wired, June 2007. https://www.wired.com/2007/06/at-youve-already-seen-them/.

[378] P. W. Singer and A. Friedman. *Cybersecurity and Cyberwar: What Everyone Needs to Know*. Oxford University Press, 2014.

[379] D. Smith. Exclusive: Inside the NSA's private cloud. Network World, Sept. 2014. http://www.networkworld.com/article/2687084/security0/exclusive-inside-the-nsa-s-private-cloud.html.

[380] C. Soghoian, Jan. 2015. https://twitter.com/csoghoian/status/556573239028113408.

[381] C. Soghoian and S. Stamm. *Certified lies: Detecting and defeating government interception attacks against SSL*, pages 250–259. Springer Berlin Heidelberg, 2012.

[382] SOMALGET memo. Media leak. https://assets.documentcloud.org/documents/1164088/somalget.pdf.

[383] Spiegel Staff. Prying eyes: Inside the NSA's war on Internet security. Der Spiegel, Dec 2014. http://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html.

[384] D. Springall, Z. Durumeric, and J. A. Halderman. FTP: The forgotten cloud. In *Proceedings of the 46th IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2016.

[385] D. Springall, Z. Durumeric, and J. A. Halderman. Measuring the security harm of TLS crypto shortcuts. In *Proceedings of the 2016 ACM on Internet Measurement Conference*, Nov. 2016.

[386] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman. Security analysis of the Estonian Internet voting system. In *Proceedings of the 21st ACM Conference on Computer and Communications Security*, Nov. 2014.

[387] List of available trusted root certificates in macOS Sierra, Feb. 2017. https://support.apple.com/en-us/HT207189.

[388] Microsoft trusted root certificate program: Participants (as of June 27, 2017), jun 2017. https://gallery.technet.microsoft.com/Trusted-Root-Certificate-123665ca.

[389] Mozilla included CA certificate list, June 2017. https://ccadb-public.secure.force.com/mozilla/IncludedCACertificateReport.

[390] Special source operations corporate partner access. Media leak. https://theintercept.com/document/2016/11/16/special-source-operations-corporate-overview/.

[391] P. B. Stark. Super-simple simultaneous single-ballot risk-limiting audits. In *Proceedings of the USENIX Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)*, Aug. 2010.

[392] W. Stein et al. *Sage Mathematics Software (Version 6.5)*. The Sage Development Team, 2015. http://www.sagemath.org.

[393] B. Sterling. Operation shady RAT. Wired, Aug. 2011. https://www.wired.com/2011/08/operation-shady-rat/.

[394] M. Stokes, J. Lin, and L. R. Hsiao. The Chinese People's Liberation Army signals intelligence and cyber reconnaissance infrastructure, Nov. 2011. https://project2049.net/documents/pla_third_department_sigint_cyber_stokes_lin_hsiao.pdf.

[395] stud: The scalable TLS unwrapping daemon, 2012. https://github.com/bumptech/stud/blob/19a7f19686bcdbd689c6fbea31f68a276e62d886/stud.c#L593.

[396] N. Sullivan. The results of the CloudFlare challenge, Apr. 2014. https://blog.cloudflare.com/the-results-of-the-cloudflare-challenge/.

[397] TallyHoGazehound, Jan. 2015. https://twitter.com/BorzoiBystander/status/556571934247624704.

[398] Discovery SIGINT targeting scenarios and compliance. Media leak, Mar. 2013. https://assets.documentcloud.org/documents/1019062/discovery-sigint-targeting-scenarios-and.pdf.

[399] T. Taubert. Botching forward secrecy: The sad state of server-side TLS session resumption implementations, Nov. 2014. https://timtaubert.de/blog/2014/11/the-sad-state-of-server-side-tls-session-resumption-implementations/.

[400] K. Thomas, D. Yuxing, H. David, W. Elie, B. C. Grier, T. J. Holt, C. Kruegel, D. McCoy, S. Savage, and G. Vigna. Framing dependencies introduced by underground commoditization. In *14th Workshop on Economics of Information Security*, June 2015.

[401] E. Thomé. Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm. *J. Symbolic Comput.*, 33(5):757–775, 2002.

[402] K. Thompson. Reflections on trusting trust. *Commun. ACM*, 27(8):761–763, Aug. 1984.

[403] TRAFFICTHIEF configuration read me. Media leak. https://assets.documentcloud.org/documents/2116187/traffictheif-readme.pdf.

[404] A. Travis. 'snooper's charter' bill becomes law, extending UK state surveillance. *The Guardian*, Nov. 2016. https://www.theguardian.com/world/2016/nov/29/snoopers-charter-bill-becomes-law-extending-uk-state-surveillance.

[405] I. Traynor. Russia accused of unleashing cyberwar to disable Estonia. The Guardian, May 2007. http://www.theguardian.com/world/2007/may/17/topstories3.russia.

[406] I. Traynor. GCHQ: EU surveillance hearing is told of huge cyber-attack on Belgian firm. The Guardian, Oct. 2013. http://www.theguardian.com/uk-news/2013/oct/03/gchq-eu-surveillance-cyber-attack-belgian.

[407] S. Tzu. The art of war. Project Gutenberg, 2005. https://www.gutenberg.org/files/17405/17405-h/17405-h.htm.

[408] Investigatory Powers Act 2016, Nov. 2016. http://www.legislation.gov.uk/ukpga/2016/25/pdfs/ukpga_20160025_en.pdf.

[409] U.S. Intelligence Community budget. Office of the Director of National Intelligence. https://www.dni.gov/index.php/what-we-do/ic-budget.

[410] L. Valenta, S. Cohney, A. Liao, J. Fried, S. Bodduluri, and N. Heninger. Factoring as a service. In *International Conference on Financial Cryptography and Data Security*. Springer, 2016.

[411] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[412] P. C. Van Oorschot and M. J. Wiener. Parallel collision search with application to hash functions and discrete logarithms. In *ACM CCS*, 1994.

[413] P. C. Van Oorschot and M. J. Wiener. On Diffie-Hellman key agreement with short exponents. In *Eurocrypt*, 1996.

[414] In re application of the federal bureau of investigation for an order requiring the production of tangible things from verizon business network services. Media leak, Apr. 2013. https://epic.org/privacy/nsa/Section-215-Order-to-Verizon.pdf.

[415] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *2nd Usenix Workshop on Electronic Commerce*, 1996.

[416] J. Wagnon. SSL profiles part 5: SSL options, 2013. https://devcentral.f5.com/articles/ssl-profiles-part-5-ssl-options.

[417] N. Weaver. In defense of bulk surveillance: It works, Sept. 2015. https://www.lawfareblog.com/defense-bulk-surveillance-it-works.

[418] M. Wertheimer. Encryption and the NSA role in international standards. Notices of the AMS, Feb. 2015. https://www.ams.org/notices/201502/rnoti-p165.pdf.

[419] WhatsApp encryption overview, Nov. 2016. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[420] M. J. Wiener. *Efficient DES key search*. School of Computer Science, Carleton Univ., 1994.

[421] R. without Borders. Enemies of the Internet: 2013 report, Mar. 2013. http://surveillance.rsf.org/en/wp-content/uploads/sites/2/2013/03/enemies-of-the-internet_2013.pdf.

[422] S. Wolchok, E. Wustrow, J. A. Halderman, H. K. Prasad, A. Kankipati, S. K. Sakhamuri, V. Yagati, and R. Gonggrijp. Security analysis of India's electronic voting machines. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, pages 1–14, 2010.

[423] S. Wolchok, E. Wustrow, D. Isabel, and J. A. Halderman. Attacking the Washington, D.C. Internet voting system. In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security*, Feb. 2012.

[424] Wyden: Declassified documents show how inaccurate statements have misled Congress, July 2013. https://www.wyden.senate.gov/news/press-releases/wyden-declassified-documents-show-how-inaccurate-statements-have-misled-congress.

[425] X-KEYSCORE as a SIGDEV tool. Media Leak, 2009. https://theintercept.com/document/2015/07/01/xks-sigdev-tool/.

[426] CNE analysis in XKEYSCORE. Media leak, Oct. 2009. https://assets.documentcloud.org/documents/2116005/cne-analysis-in-xks.pdf.

[427] XKEYSCORE for counter-CNE. Media leak, Mar. 2011. https://assets.documentcloud.org/documents/2116354/xks-for-counter-cne.pdf.

[428] Overview of the package of changes into a number of laws of the Russian Federation designed to provide for additional measures to counteract terrorism. The International Center for Not-for-Profit Law, July 2016. http://www.icnl.org/research/library/files/Russia/Yarovaya.pdf.

[429] Cyberways 2017: Balance of forces in the world. Zecurion Analytics, Jan. 2017. http://www.zecurion.ru/upload/iblock/cb8/cyberarmy_research_2017_fin.pdf.

[430] K. Zetter. Google hack attack was ultra sophisticated, new details show. Wired, Jan. 2010. https://www.wired.com/2010/01/operation-aurora/.

[431] Y. Zhu. Why the web needs perfect forward secrecy more than ever. EFF Deeplinks Blog, Apr. 2014. https://www.eff.org/deeplinks/2014/04/why-web-needs-perfect-forward-secrecy.

[432] P. Zimmermann et al. GMP-ECM, 2012. https://gforge.inria.fr/projects/ecm.