UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Network Attacks and Defenses for IoT Environments

Permalink

https://escholarship.org/uc/item/6b80g2jc

Author

Alharbi, Fatemah M

Publication Date

2020

Supplemental Material

https://escholarship.org/uc/item/6b80g2jc#supplemental

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Network Attacks and Defenses for IoT Environments

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Fatemah Mordhi Alharbi

June 2020

Dissertation Committee:

Professor Nael Abu-Ghazaleh, Chairperson
Professor Jiasi Chen
Professor Michail Faloutsos
Professor Silas Richelson

Copyright by
Fatemah Mordhi Alharbi
2020

The Dissertation of Fatemah Mordhi Alharbi is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I always had doubts that finishing my Ph.D dissertation feels like it will not happen, but in fact, I did it! It is exhilarating, exhausting, terrifying, and thrilling all at once. Thank you Almighty Allah, the most gracious and the most merciful, for blessing me during this journey. I also would like to thank my sponsors, the Saudi Arabian government, Taibah University (TU), and University of California at Riverside (UCR) for sponsoring my Ph.D study. It would not have been possible to write this doctoral dissertation without the help and support of the amazing people around me, and I would like to thank each one of them individually for making this happens, finally!

First of all, I would like to express my deepest appreciation to my academic advisor, Professor Nael Abu-Ghazaleh. He always believed in me, supported me in my professional and personal lives, and guided and mentored me during this academic journey. He is and will always remain my best role model for a scientist, mentor, and teacher. It is truly my great pleasure and honor knowing him.

I would like to thank my committee members, Professor Jiasi Chen, Professor Michail Faloutsos, and Professor Silas Richelson. They generously offered their time, support, guidance and good will throughout the preparation and review of this dissertation.

It was my honor to work with the best minds in the field, my collaborators, Yuchen Zhou, Feng Qian, and Zhiyun Qian, Arwa Alrawais, Michail Faloutsos, Silas Richelson, and Abdulrahman Bin Rabiah. Without their collaborations, I would never finish and publish my work in the past few years. I also would like to thank my wonderful lab-mates: Khaled Khasawneh, Hoda Naghibijouybari, Jason Zellmer, Ahmed Abdo, Hodjat Asghari,

Abdulrahman Bin Rabiah, Shafiur Rahman, Shirin HajiAminShirazi, Sakib Md Bin Malek, Esmaeil (Reza) Mohammadian Koruyeh, Bradley Evans, Ashay, Shirwadkar, and Sankha Dutta. My experience in the lab was greatly enhanced with their support and kindness. Also, I would like to thank my childhood friends back home, in Saudi Arabia, and here, in the US, for all their love and support. I am thankful our paths have crossed.

This journey would not have been possible without the support of my beloved parents, Mordhi Alharbi and Thuraya Solaimani. I would always hope that you are proud of me, and may Almighty Allah protect you. In addition to my sisters: Naseem, Atheer, Khadijah, Sara, and Wejdan, my lovely nephews and nieces. Throughout my life, their love and support has enabled me to pursue what truly interests me and has made me the person I am today.

Finally, and also most importantly, to my lovely husband, Azzam Meeralam, and my little daughter, Zaina: you are my life! Your contribution is invaluable and no words can express how thankful I am that you are my little family. You sacrificed a lot to help me accomplish my childhood dream—to get this Ph.D. Without your support, none of this work would have seen the light of the day.

Dedicated to my beloved parents, lovely husband, and little daughter

ABSTRACT OF THE DISSERTATION

Network Attacks and Defenses for IoT Environments

by

Fatemah Mordhi Alharbi

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2020
Professor Nael Abu-Ghazaleh, Chairperson

Security of resource-constrained edge devices, such as Internet of Thing (IoT) devices, is one of the primary challenges facing the successful deployment of this potentially transformative technology. Due to their resource limitations, often developers have to make a choice between security and functionality/performance, leaving many devices partially or completely unprotected. To illustrate the implications of this situation, I consider a scenario where IoTs (or generally, edge computing devices) are being connected to the Internet and participate in Internet scale protocols such as the Domain Naming System (DNS), Transport Layer Security (TLS), and others. Security solutions for these protocols rely on expensive cryptographic operations that challenge the capabilities of the resource-limited IoT devices. In such a scenario, we are faced with one of three options: (1) Extend security to the edge/IoT devices, therefore sacrificing performance and energy; (2) Sacrifice security, leaving the last link to reach the edge devices insecure; or (3) develop new specialized security protocols, which unfortunately is limited by development type and the need for compatibility with these existing protocols, placing large burdens on developers.

Against this backdrop, I demonstrate a new attack on DNS targeting the last hop, demonstrating that leaving last hop devices poorly protected can lead to their compromise. In the second direction, I propose a new lightweight cryptographic based defense that can promote end-to-end security for IoT and edge computing environments. In the third direction, to understand the nature of operation of IoT devices, I analyze the cryptographic overhead occur on resource-constrained devices when conventional cryptographic algorithms are used: I study the performance of on Arduino MKR WiFi 1010—a single-board microcontroller and compare the results with my lightweight cryptographic algorithm. Lastly, in a different direction, the last contribution is a systematic longitudinal study on Internet filtering in the Kingdom of Saudi Arabia, a traditionally conservative country that has embarked on economic and societal changes in many aspects of its daily operations and public policies with the stated objectives of modernization and openness. These directions are described next.

In the first direction, I identify and characterize a new class of attack targets the DNS service. Specifically, unlike previously reported attacks where the main target is the DNS resolver, the attack targets the client-side DNS cache. The attack allows an off-path attacker to collaborate with a piece of an unprivileged malware to poison the OS-wide DNS cache on a client machine. IoT environments are best fit for this attack since typically network communications between the two last hops (*i.e.,* the default gateway and IoT devices) are un-encrypted. The results demonstrate the effectiveness of the attack on three different platforms: Microsoft Windows, Apple macOS, and Linux Ubuntu. The results show that we can reliably inject malicious DNS mappings within short times that vary

with the specifics of the Operating System implementation. In addition, I present a new analytical model of the attack and compare its prediction with the empirical measurements. The results show that the model correlates with the observed experimental results. I also propose an ad hoc defense which requires only changes to the client DNS cache software, making it practical to deploy immediately through an OS patch. The results show that the defense completely closes this class of attack. For instance, after running the attack for 24 hours, the defense mitigates attacks with no observed successes for the full period. On the other hand, we recorded 1705, 152, and 18 successful attacks on Windows, Ubuntu Linux, and Mac OS, respectively, when the defense is not deployed.

In the second contribution of the dissertation, I propose a more principled approach that can be generalized to provide backward-compatible, low-complexity, end-to-end security for different applications and services enabling the extension of security coverage to resource-constrained environments. More precisely, I introduce a new cryptographic primitive which is called *ciphertext and signature propagation* (CSProp) in order to deliver security to the weak end-devices. I further provide the instantiation of CSProp based on the RSA cryptosystem and the proof of security. I demonstrate CSProp by using it for DNS SECurity (DNSSEC) validation and TLS. The results demonstrate that CSProp provides efficient security support at low additional complexity for IoT environments. I show that the propagated signature verification in DNSSEC (vs. traditional DNSSEC validation) reduces latency by $91x$ and energy consumption by $53x$ on the Raspberry Pi Zero W. For TLS handshake, the advantage to latency and energy by an average of $8x$ and $8x$, respectively. For completeness, CSProp is compared with Elliptic Curve Cryptography (ECC)

cipher suite and found that CSProp outperforms ECC by $2.7x$. On an Arduino MKR WiFi 1010, CSProp achieves significant reduction in latency and power consumption comparing to conventional cryptographic primitives.

The third contribution of the dissertation demonstrates that the first option (*i.e.,* sacrifice performance to retain security using traditional cryptographic algorithms) is not desirable in resource-constrained environments. Specifically, I present a measurement based study to characterize IoT devices with two components: (1) profiling existing devices to understand the cryptographic demands on IoTs; and (2) evaluating their performance on the new proposed primitive, CSProp, and compare the results with a widely used conventional cryptographic primitive which is RSA. For (1), I analyze the cryptographic overhead that occur when an IoT device is used in a home-based environment: the IoT device is a Wyze Cam V2 IoT camera. The results show that the camera uses cryptographic operations intensively. For (2), I conduct a study on a well-known IoT device called Arduino MKR WiFi 1010 which is a single-board microcontroller. I also implement a prototype of the proposed lightweight scheme, CSProp. The results confirm the findings in research direction three that CSProp always outperforms traditional RSA public-key operations in both latency and power consumption. For instance, the execution time for CSProp-encryption and CSProp-verification is 57 and 61 times faster, respectively, compared to traditional RSA encryption for all key sizes. For energy consumption, CSProp provides efficient reductions by $36x$ and $42x$ for encryption and verification, respectively.

The last contribution of the dissertation is a systematic, comprehensive and longitudinal study on the Internet filtering in the Kingdom of Saudi Arabia. Specifically, I

investigate the impact of the Saudi Vision 2030 (announced in April 2016) on the Internet over a period of three years. The investigation shows evidences that Saudi Arabia is cautiously yet decisively opening its digital borders. For instance, I conduct measurements to evaluate Internet behavior by probing Alexa's top 500 websites in 18 different categories and find that the web is becoming more open and accessible. In addition, we find evidence that the emphasis on modernization is leading to relaxing regulations on filtering (67% and 93% of the blocked mobile apps over the period 2013-2017 were accessible in 2018 and 2019, respectively, and all tested apps were accessible in 2020, except WeChat which is still debatable). The investigation also studies the impact of geopolitical events on the filtering in Saudi Arabia. The results show that the filtering policies are reflected in this context. For instance, the results show that ISIS-friendly website are blocked, as ISIS supports terrorism and destabilization to the region. We also find that some news sites from the countries of Qatar, Iran, and Turkey got blocked, amid rising diplomatic tensions between the kingdom and these countries.

For future work, I hope that the lessons learned from these directions help me to build (or at least critically understand) the best framework for IoT and edge computing devices. More precisely, I need to understand the required security primitives that overcome all the three aforementioned challenges.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet of Things (IoTs) has become a ubiquitous term that describes devices that have sensing or actuation capabilities (such as wearable watches, smart home appliances, medical device, and even automobiles). When it first appeared, IoT devices were simply new interconnection of existing technology. However, it has been gaining increasing traction due to its simplicity and efficiency of streamlining the tasks and integrating computing with the real-world. It is projected that there would be more than 75 billion IoT connected devices installed worldwide by the year of 2025 [111]. Given the increasing market penetration of these devices, security considerations in their design are also gaining increasing importance. In particular, the compromise of these devices can introduce not only cybersecurity threats but also threats to safety as IoT devices can be integrated with safety critical systems such as automobiles or medical devices.

According to a data threat report by Thales [259], IoT has become a prime target of cyber-criminals. Nearly 32.7 million IoT-centric attacks have been detected by SonicWall

in 2018 with 217.5% increase comparing to 2017 [138]. The fact that IoT attacks are accelerating at an unprecedented rate is due to manufacturers who in a rush to market do not implement proper security controls. These shortcomings allow cyber-criminals to launch attacks that can be large scale in terms of the number of devices they affect. For instance, in 2016, the world experienced one of the most sophisticated botnet attacks on the Domain Name System (DNS) service, known as Mirai botnet [113, 41]. Mirai exploited over 300,000 IoT devices [195] in 164 countries [112]. Technically, the attack was based on a script to send malicious Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic over port 53. The attack was massive and caused a Distributed Denial of Service (DDoS) on the DNS infrastructure of DNS provider Dyn [158], OVH [187], and Krebs on Security [196]. The attack was feasible because of two vulnerabilities: (1) The software versions of the Linux kernel were outdated. The attackers took advantage of this vulnerability since there were not enough storage space on the IoTs to install the new updated versions of the kernel; and (2) Users did not pay attention to change the default (and weak) usernames/passwords on their devices. A year later, Verizon released a report about another DDoS attack on DNS. The attack was launched by college students [266] and affected more than 5,000 IoT devices. This botnet attack used a brute force approach to break through weak passwords on the IoTs[1]. In Chapter §2, I will describe other attacks and additional related research to the work presented in this dissertation.

[1]I refer interested readers to excellent surveys on IoT attacks for more details [193, 110, 221].

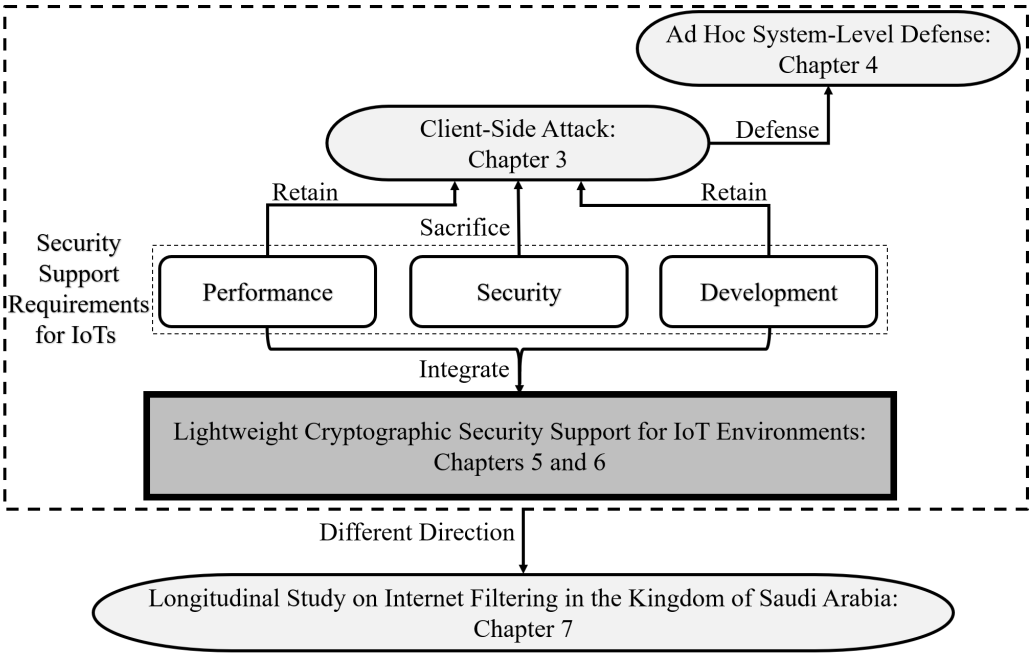


Figure 1.1: Dissertation Overview

1.1 Motivation and Scope of Work

My work considers a common scenario in networked IoT systems where IoT devices (or generally, edge computing devices) are being connected to the Internet and participate in Internet scale protocols such as the Domain Naming System (DNS) [218, 219], Transport Layer Security (TLS) [240], Hypertext Transfer Protocol Secure (HTTPS) [239], and others. Traditionally, security solutions for these protocols rely on cryptographic algorithms which require computationally expensive operations that can be far too complex for such low-power and inexpensive devices. In such a scenario, one of three options will occur: (1) Retain security and extend it to the edge/IoT devices — in this case performance and energy will suffer; (2) Sacrifice security, avoiding the resource overhead, but leaving the last link to be insecure; or (3) Rewrite the entire system or even develop new security protocols

specifically to incorporate these lightweight devices. The third option is undesirable since it does not support backward compatibility and places large burdens on developers and operators.

In the light of this context, my research pursues three main directions in addition to a side direction to recommend further research (Figure 1.1 shows an overview of the chapters that compose this dissertation):

1. The first contribution demonstrates that *manufacturers should mandate security* in developing their edge/IoT systems. Specifically, I demonstrate a new attack on the DNS infrastructure targeting the last hop. I present the attack component of this contribution in Chapter §3, and also introduce an ad hoc defense specifically for this vulnerability in Chapter §4.

2. Motivated by the aforementioned research direction, in the second direction, I propose a novel cryptographic function that allows full participation in cryptographic operations but at a low computational cost, allowing the integration of security, performance, and development without sacrificing one to save the other. More details about this research direction is presented in Chapter §5.

3. In a third direction, I explore general usage of cryptography in resource-constrained environments using a measurement study. Specifically, I analyze and measure the cryptographic overhead of conventional cryptographic primitives that has been used extensively in various applications. I present performance results related to latency and energy consumption confirm with the measurement results in the third research direction. I present this contribution in Chapter §6.

4. The last contribution explores a different research direction–Internet Filtering in Saudi Arabia. Specifically, I present a systematic longitudinal study on Internet filtering in the Kingdom of Saudi Arabia, a country that has embarked on economic and societal changes in many aspects of its daily operations and public policy decisions, with the stated objectives of modernization towards building a more tolerant Islamic country. This contribution is detailed in Chapter §7.

An overview of these contributions are presented in more detail in the remainder of this Chapter.

1.2 Contributions of the Dissertation

The research in this dissertation focuses on showing that the integration of three criterias which are security, performance, and development can help to provide efficient end-to-end security. Such an integration adds lightweight security support for resource-constrained devices such as IoTs and edge computing devices. Therefore, in this section, I describe my contributions towards solving the problems described in the previous section (Section §1.1); network attacks and defenses for IoT environments. Finally, I overview the contributions in regards to Internet Filtering in the Kingdom of Saudi Arabia.

1.2.1 Contribution I: Client-Side DNS Attack

The first contribution of this dissertation demonstrates a new and dangerous DNS cache poisoning attack targeting edge devices. Generally, DNS poisoning attacks inject malicious entries into the DNS resolution system, allowing an attacker to redirect clients

to malicious servers. These attacks typically target a DNS resolver allowing attackers to poison a DNS entry for all machines that use the compromised resolver. However, recent defenses can effectively protect resolvers rendering classical DNS poisoning attacks ineffective. I contribute and present a new class of DNS poisoning attacks targeting the client-side DNS cache. The attack initiates DNS poisoning on the client cache, which is used in all main stream operating systems to improve DNS performance, circumventing defenses targeting resolvers. The attack allows an off-path attacker to collaborate with a piece of an unprivileged malware to poison the OS-wide DNS cache on a client machine. I developed the attack on Windows, Mac OS, and Ubuntu Linux. Interestingly, the behaviors of the three operating systems are distinct and the vulnerabilities require different strategies to exploit. The attack is also generalized to work even when the client is behind a Network Address Translation (NAT) router. The results show that we can reliably inject malicious DNS mappings, with on average, an order of tens of seconds. The attack is described in more details in Chapter §3.

I also introduce an ad hoc System-Level Defense that targets this specific vulnerability (presented in Chapter §4.) The defense hardens the clients against DNS cache poisoning attacks by first detecting an attack, and then deploying both operating system and networking defenses against it. It requires only changes to the client software; thus, it can be deployed immediately through an OS patch. The results show that the defense successfully defeats all attacks with 0% attack success probability.

1.2.2 Contribution II: CSProp, a Lightweight Cryptographic Primitive for Securing IoT

The defense introduced earlier is both ad hoc, and also specifically targeted at the attack I discovered. To provide a more systematic option to enable protection of IoT devices across many protocol settings, I propose CSProp– a new cryptographic primitive providing full security but with an overhead acceptable for resource-constrained devices. More precisely, in networked and distributed systems, cryptographic operations are used to ensure the privacy and authenticity of information as it travels through multiple intermediaries. These operations can be computationally expensive, and typically must be repeated every time information changes hands. This poses a security challenge when lightweight, resource-constrained devices (such as those common in IoT systems) are connected to a network of more capable machines. As discussed earlier, one of three sub-optimal outcomes will occur if cryptography is applied naively: (1) performance will suffer (*e.g.*, if resource constraints of the lightweight devices are ignored); (2) security will suffer (*e.g.*, if cryptography is removed from all interactions involving the lightweight devices); or (3) developers will suffer (*e.g.*, if the entire system is rewritten specifically to incorporate these lightweight devices). The third might not even be a possibility due to the system's security requirements or due to the need to preserve legacy systems. Motivated by these concerns, I introduce a new cryptographic primitive which is called *ciphertext and signature propagation* (CSProp) in order to deliver security to the weak end-devices. The core of this contribution is a propagation algorithm whereby a capable machine sitting upstream of a lightweight device can modify an authenticated message so it can be efficiently verified by a lesser machine. Crucially,

verification of the modified signature is less computationally intensive than verification of the original (thus avoiding pitfall 1 above). Nevertheless, security guarantees that the only way to produce a lightweight signature is by modifying an original (thus avoiding 2). Best of all, the proposed propagation algorithm works for RSA signatures and so is compatible with a large fraction of internet traffic (avoiding 3). A similar propagation technique is given for RSA ciphertexts which allows a weak device to use an efficient encryption procedure to produce a ciphertext that can then be propagated forward by a capable machine into a standard RSA ciphertext of the same message. CSProp is demonstrated by using it for the extended version of DNS known as DNSSECurity (DNSSEC) validation and TLS. The propagated signature verification in DNSSEC (vs. full DNSSEC validation) reduces latency by $91x$ and energy consumption by $53x$ on the Raspberry Pi Zero W. For TLS handshake, the advantage to latency and energy by an average of $8x$ and $8x$, respectively. I also compare CSProp with Elliptic Curve Cryptography (ECC) cipher suite and found that CSProp beats up ECC by 2.7 times. CSProp is presented in more details in Chapter §5.

1.2.3 Contribution III: Analysing Cryptographic Overhead on IoT Devices

The third contribution of this dissertation is a measurement based study to characterize IoT devices with two components: (1) profiling existing devices to understand the cryptographic demands on IoTs; and (2) evaluating their performance on the new proposed primitive, CSProp, and compare the results with a widely used conventional cryptographic primitive which is RSA. For (1), I analyze the cryptographic overhead that occur when an IoT device is used in a home-based environment: the IoT device is a Wyze Cam V2

IoT camera. The results show that the camera uses cryptographic operations intensively. For (2), a study is presented to measure the performance of Arduino MKR WiFi 1010—a single-board microcontroller representative of IoT devices, when it participates in IoT operations. The performance is analyzed in terms of cryptographic overhead on latency and energy consumption. Based on the obtained results, we can confirm the finding in the third reasrch direction that CSProp is suitable for resource-limited environments. This measurement study is detailed in Chapter §6.

1.2.4 Contribution IV: Characterizing Internet Filtering in Saudi Arabia

The last contribution in this dissertation considers is in a different research space, considering another aspect of Internet operation. Internet filtering is predominantly used by countries or organizations to restrict access to websites or other resources on the Internet that are perceived to hold inappropriate, offensive, or otherwise harmful content with respect to their governing laws, values, or policies. In this research direction, I study Internet filtering in the Kingdom of Saudi Arabia: a traditionally conservative country that has embarked on economic and societal changes in many aspects of its daily operations and public policies with the stated objectives of modernization and openness.

I present a comprehensive longitudinal study of Internet filtering in Saudi Arabia over the period of three years. Specifically, I conduct measurements to evaluate filtering behavior by probing Alexa's top 500 websites in 18 different categories from viewpoints covering the three largest telecommunications companies in Saudi Arabia. I observe that the filtering is most common for sites in the *Adult*, *Shopping*, *Games*, and *Global* categories. I also conduct measurements to test mobile application accessibility by examining the status

of 18 of the most popular mobile social network applications worldwide and in the middle east such as WhatsApp, Facetime, WeChat, and Skype. Our results show that Saudi Arabia is cautiously yet decisively opening its digital borders. For example, we find evidence that the emphasis on modernization is leading to relaxing regulations on filtering (67% and 93% of the blocked mobile apps over the period 2013-2017 were accessible in 2018 and 2019, respectively, and all tested apps were accessible in 2020, except WeChat which is still debatable).

The availability of multiple measurements over time enables us to study changes in the filtering policy and view these changes in the wider geopolitical context experienced by the kingdom and the region. For instance, we find that ISIS-friendly sites are blocked, as ISIS supports terrorism and destabilization to the region. Interestingly, news sites from the countries of Qatar, Iran, and Turkey got blocked, amid rising diplomatic tensions between the kingdom and these countries. I also investigate and characterize the technical mechanisms and the network topology used in the implementation of the filtering. This direction is presented in detail in Chapter §7.

Finally, Chapter §8 concludes before highlighting potential future work.

Chapter 2

Background and Related Work

In this chapter, I present the related work on this dissertation. The organization of this chapter is as follows. Relevant to the first contribution, in Section §2.1, I present the related work of DNS cache poisoning attack and in Section §2.2, I provide the state of the art defenses to mitigate the attack. With respect to the second contribution, in Section §2.3, I describe lightweight cryptographic solutions in favor of adding security support in IoT environments. To conclude this chapter, I present related work on the third contribution on Internet filtering in Section §2.4.

2.1 DNS Cache Poisoning Attacks

DNS cache poisoning is a dangerous class of attacks that has been the focus of studies in the past [60, 268, 189, 250]. It is accomplished when an attacker injects a malicious mapping into a DNS cache to redirect communication to an adversarial server enabling the attacker to intercept packets content. In 2007, Amit Klein introduced sophisticated cache

poisoning attacks against BIND 9 DNS resolvers [188] and Windows DNS servers [190]. At that time, the attack's entropy was totally based on the TXID, and the implementation of the randomized algorithm facilitated the attacks. In 2008, Dan Kaminsky presented another significant attack [180] against DNS resolvers which also depends on TXIDs for authentication. The attack assumes both the UDP source and destination ports are fixed as 53. Indeed, Paul Vixie already reported this vulnerability in 1995 [268], and in response, Bernstein proposed a challenge-response defense to substantially use ephemeral ports randomization in order to expand the entropy of the correct response packet [61, 62]. However, this solution was not practically supported until Kaminsky's attack [167, 198, 197].

More recently (starting in 2011), several new cache poisoning attacks against resolvers were proposed which have varying degrees of assumptions of the attack requirements and the network [156, 143, 155, 157, 254, 191]. For example, some attacks [156, 254] assume an attacker collaborates with a malicious script (*e.g.,* Javascript in a browser) to poison the DNS cache of a resolver. Herzberg and Shulman [156] propose an attack that exploits packet fragmentation of UDP packets of long DNS responses to spoof the second fragment of a DNS response (only the first fragment includes the TXID). The same authors [157] propose a poisoning attack that exploits delegation of DNS resolution where intermediary network devices perform recursive lookups on behalf of the resolvers. In contrast, they also show attack principles (but do not demonstrate the attack) [155, 143] that can poison the cache of a DNS resolver located behind a NAT. These attacks can be prevented by full installation of DNSSEC. They do not target the client caches as my attack does. Recently, the attack in [191] does not focus on the challenge-response authentication parameters, such

as the UDP source port and the TXID, and assumes they are known to the attacker (which we doubt it is a very strong assumption). However, none of them is a general attack that can work universally (as the Kaminsky's attack [180]).

2.2 DNS Cache Poisoning Defenses

Because attacks that undermine DNS resolution are extremely powerful, several defenses were proposed to address DNS cache poisoning attacks. The preponderance of these defenses targets improving DNS resolver security and therefore are not effective against my attack. The defenses can be classified into three categories: challenge-response defenses, cryptographic defenses, and using alternative architectures. Challenge-response defenses rely on the idea of increasing the entropy of DNS request/response, such as UDP source port randomization [180, 9], 0x20 encoding [104], random selection of authoritative name servers [197, 167], and adding random prefixes to domain names [229]. All these defenses attempt to make the space of packets from which a correct response must be selected larger, substantially increasing the attacker's difficulty in generating a valid response. Challenge-response defenses are vulnerable to MitM adversaries who can intercept the DNS communication. To protect against this type of eavesdropping attack, cryptographic defenses were proposed which include primarily DNSSEC [272] that is based on digital signatures for authentication; this solution in principle closes all cache poisoning attacks since the validity of the response is no longer only a function of the contents of the packet. Despite the fact that DNSSEC is effective, the deployment is very slow. For example, the Internet Society organization reported that roughly 10% of a sample of size more than 735 million

resolvers use DNSSEC validation [8]. Moreover, a recent study [7] discovered that 0.67%, 0.91%, and 0.91% of .com, .net, and .org Top Level Domains (TLDs) are signed. Recently, Klien et al. did an Internet-scale measurement study on the vulnerability of DNS resolvers and discovered that 92% of resolvers are vulnerable to at least one type of poisoning attack [191]. All these studies conclude that complete adoption of DNSSEC would prevent these vulnerabilities, but a combination of partial deployment and permissive policies (e.g., accepting responses even if the DNSSEC signature does not match) lead to the current resolver vulnerabilities. Unless DNSSEC is extended to cover end clients, which introduces a substantial key distribution problem and is likely to infringe on usability, it does not prevent my attack.

A third defense alternative considers rethinking DNS implementation radically, resulting in different security properties. For example, Schomp et al. [249] propose a radical change to the DNS ecosystem by eliminating shared resolvers entirely to have clients perform the recursive resolutions directly. However, since my attack targets the endpoints, it should still be effective against this architecture.

Currently, the security of patched operating systems relies mostly on the randomness of source ports. In my attack model, I expose vulnerabilities in the source port allocation algorithms used by three operating systems: Microsoft Windows, Mac OS, and Linux Ubuntu.

2.3 Lightweight Cryptographic Defenses for IoT Environments

Lightweight cryptography is a term that refers to low overhead cryptographic algorithms designed for energy- or computationally-constrained machines, specially in IoT environments. This is an active area of research in both academia and industry [116, 129, 75, 65, 182, 77, 54, 67, 253, 242]. The main goal of such algorithms is to achieve efficient security suitable for such resource-constrained devices. However, due to the limited resources (such as processing power, battery, and storage capacity) of these devices, it is a challenge to fulfill this goal. As a result, designing lightweight cryptographic schemes for resource-constrained devices is a topic that has attracted broad interest from the research community [182, 203, 237, 124, 175, 278, 236, 281]. Since Public Key Cryptography (PKC)/asymmetric cryptography requires substantially higher resources for computation than symmetric cryptography, most lightweight cryptographic algorithms have been developed for symmetric cryptosystems.

Symmetric Lightweight Cryptography. In mCrypton [203], Lim et al. follow the architecture of Crypton [202] to design a lightweight protocol for resource-constrained devices such as Radio Frequency Identification Systems (RFIDs) [270]. More precisely, the authors propose 64-bit block cipher with three key sizes ranging from 64 to 128 bits to optimize resource usage and power consumption. Nonetheless, the smaller key sizes would led to a weak security. Whereas the proposed scheme, CSProp (see Chapter §5 for more details), utilizes a sufficient key size to form a secure and scalable connection. Similarly, the Scalable Security with Symmetric Keys (S3K) scheme [236], which is a key management

architecture, was proposed to provide a scalable energy efficient mechanism to establish trust relationships among entities in IoT environments. The architecture consists of two phases: (1) establishing a new pre-shared key within the Datagram Transport Layer Security (DLTS) handshake; and (2) applying lightweight public key cryptography. Basically, they depend on shared keys which might increase the risk to key disclosure and endanger the security services. This sets S3K scheme apart from CSProp, where the latter does not require any pre-shared key and thus maintain the security for both ends. Another proposal is Hummingbird [124] which uses a hybrid structure of block and stream ciphers with 16-bit block size and 256-bit key length; an improved version of this work has been developed by Engels et al. [125]. Specifically, the algorithm increases the internal state to 128-bit and enhances the state of the mixing function of the algorithm. It is tested against most of the well-known attacks and it appears to be resilient against them. However, Zhang et al. [279] show that the key can be easily recovered. Unlike CSProp is based on unbreakable and scalable RSA signature algorithm. Jan et al. [175] propose a different approach for IoT device identification. Specifically, the scheme observes resources of IoTs that are based on the Constrained Application Protocol (CoAP) to provide mutual authentication by validating the identities of the participating devices before engaging them in communication. Even though the scheme seems to close vulnerabilities that may lead to eavesdropping and key fabrication attacks, it does not provide protection from several powerful attacks such as the well-known Sybil attack [118] in which a device can illegitimately impersonate multiple identities. In CSProp, major attacks are discussed in §5.1.1 that CSProp can efficiently defeat.

Asymmetric Lightweight Cryptography. Other efforts aim to propose a lightweight cryptographic algorithm based on asymmetric ciphers. For instance, Lithe [237] is proposed to provide integration of security between the DLTS protocol at the transport layer and the CoAP protocol at the application layer. Lithe compresses the datagrams of DLTS to increase efficiency and applicability of DTLS and CoAP for constrained devices. On the other hand, the system involves expensive cryptographic processing for both the record and the handshake protocols. Comparing with CSProp, it does not require any preprocessing expensive computation that increases the overall overhead time. Zhang et al. [281] propose a scheme to provide resilience against a large number of sensor node compromises. The authors propose a message authentication scheme by adopting perturbation polynomials. The scheme incurs lower overhead and higher adaptability than existing techniques that utilize traditional asymmetric algorithms. However, Albrecht et al. [32] show an attack that fundamentally undermines the viability of using perturbation polynomials for designing secure cryptographic schemes. Whereas CSProp utilizes an efficient and secure cryptographic algorithm based on RSA signature (See §5.2 for more details). There also exist approaches that utilize custom authentication protocols that rely on Public Key Infrastructure (PKI) and TLS protocols [39]. However, PKI and TLS protocols require resource-consuming cryptographic operations making them inappropriate for IoT and other constrained devices. CSProp does not utilities PKI platform since is not suitable for most types of constrained devices.

Private-Key Lightweight Cryptography. Most work on lightweight cryptography targets private-key cryptography (symmetric cryptography algorithms such as block/stream

ciphers and hash functions [203, 206, 160]), with ongoing efforts to standardize some proposals [173].[1] Progress on lightweight (PKC) has been slower: PKC is much more computationally demanding, and protocols attempt to minimize their use for encryption and decryption (by using PKC to negotiate session keys for private key algorithms). However, PKC remains critical for authentication and verification as well as to bootstrap the use of private key cryptography. Elliptic Curve Cryptography (ECC) can reduce the overhead of some PKC cryptographic operations relative to RSA [75, 264]. Primarily, the lower overhead occurs because ECC can provide equivalent security to RSA with much smaller keys, making key generation as well as operations that use private keys substantially faster. However, because RSA can use low public exponents (effectively much smaller public keys) operations using public keys are considerably faster in RSA. These operations include authentication and signature verification (my first application), which has been measured to be 6.6x, and 3.4x faster in 1024-bit RSA and 2048-bit RSA, respectively, than ECC [84, 265]. Considering DNSSEC, this is a serious issue because overloading DNS resolvers could potentially slow down name resolution with wide impact on many applications. CSProp improves the performance of RSA, which should result in the fastest known PKC signature verification with security equivalent to RSA, and with backward compatibility.

Proxy Assisted Cryptography. The proposed lightweight cryptographic scheme, CSProp, bears similarity, with prior work on proxy-based re-signature schemes, first introduced by Blaze et al. [66] and later revisited by Ateniese and Hohenberge [52]. A significant difference from these works is that CSProp provides security by construction

[1]The desired properties of lightweight cryptography have already been discussed in the International Organization for Standardization ISO/IEC 29192 in ISO/IEC JTC 1/SC 27: https://www.iso.org/committee/45306.html

and therefore does not require a trusted proxy to propagate signatures. In contrast, these prior works require a trusted proxy to take a signature as input and generate a new signature as output using the public keys of both parties (the signer and the verifier). It is worth noting that this setting is also vulnerable to a known attack on RSA [68, 255, 256].

Joye et al. [178] propose a solution to overcome hardware restrictions enforced by vendors such as Intel Software Guard Extensions (SGX). Although it uses a proxy, the application is different: for CSProp, Patty is helping a weak Alice verify a signature from a more powerful Bob. On the other hand, in this work, Patty is helping a weak Bob sign a message and transmit the signature to a powerful Alice, which is not a common scenario for IoT settings. Critically, Patty needs to know Bob's private key, whereas for us, Patty does not. For this reason, the warning in [178] that $e'|e$ is problematic does not apply in my setting. This is because e' in [178] is generated using knowledge of Bob's private key, whereas for us e' is computed publicly. Another major difference between this work and CSProp is the model of security they consider: in addition to the proxy being trusted in this scheme, both public keys must be kept secret, which is incompatible with my requirements and assumptions.

Ding et al. [114] attempt to carry out proxy assisted cryptography, making the scheme similar to CSProp at a high level. They propose a solution to speed up public key signature generation for resource-constrained devices using Server-Aided Signatures (SAS). This scheme requires that an original signer should have a PKI certificate, then compute a hash chain of one-time secret keys generated from a root key associated with the certificate, and store the chain locally. The original signer needs to collaborate with a trusted proxy

which is trusted and has a PKI certificate. It uses its own private key to *fully* sign the message along with one of the one-time secret keys. These keys are used to authenticate the original signer and can be used only once for each individual signature. Under heavy network traffic, this introduces a performance bottleneck for the original signer unless she has enough storage to store the hash chain– a limitation in resource-constrained devices. In contrast, CSProp does not require additional PKI certificates, and the propagator can be any entity since it is not required to be trusted.

Several studies have been conducted presenting the computational and energy costs of cryptographic protocols on embedded IoT processors [55, 213, 232, 148, 166]. For instance, Potlapally *et al.* [232] shows how fast the battery gets drained (more than twice) in the presence of encryption comparing to no encryption. They also show energy analysis of common cryptographic algorithms (such as RSA [241], DSA [224], and ECDSA [176]) on a client device running Compaq iPAQ H3670 whose processor is clocked at 206MHz. In addition, a good body of work analyzes the performance of specific applications used in resource-constrained environments [213, 142, 123]. For instance, Miranda *et al.* [213] analyzes the energy consumption of the Transport Layer Security (TLS) protocol transactions on a mobile device and found that more than 60% of total energy is consumed by TLS overhead.

2.4 Internet Filtering

As the Internet has grown to be an essential service for accessing information, sharing opinions, coordinating activist organization, many countries have sought to regulate

this open access through Internet filtering. As a result, many studies have been conducted to examine the filtering practices and mechanics in different countries around the world. In this section, I briefly survey a number of these studies and explain their relationship to my work where appropriate.

Country-Specific Internet Filtering studies. For some countries, the underlying motivation to effect filtering believed to be political. For instance, Nabi [220] used a publicly available dataset of websites to check their accessibility in Pakistan. He found that the government performs filtering at DNS and HTTP levels. Aryan et al. [51] showed evidence that all Internet traffic in Iran is directed to a centralized equipment which is apparently controlled by the government. In 2011, during the political events in Libya and Egypt, Dainotti et al. [105] analyzed country-wide government-ordered Internet outages using a variety of publicly available datasets. Furthermore, Chaabane et al. [79] presented results of measurements analysis of a sophisticated Internet filtering system enforced by the Syrian government. They discovered that Instant Messaging is heavily censored. Many studies have explored the Internet filtering infrastructure of the Great Firewall of China (GFW) over the years [276, 201, 83]. In the UK [82] a system filters pedophile advocacy websites that promote sexual exploitation of children. In Germany, all Nazi promoting websites are blocked by the government [117]. Internet filtering can also be imposed by Internet users on themselves. For instance, Gebhart et al. [139] presented an adequate and a comprehensive study on Internet filtering in Thailand. They distributed a survey on 160 respondents and found that nearly 70% of them enable filtering settings on their connections. At the economic level there is evidence of some companies or organizations using Internet filtering to

force customers to use specific products [85]. We expect such economically driven filtering to increase if net neutrality repeal efforts go into law in the USA.

Studies on Internet Filtering in Saudi Arabia. Most related to my study, Zittrain and Edelman conducted the first study on Internet filtering in Saudi Arabia [282]. In 2002, they used proxy servers to prop a list of random distinct websites (approximately 64,557) from 6 different categories. They performed a lightweight measurement study and concluded that nearly 3.15% of the websites were blocked based on different categories. Likewise, in 2004, the OpenNet Initiative (ONI) organization published a similar report [170] and confirmed the findings in [282]. Most recently, in 2012, Verkamp and Gupta [267] studied the filtering mechanics used in 11 countries, including Saudi Arabia. They claimed that Internet filtering in Saudi Arabia is based on destination IP address filtering and directs users to an HTTP response with a status code of 200. My study is similar in spirit, but substantially larger in scale, covering systematically classes of websites and mobile applications. The study is also repeated over multiple years at a time where Saudi Arabia is experiencing a shift to modernize. I also explore the technical mechanisms underlying the observed behavior in detail. In fact, the results show substantial differences from these earlier studies; for example, we found that filtering is applied at the HTTP and TLS levels instead of IP. The analysis was repeated multiple times and observe trends in filtering over time.

Filtering Measurement Tools. Several research efforts developed tools to measure and study Internet filtering. For instance, CensMon [252] was developed as a tool to monitor and detect filtering characteristics globally. UBICA [29] was designed to aggressively collect filtering-related data from different vantage points by running the tool on home gateways

and personal computers. OONI [134] and ICLab [169] are two other platforms that are designed to detect filtering using embedded devices such as Raspberry Pis [230]. Iris [228] is also developed as a scalable, accurate, and ethical method focusing on the problem of measuring manipulations on the DNS protocol. In addition, Nabi used a test script, dubbed Samizdat [220], that inspired my filtering tool. Samizdat first downloads a list of websites and prepares it for testing. For each website, the tool performs a DNS lookup to check for DNS level filtering. It then tries to establish a TCP connection. Next, the tool checks for HTTP-URL-keyword filtering. In the last step, the tool checks for HTTP-FQDN filtering. For each test, the results are recorded in a log file locally consistent with recommended ethical practices (*i.e.,* not in a remote server as in [252]). The tool which is developed for this study provides significant new functionality. For example, some of the modifications include a different website input and preparation process: the websites are distinct per category since they are crawled directly from Alexa and do not need cleaning to remove redundancy.

Data Resources. The are also some efforts in terms of data resources. For instance, ONI [226] makes global Internet filtering data more accessible to researchers and journalists. However, their data is outdated with the last release being in September 2013 while for Saudi Arabia the latest release was in 2009. The University of Michigan, on the other hand, established a project called Censored Planet [78] to frequently update the filtering data. The project contains a publicly accessible database at global scale.

Chapter 3

Collaborative Client-Side OS-Wide DNS Cache Poisoning Attack

The Domain Name System (DNS) protocol provides an integral service underlying the Internet: It is an essential component that provides resolution primarily of Fully Qualified Domain Names (FQDNs) (*i.e.,* human-readable domain names such as `foobar.com`) to their corresponding Internet Protocol (IP) addresses [218, 219]. DNS resolution information is maintained by a hierarchical and distributed set of name servers in order to support scalability and to enable distributed management at each individual organization. This hierarchy consists of 13 trusted root servers (denoted by `.`) which are geographically widespread across the world, top-level domains (TLDs) (*e.g.,* `.com`, `.edu`, and `.gov`), and authoritative name servers (*e.g.,* `foobar.com`, `ucr.edu`), and `usa.gov`. DNS queries from clients are serviced using a set of resolvers that can walk the DNS hierarchy to reach an authoritative name server that provides the answer to the DNS query. To improve perfor-

mance, resolvers and end hosts heavily use caching, exploiting locality to avoid unnecessary and slow queries that consist of several round-trips as a resolver walks the DNS hierarchy.

DNS has gradually evolved since its first specifications [215, 216, 217, 218, 219]. It is widely used in verity of applications and Internet services and architectures which rely on the integrity and availability of the DNS infrastructure.

The security of DNS is critical to the security of the Internet: if an attacker can manipulate the mapping, she can redirect connections to cause users to access a malicious server, to facilitate Man-in-the-Middle (MitM) attacks, or to cause denial of service (DoS).

One of the most serious attack classes against DNS is the cache poisoning attack, where an attacker attempts to inject malicious DNS mappings to the cache of a DNS resolver [53, 180, 174, 188].

DNS is vulnerable to this type of attack because, in the process of resolving a domain name, requests are sent to authoritative name servers which can be far away, leaving an open window of time for an attacker to inject a false response. More precisely, an attacker can poison the cache by impersonating authoritative name servers of a target domain [180, 143, 189]. For example, an attacker who wants to target the domain name *www.bank.com* first needs to have a server capable of spoofing the IP address of the *bank.com*'s authoritative name server. When the victim resolver sends a DNS query to resolve *www.bank.com*, the attacker impersonates the legitimate DNS server by spoofing a response with a malicious IP. The malicious mapping provided by the attacker is cached by the resolver and all future queries for *www.bank.com* will return the malicious IP address redirecting victims to the IP address chosen by the attacker, allowing several dangerous exploitation classes. For

example, users may be fooled to enter their credentials on a malicious website identical to the website they were trying to access. To protect against cache poisoning attacks on DNS resolver caches, Source UDP Port Randomization (SPR) [180, 9] was introduced and is currently widely deployed. In this defense, a DNS query from a resolver uses a random source UDP port when forwarding a DNS query. An off-path attacker must guess this random port number in order to successfully spoof a reply to the same port (otherwise, the reply will not be accepted), in the time window before the true response is received. Although it does not close the vulnerability completely, SPR substantially reduces the chances of effective cache poisoning attacks. Even though fundamentally secure DNS protocols such as *DNSSEC* [272] have been proposed, it has been difficult to get traction with respect to real-world deployment, and most websites continue to run insecure versions of DNS.

In this work, I introduce a new and dangerous DNS poisoning attack targeting the end user devices. Most operating systems on client devices use DNS caches that retain DNS responses and share them across applications including browsers. The results show that these caches can be compromised via a DNS cache poisoning attack oftentimes in a couple of seconds for Windows and a few minutes for Ubuntu Linux and Mac OS. Specifically, the attack is initiated by an *unprivileged* malicious program (*e.g.,* a malware or a malicious JavaScript) who simply asks for DNS resolution for a domain it is attempting to poison. The malicious program coordinates with an off-path attacker (*i.e.,* an attacker anywhere on the Internet) that responds to the DNS request attempting to poison the cache entry and succeeding with high probability. To succeed in the attack, a malicious response with a matching TXID has to arrive before the real response. This is challenging task as there

is really only an attack window equivalent of a round-trip time between the client and resolver. To make matters worse, once the authentic DNS response is cached, one may need to wait for the entry to timeout before being able to launch the next round of attack on the same domain. However, I discover that *specific OS implementations* and *real-world TTL/network latency* make the proposed attack highly feasible. Through analysis of NAT implementation on commercial routers, reliable strategies are presented to launch the attack even through a NAT router.

Client devices are typically not considered to be part of the DNS hierarchy and therefore have not been considered by defenses against DNS cache poisoning. Thus, defenses against resolver cache poisoning attacks including SPR [180, 9] and 0x20 [104] do not protect against this new attack. Even new proposals such as DNSSEC which rely on cryptography to completely close cache poisoning [46] operate at the resolver level but leave the network behind the resolver unprotected. As a result, the attack represents a new and dangerous vulnerability that threatens most computing devices. The attack also expands my understanding of the threat surface of DNS cache poisoning attacks when designing mitigations within DNS.

Additionally, I try to fulfill my understanding on the attack behavior. I add theoretical analysis and compare it with the measured results. Specifically, more measurements are added to help understand the impact of number of attack trials on attack success.

This research direction makes the following contributions.

1. I describe a new and dangerous DNS cache poisoning attack against client caches, which overcomes the challenges of source UDP port randomization. The techniques

can defeat the challenge-response defenses that are mostly based on randomizing TXIDs and port numbers.

2. I explore the attack when the victim is behind a NAT. The results show that many NAT boxes are fixed-port and/or port-preserving, enabling the attack to work without modification. Although I did not have access to a random-port NAT router, I conceptually describe approaches to extend the attack to work with such routers.

3. The attack is demonstrated on three different operating systems: Microsoft Windows, Mac OS, and Linux Ubuntu machines. Each implementation introduces challenges, but I show that they are all addressable, enabling the attack to work with reliably. The attack's effectiveness is measured under various settings, and the results show that the attacks are potent under realistic conditions. For instance, the attack typically succeeds in a few seconds on Windows.

4. I present a new analytical model of the attack and compare its prediction with the empirical results. Specifically, I extend the analytical model of the attack to estimate the number of rounds before an attacker successfully poisons the cache. Although the model does not capture network effects such as congestion and packet drops in the receiver buffers, I show that it correlates with the observed experimental results.

Disclosure. I reported the attack to Apple, Microsoft, and Ubuntu. In response, Apple released a security update[1] fixing the `mDNSResponder` daemon, also known as Bonjour, that is used by the DNS Service Discovery API in Mac OS X (version 10.2 and later) and iOS

[1] https://support.apple.com/en-us/HT209446

operating systems. Ubuntu confirmed the vulnerability[2]. I understand that Microsoft is considering mitigations.

The remainder of the chapter is organized as follows. Section §3.1 provides the background and describes the threat model. The attack is introduced in Section §3.2. Tailored attack strategies and analysis are presented in Section §3.3. The attack is evaluated in Section §3.4.

Readers are encouraged to watch a video demo illustrating the proposed attack on Windows:

`https://www.youtube.com/watch?v=ulaccbjA6ZU`

3.1 Attack Fundamentals and Threat Model

In this section, I set the table for the client-side DNS cache poisoning attack. First, the behavior of the OS-wide DNS cache of operating systems is described and then I discuss the threat model.

3.1.1 OS-Wide DNS Caches

Modern OSes have built-in DNS caches. These caches are shared OS-wide, meaning that if an application populates an entry in the cache, this entry will be used by any other application that requires resolution for the same name. Similar to records cached at the DNS resolvers, an OS-wide DNS cache record is stored along with a Time-To-Live (TTL) value which is set by the domain authoritative nameserver to determine the lifetime

[2]https://bugs.launchpad.net/ubuntu/+source/systemd/+bug/ 1782225

of the record in the cache. The purpose of having such a cache is to improve the performance of DNS resolution as it is on the critical path of accessing Internet resources, especially for applications that have many short-lived connections. Unlike previous work where the focus was on resolver's caches, I study the behavior of OS-wide DNS caches of operating systems.

When a client machine issues a DNS request, the request reaches the authoritative name server having the answer after traversing a chain of DNS servers/resolvers and different layers of caches. Some operating systems, e.g. Linux Ubuntu 16.04.6 LTS and earlier, are configured to use an external DNS server as their first resolver and uses its cache for future queries. Another alternative is to implement a client-side DNS cache internally. In this case, the OS first accesses the OS-wide cache and checks if the DNS record corresponding to the client's request is cached; if it is not cached, the OS forwards the request to the first external resolver in the Internet cloud; e.g. carrier's resolver. The latter implementation is vulnerable to client-side cache poisoning attacks. The attack differs from other DNS cache poisoning attacks in that it targets the client-side cache, and therefore bypasses all known defenses that protect the resolvers.

I surveyed a number of modern operating systems, including macOS Sierra version 10.12, several versions of Microsoft Windows (Microsoft Windows 7 Professional Edition, Microsoft Windows 8.1, and Microsoft Windows 10), as well as several Linux distributions. By default, the OS-wide cache is enabled in all versions of Windows, Mac OS, and in Ubuntu 17.04 and later. It is checked by DNS APIs. I also verified that all applications first consult this cache before issuing an external DNS request. Specifically, the OS-wide DNS cache is implemented by introducing a DNS system service, running in a separate process isolated

from applications, that manages all the mappings. such as `getaddrinfo()`: only if the record is not found in the cache, does the DNS cache service perform a DNS request on behalf of the application. Thus, this client-side cache acts as the *de facto* first level of resolution. Recently, Ubuntu started using the OS-wide DNS cache by default, but earlier distributions including Ubuntu 16.04 LTS have implementations that have to be optionally installed and enabled by users.

The OS-wide DNS cache is stored in memory. I measured the size of the cache starting with an empty cache, warming the cache with a number of domain names (say x), and then resolving these names again while timing to see if the entry is being resolved from the cache. I keep increasing x until I start observing misses, which identifies the size of the cache. The results show that the cache size to be 2050, 5076, and 4094 entries in Windows, Mac OS, and Ubuntu Linux respectively. Furthermore, we find that the OS-wide DNS cache in all operating systems stores all types of records (A, AAAA, CNAME, PTR, RRSEG ...etc.) from only the *answer section* of DNS responses. Thus, Kaminsky's attack [180] relying on malicious records from the *additional section* does not apply to OS-wide DNS caches.

3.1.2 Threat Model

In this section, I describe the thread model of the attack. In such a model, I consider four entities below:

1. The victim client machine and its OS-wide DNS cache.

2. A legitimate resolver which acts as a DNS server for the client machine. The client

may connect to this resolver via a NAT device which has its own DNS cache.

3. The on-device malware, which is unprivileged and cannot tamper with other applications directly (I will instantiate this later in §3.2). This malware could be a malicious JavaScript running in the browser after an attacker browses a malicious/compromised website, or a malicious application downloaded to the user's phone.

4. The off-path attacker, who is capable of spoofing the IP address of the legitimate resolver. The malware and off-path attacker collaborate to poison the OS-wide DNS cache with a malicious mapping for a target domain name.

Note that the IP spoofing capability of the off-path attacker is commonly available in networks unless ingress filtering [183] is implemented [63]. A significant number of Internet Service Providers (ISPs) and networks do not implement ingress filtering and therefore an attacker connected to such a network can directly spoof IP addresses [64, 214, 122]. Also, an attacker on the same network does not have to pass through the ingress filter. Another scenario where the attack is possible is one where two machines are located in the same network (*e.g.,* enterprise or university network). It is common that a machine can spoof the IP address of the other since a packet does not pass through the ingress filter if it stays in the same network (I confirmed, after obtaining permission, this behavior in one enterprise and several university campuses).This threat model matches the threat model used in recent papers (e.g., off-path TCP injection attacks [234, 235]). In a sense, the attack can be considered a special type of local privilege escalation [108], where an unprivileged piece of a program can overwrite the OS-level DNS cache without authorization (with the help of an off-path attacker).

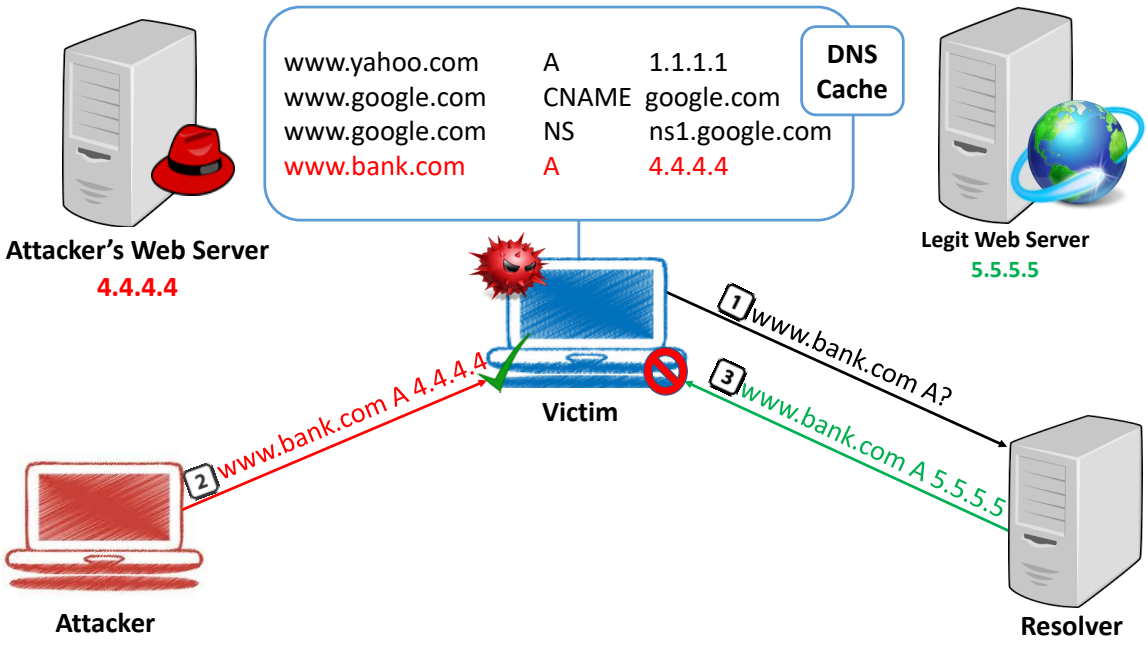


Figure 3.1: High Level Attack Overview

I also note another scenario where the attack applies: machines that have multiple users. One user can log into the machine, execute the attack causing the OS-wide DNS cache to be poisoned. When this user logs off, any subsequent user will be using the poisoned entries. I confirmed this attack scenario on several networks (with permission).

Another interesting threat model that is not considered exploits the browser level caches: since some browsers (*e.g.,* Firefox and Chrome) have their own DNS caches which are shared among different tabs/windows, the attack can also be applied using a malicious JavaScript piece of code assuming the success time is relatively small due to page visit time. In this scenario, the poisoning affects both the OS-wide and the browser-level DNS caches.

3.2 Attack Construction and Analysis

I first overview the attack at a high level and introduce possible attack scenarios. I then explain how to overcome a number of challenges needed to implement the attack, leading to a complete end-to-end attack under realistic conditions.

3.2.1 Attack Overview

The basic attack is overviewed in Figure 3.1. A malicious user-level program requests a DNS resolution for the target DNS domain (in this example, *www.bank.com*). The goal of the attacker is to poison the cached DNS entry such that it resolves to the IP address of an attacker server redirecting user connections targeting *www.bank.com* to go to the attacker's web server. Once the DNS request is sent, the attacker attempts to respond with fake responses. As it will be discussed later in Section §3.3, this is possible as the off-path attacker can start flooding spoofed responses even before the client initiates a DNS request. Alternatively, the malicious program on the victim can coordinate with the off-path attacker such that the responses are sent right after the query is issued. DNS resolvers accept the first correct response: a response with both a matching port number so that it is received correctly, and a matching TXID field; if a correct response from the attacker is received before the response from the DNS authoritative name server, the client simply accepts this response and caches it in its OS-wide DNS cache. The attacker's response uses a large TTL to ensure that the poisoned value remains in the cache. Any future connections from this machine to *www.bank.com* will redirect to the attacker's server.

3.2.2 Attack Scenarios

The attack requires a malicious user level program to execute on the victim machine. I consider two main scenarios for launching the attack.

• Public/shared machines. Such machines are commonly found in many places including universities, libraries, hotels, and stores. Any user who can log in to the machine can run a malicious program and collaborate with an off-path attacker to conduct the attack, poisoning the DNS cache, and leaving the machine to be used by victims. For all tested operating systems, the results show that the OS-wide DNS cache is in fact *shared across multiple users*. This means that a malicious user (*e.g.,* guest) capable of poisoning the OS-wide DNS cache can cause a different user (admin or guest) to also use the poisoned cache. Furthermore, for Windows, any user (including guest) can clear the cache directly without requiring admin privilege, so that the malware can clear legitimate entries to make room for poisoned entries. However, without admin privileges, the attacker may wait for TTL to expire, or evict the cache by requesting a large number of DNS resolutions to get older entries removed from the cache. I confirmed, after obtaining permission from the system administrators to conduct an experiment then clearing the cache, that shared public machines in four large universities are vulnerable to the attack.

• Malware. Applications downloaded from an App store, or malicious JavaScript on a website that is malicious or compromised, can also be used to launch the attack. In the public machine scenario, the attacker may need physical access to the machine. In this attack scenario, the victim unknowingly downloads a malicious application that launches the attack, without requiring physical access to the machine. On a smartphone, typically

the malware does not have access to remove entries from the DNS cache and must find a different way to clear already cached values before poisoning them.

As it will be shown later in Section §3.3, it is possible to launch attacks in both scenarios above.

3.2.3 Challenges and Detailed Attack Procedure

Source Port Reservation. In order to send spoofed responses, the off-path attacker must first obtain the DNS request's source IP address, source UDP port, destination IP address, and destination UDP port. DNS primarily use UDP protocol to transmit packets. TCP is used in cases where the data size of requests and responses exceeds 512 bytes; see RFC 1035 [219], but there is an extension to the DNS protocol that allows increasing payload size in UDP datagram; see RFC 6891 [106]. TCP also used in transferring Zone data because it is reliable. In the attack, I assume that all DNS traffic between victim, the source, and resolver, the destination, is over UDP. This is a valid assumption because DNS packets size is usually smaller than 512 bytes. IP addresses of the victim and resolver can be obtained easily using the unprivileged malware on the victim through standard OS interfaces, and the well-known destination UDP port for DNS requests is 53. The final challenge is to identify the source UDP port. As stated in RFC 6056 [198], the 16-bit dynamic port range of UDP is 49152 through 65536 which is typically used in Windows and Mac OS operating systems. However, we find that in Ubuntu Linux, the ephemeral port range is 32768 through 60999.

There exist many port selection algorithms, including Simple Port Randomization, Simple Hash-Based Port Selection (which is implemented by Linux), Double-Hash Port Selection, and Random-Increment Port Selection algorithms [198]. Many are proposed

specifically to defeat port predictions which means the port allocated each time for a new socket will appear to be random; however, the work in [155, 254] show that these algorithms are not efficient. Without documentation, it is unclear which algorithm is used in Windows or Mac OS. Nevertheless, after testing the source UDP port number selection of the DNS services in Windows and Mac OS, we find that they appear to be unpredictable. The old editions of Windows (e.g., Windows 2000 and Windows XP) assign ports to connections sequentially; however, the ports are completely random in all editions of Windows that I use in this work. Using my technique, the attacker can exactly specify which port can be used in the DNS connection.

A basic building block of the attack is the ability to predict or infer the source UDP port of a DNS request. Based on the measurements, I discover that surprisingly all operating systems I tested are permissive in terms of the number of simultaneously open sockets they allow to any program. This allows an application (*e.g.,* malware) to reserve all local port numbers but one so that the system DNS service will be forced to pick the one and only available port. Specifically, in Windows, any unprivileged application by default can open as many UDP sockets as desired and bind to a selected ephemeral port number. In Mac OS, there is a limit of the system resources consumption (which is 10240 file descriptors by default) but can be raised to a higher number (*e.g.,* 100,000) without root privileges [6]. Likewise, Ubuntu Linux has a default limit of 4096 file descriptors for each process which also can be raised to meet the attack requirements [2]. Even without raising the per-process limit, we can simply create a single application to fork multiple child processes (*e.g.,* 2 and 6 processes in Mac OS and Ubuntu Linux respectively) to be able to reserve the required

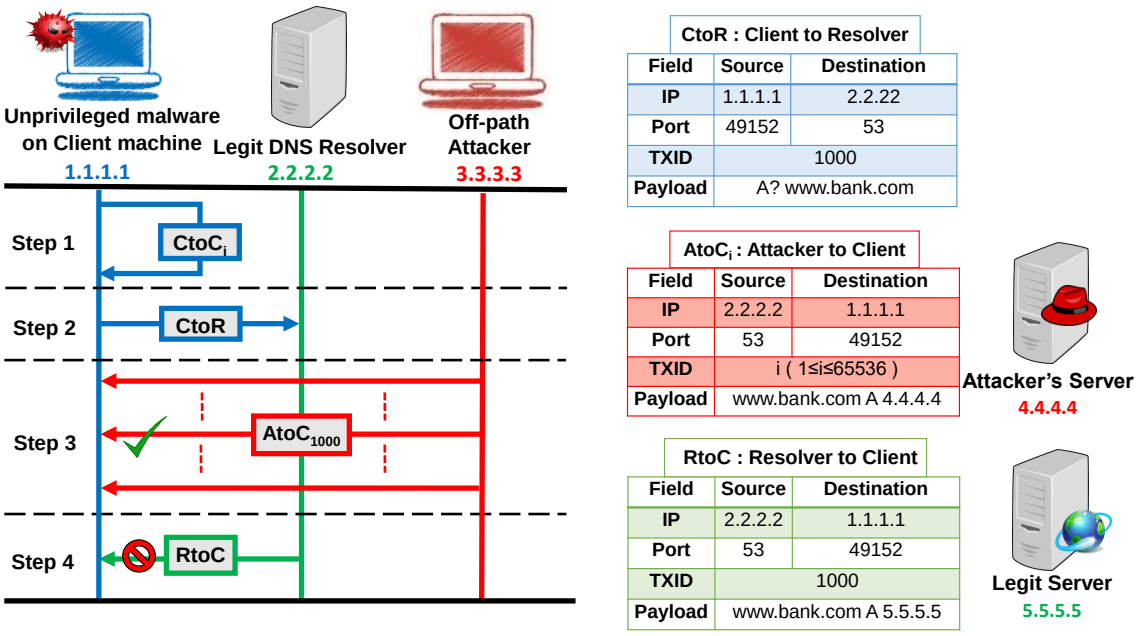
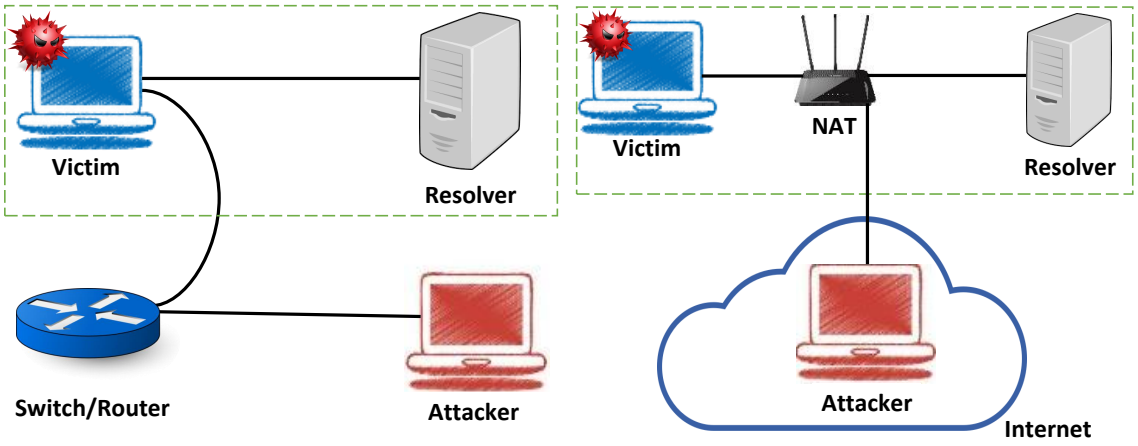


Figure 3.2: Design of client-side DNS cache poisoning attack

number of ports. I have verified that Android have similar behaviors to Mac OS and Ubuntu Linux.

Cache Poisoning. After the port reservation is done, the cache poisoning attack is started. The unprivileged malware on the client machine contacts the off-path attacker machine to coordinate the attack. I assume there is only one unoccupied UDP port (*e.g.*, port 49152) and the target domain name is *www.bank.com*.

The steps of the attack are illustrated in Figure 3.2. First, the malware on the client machine, at address 1.1.1.1, reserves all UDP ports except 49152. Second, the malware triggers a DNS query, denoted by *CtoR*, for the target domain name *www.bank.com* to the legitimate DNS resolver at address 2.2.2.2 with the source UDP port of 49152. The client OS randomly selects a TXID (say 1000). Third, the attacker repeatedly sends spoofed



(a) Client without NAT (b) Client Behind NAT

Figure 3.3: Attack Network Topologies

responses, denoted by $AtoC_1$, $AtoC_2$, ..., $AtoC_{65536}$, each with a different TXID field. If one of these responses contains the correct TXID (which is $AtoC_{1000}$), the cache can be poisoned to store a malicious IP address for *www.bank.com*.

Since TXID is a 16-bit field, a brute-force attack is possible even though the number of guesses seems large, especially given the attack may need to repeat over many trials (*i.e.,* by simply issuing `getaddrinfo()` calls). Finally, the resolver responds to the DNS query issued by the malware and sends an authentic response, denoted by *RtoC*. However, the response is ignored by the DNS system service since there is no longer a pending query. Otherwise, the authentic IP address will be cached and the attacker will repeat the attack starting from the second step. The steps are the same as if the client is behind NAT except that the attacker tries to poison the response of the NAT instead of the resolver.

3.3 Tailored Attack Strategies and Analysis

In this section, I consider OS-specific attack strategies depending on whether the client is behind a NAT router and based on the different operating systems. I organize the discussion into first attacks without considering NAT, and then the situation where the client is behind a NAT router.

3.3.1 Basic Attack Scenarios (Without NAT)

In this scenario, I assume the attacks are against networks without a NAT device on the route between the off-path attacker and the victim client (Figure 3.3-a). In other words, the communications between the client, resolver, and attacker are direct and not translated. This setting is common; many networks do not use NAT. Moreover, the attacker can be a legitimate user on the same organizational network with the victim (*i.e.,* on the same private network) or on a public WiFi wireless network in the same organization, or the attacker may control a compromised host (*e.g.,* using a malware) on the same LAN. If the attacker and victim are in the same wired or wireless LAN, then a NAT will not be traversed.

To succeed in the attack, a malicious response with a matching TXID has to arrive before the real response. On paper, this may be a challenging task as there is really only an attack window equivalent of a round-trip time between the client and resolver. Even with a high bandwidth, the attack has a fairly low probability of success. Assuming an RTT of 5msec (client and resolver are close), and an attack bandwidth of 10,000 spoofed responses per second, only 50 packets will be received before the authentic response, leading to a

success probability of $\frac{50}{65536} = 0.076\%$. To make matters worse, once the authentic DNS response is cached, one may need to wait for the entry to timeout before being able to launch the next round of attack on the same domain. However, the measurements show that *specific OS implementations* and *real-world TTL/network latency* make the proposed attack highly feasible. I next detail three OSes: Windows, Mac OS, and Ubuntu Linux.

Windows

I observed an interesting behavior when testing the attack on Windows. In particular, we find that the DNS system service (*e.g.,* `getaddrinfo()`) retransmits the request as soon as it receives a (spoofed) response with an incorrect TXID, and it simply aborts after five failed retransmissions. I reverse engineered the DNS system service binary (`dnsapi.dll`) and found that it indeed has a simple loop with `select()` for up to five times. Thus, if the legitimate response is preceded by five spoofed responses, it will not be accepted (since the request resets, and the new request has a different TXID). Although this also means that the attacker only gets five chances to guess the TXIDs before each invocation of `getaddrinfo()`, this has a smaller effect since the attacker is guessing the TXID, and the chance of guessing it correctly does not depend on the actual TXID; the strategy works across multiple invocations as the attacker keeps invoking `getaddrinfo()`. This makes the attack much easier as the authentic response can hardly be cached when attack traffic is present. Meanwhile, although the attacker only gets five chances to guess the TXIDs before each invocation of `getaddrinfo()`, the attacker can always call `getaddrinfo()` multiple times until the attack succeeds.

Table 3.1: TTL for Alexa top 10 global websites

Rank	Website	TTL (seconds)	Rank	Website	TTL (seconds)
1	Google.com	60	6	Reddit.com	30
2	Youtube.com	60	7	Yahoo.com	60
3	Facebook.com	60	8	Google.co.in	60
4	Baidu.com	300	9	Qq.com	20
5	Wikipedia.org	600	10	Taobao.com	180

Mac OS

For Mac OS, unlike Window's 5 response limit behavior, Mac's DNS system service continues to accept DNS responses until receiving a response with a matching TXID. If none is found before the timeout, it will simply retransmit and continue to wait for the correct response to arrive. This means that the attack window is only a single round-trip time before the legitimate response is received. This makes the attack window somewhat limited: if we can send 10K spoofed messages per second, and $RTT = 5msec$, then the number of chances is 50. Also, if the current attempt fails, we have to wait for the cached authentic response to timeout before we can retry (recall that this is not the case for Windows whose cache can be cleared even by a non-administrator user, *e.g.,* guest). While this slows down the attack, the results show that the TTL values are typically short. Table 3.1 shows the TTLs of the top 10 global websites based on Alexa [17]. We find that 58%, 27%, and 19% of the Alexa top 500 global websites have a TTL value less than or equal to 60sec, 30sec, and 20sec, respectively. Importantly, since at the start of every attempt, the value has expired in all the caches, this

Table 3.2: The average RTT (in milliseconds) for Alexa top 500 global websites from different vantage points

DNS Server	VP1	VP2	VP3	VP4	Google Cloud	EC2
Google DNS	61	141	62	57	36	50
Quad9	104	191	109	103	77	70
OpenDNS	70	156	68	57	60	55
Norton	34	121	38	65	293	35
Comodo	164	185	130	80	82	94
Level3	58	136	77	71	58	58

gives us *an RTT window of the full resolution through the DNS system*, traversing several resolvers, which can be in the tens if not hundreds of msecs. Thus, the attacker gets a larger number of guesses before the authentic response is received. To confirm the larger RTT time, I tested the RTTs from 6 different vantage points (VPs) including EC2 and Google cloud servers and 4 large universities to 6 public DNS servers. The dataset is the top 500 global websites based on Alexa [17]. I used `nslookup` to forcibly send a DNS query to open DNS servers (*e.g.,* 8.8.8.8 for Google public DNS server) regardless of its caching status. As shown in Table 3.2, the RTTs are indeed relatively high.

Ubuntu Linux

The first release of Ubuntu Linux that supports OS-wide DNS caching is Ubuntu 17.04 that uses `systemd-resolved` as the default system DNS service [186]. The results

show that the behavior is almost identical to Mac OS. The main difference appears to be that when all UDP ports are reserved on Ubuntu Linux, DNS queries can still be sent to the resolver using random source TCP ports. To overcome this problem, I use the same port reservation technique in Section §3.2.3 to reserve *all* TCP ports and force the DNS service to use the one and only available UDP port for all outgoing queries. Ubuntu Linux kernel does not quite follow the rules standardized in RFC 6056 [198] regarding the ephemeral port range for both UDP and TCP as already discussed in §3.1.3. In addition, when all UDP ports are reserved, we find that DNS queries can still be sent to the resolver using random source TCP ports. To overcome this problem, the same port reservation technique in §3.2.3 can be used to reserve *all* TCP ports and force the DNS service to use the one and only available UDP port for all outgoing queries. For completeness, I also measured the behavior of `dnsmasq`, a popular DNS/DHCP software [15], for other Linux distributions which behave very much the same way as Mac OS and Ubuntu. In other words, the attack is also effective on any Linux-based system running this DNS API.

To further improve the attack time for MacOS and Linux, we can try to accelerate flushing of the DNS entry from the cache, for example, by filling the cache with new requests. In addition, I developed a strategy to interfere with the resolver's ability to respond to the DNS requests, which provides a larger time window for the attacker to operate. Specifically, the attacker can launch a DoS attack that floods the external resolver with spoofed DNS requests for domain names different than the one the attacker targets (*e.g.,* www.bank1.com, www.bank2.com, ... etc) to fill its socket buffer. The idea is that since we can predict the source UDP port of the DNS request, all spoofed DNS requests will target the same exact

socket buffer to which the real DNS request will also go. If the legitimate DNS request is received while the socket buffer is full, it is dropped and no response is sent back. For example, I configured my own DNS server which runs Ubuntu Linux 14.04 LTS, and I measured the per-socket buffer for the OS and found that the default per-socket buffer size is 17KB which can hold only $\simeq$ 300 DNS packets.

3.3.2 Client behind NAT Attacks

As shown in Figure 3.3-b, a NAT allows clients on a private network to connect to the Internet by remapping their private addresses to its own IP address and using port numbers to keep track of the mapping of internal connections to external ones [257]. In more detail, the benefits of NAT include: (1) allowing administrators to assign private IP addresses to machines without having globally assigned addresses; and (2) hiding the internal structure of a private network from the outside. Specifically, the NAT router's external facing IP is a valid static IP address. Since the machines behind the NAT do not have a globally resolvable address, the NAT translates their internal address on packets they send to the Internet to its external address [257]. To be able to resolve incoming packets (which all use the IP address of the NAT router) to the correct internal machine, NAT uses ports to keep track of connections belonging to different machines. In particular, the router also assigns an external source port which could be different than the source port selected by the operating system of the client who initiated the connection and keeps the mapping between the ports in a mapping table. When an incoming packet comes to a particular port, the NAT router replaces the destination IP and port number with the IP address and port number of the internal port. With respect to the attack, since the attacker

does not generally know the external port assigned to the DNS query, NAT increases the entropy. Fortunately, we find that under several popular settings, we can derive the external port allowing the attack to proceed. I tested three NAT devices (Linksys WRT3200ACM, Netgear WNDR4500 v3, and Netgear WGR614 v9). The measurements show that the NAT's port translation behaviors for DNS sessions depends on how DNS is configured on the client and on the NAT:

1. **Both Client and NAT Use DHCP.** By default, DHCP configures both the client and NAT to use the local DNS server. In this scenario, to use the default settings of the client's ISP, we find that the NAT translates the source UDP port of the client to a random port each time the client contacts the local DNS server.

2. **Client Manually Configures DNS.** When a client changes the DNS settings to an alternative DNS server (*e.g.,* 8.8.8.8 for Google public DNS), the NAT preserves the source port of UDP sessions to that DNS server. Using an open resolver is becoming increasingly common: a recent study shows that 12.70% of a sample of size $\simeq$ 735 million machines around the world use Google Public DNS server [8]. Moreover, there are other Open DNS servers that are popular (*e.g.,* Quad9 (9.9.9.9), OpenDNS (208.67.222.222), Norton (199.85.126.10), Comodo (8.26.56.26), and Level3 (209.244.0.3)).

3. **NAT Manually Configures DNS.** If the client uses DHCP but NAT uses a manually configured DNS server, the NAT assigns a fixed source port to all DNS sessions to that DNS server.

4. **Both Client and NAT Manually Configure DNS.** In this case, similar to Case 2, the NAT preserves the client's source UDP port.

The attack can be easily carried out in all above scenarios except Case 1: in Cases 2, 3, and 4, the external source UDP port is known to the attacker, and the attack can be easily modified to attack a client behind NAT. For Case 1, I can bootstrap the attack by using principles proposed by Herzberg et al. [155, 143] to reserve the source port. In particular, this attack creates a large number of connections to attempt to reserve all the external source UDP ports of the NAT router. If only one port is left, the NAT router is forced to use it and the attacker no longer has to guess the port number.

3.4 Evaluation

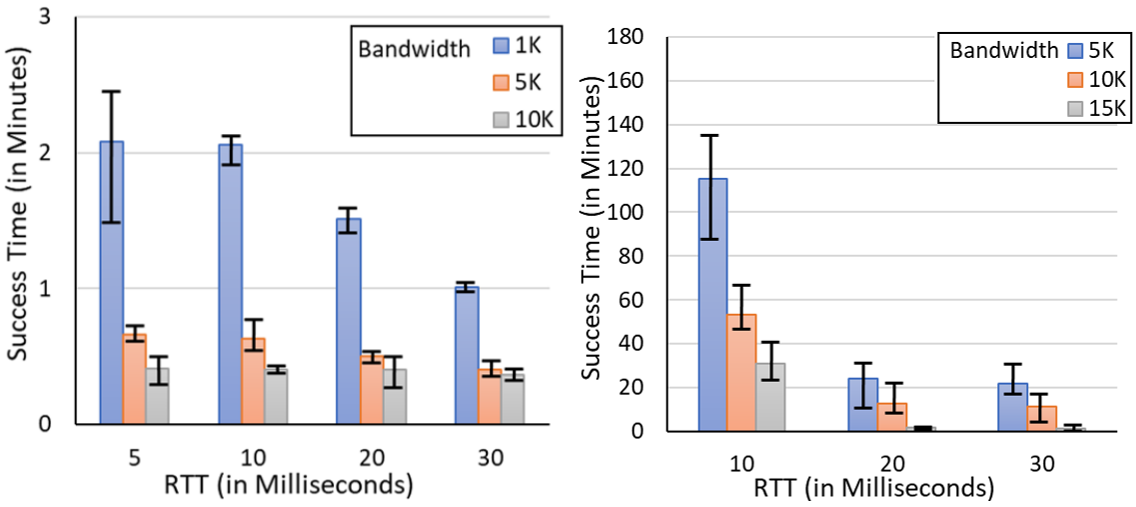
In this section, I conduct measurements on real systems to assess the effectiveness of the end to end attack in realistic settings for all the attack scenarios. The attack is successful within reasonable time against all these operating systems confirming that this is an extremely dangerous vulnerability.

Client Platform. I conduct the experiments on client machines running Microsoft Windows 7, 8.1, and 10, MacOS Sierra, and Ubuntu 17.04 which all support an OS-wide DNS cache. I note that most other Linux distributions, including Ubuntu versions prior to 17.04 do not enable an OS-wide cache by default, although DNS cache implementations could be installed for example by enabling `dnsmasq` [15]. In addition, the malware runs in user space; however, since some browsers (*e.g.,* Firefox and Chrome) have their own DNS caches which are shared among different tabs/windows, the attack can also be applied using JavaScript

assuming the success time is relatively small due to page visit time. In this scenario, the poisoning affects both the OS-wide and the browser-level DNS caches.

Network Configuration. Two different network topologies are used mirroring those shown in Figure 3.3-a (without NAT) and Figure 3.3-b (with NAT). For the NAT-based attacks, I configure an internal DNS resolver, using the BIND name server software (BIND9) on Ubuntu 14.04 LTS; (which I denote by `U14`). The NAT acts as port-preserving, attempting to allocate the same outside port as the inside port, (case 2 as described in §3.3). I also verified the attack on fixed-port NATs (case 3). In fact, this attack is easier since there is no need to reserve the ports on the client machine to force a particular port to be used since the NAT assigns a fixed-port for each client.

All nodes are connected to the network using Ethernet cables. The network bandwidth between all nodes is 1Gbps. In a real attack environment, there may not be such a high bandwidth and therefore intentionally we can limit the attacker's throughput using the Linux Traffic Control utility `tc` [76]. TC allows us to configure the Linux packet scheduler to simulate lower bandwidth connections and also control the effective RTT to the attacker. Note that I will use the number of spoofed packets the attacker can send per second to represent the bandwidth. For example, giving a packet size of 97Bytes, we say the bandwidth is 5K/s instead of using 3.88Mbps. In addition, since the off-path attacker knows which source UDP port the DNS request will use, she can start flooding the client/NAT with spoofed responses even before the client initiates the DNS request. With this strategy implemented, the attacker has a full round-trip time window to try and guess the correct TXID.



(a) Windows (b) Ubuntu Linux

Figure 3.4: Median time to an attack success without NAT

Experimental Details. I show the measurement results of the attack against all operating systems. Each data point is generated by 50 repeated experiments. Given the large dispersion in the time to succeed (due to the geometric distribution of the number of tries until success of the attack), it is estimated that bounding the confidence interval of the mean requires many thousand experiments in several of the scenarios. This is not feasible since some experiments take on the order of hours. Thus, instead of using the mean, the median is used of 50 experiments for each point since the median is more robust to outliers. I also calculate the 95% confidence intervals of the medians and show those on the figures. Note that confidence intervals for medians are not symmetric around the median. The formula in [40] is used to calculate the *rank order* of the upper and lower bounds instead of their actual values. For one representative case, the histogram of the time to succeed for individual experiments is shown to provide insight into the distribution of the time to success of the attack for the different operating systems.

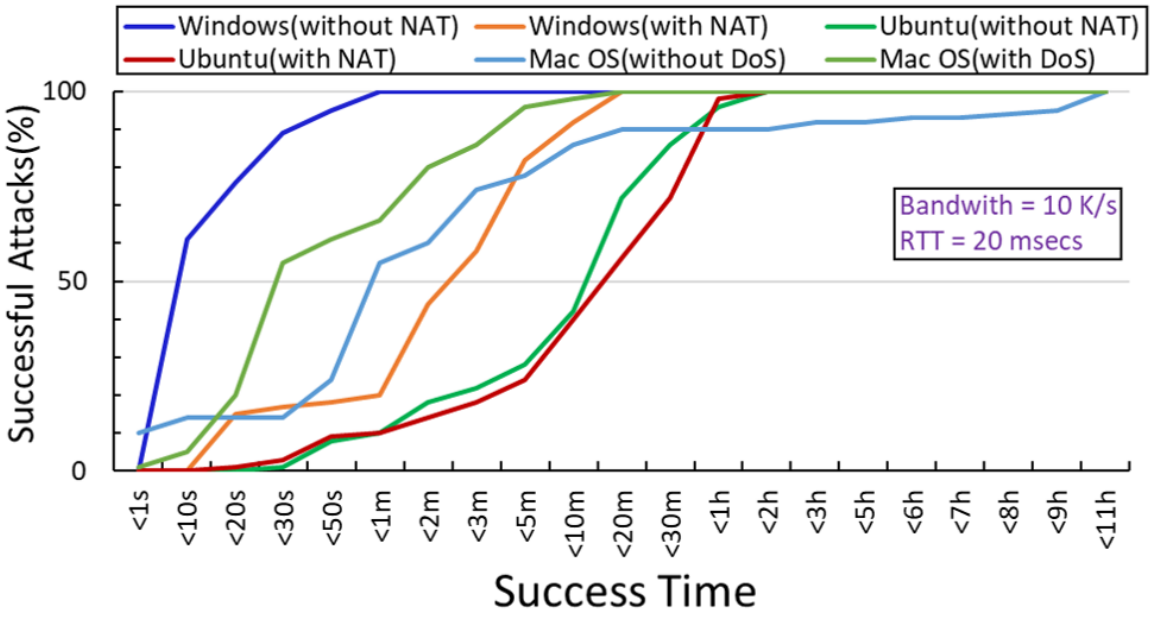


Figure 3.5: Success Time Statistics

3.4.1 Results of attacks without considering NAT

In this section, I analyze and report the evaluation results of the attack when the client is not behind a NAT. The experiments are conducted on all three operating systems.

Windows

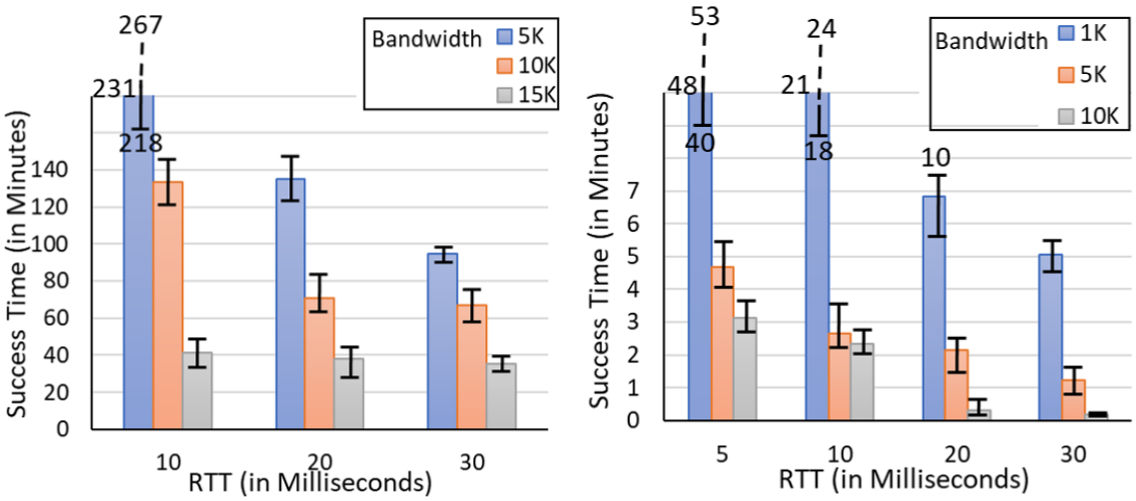
Figure 3.4-a shows the measurement results of the attack against Windows. The time to succeed is small, one the order of 10s of seconds, in most configurations as shown in Figure 3.5, even though the number of trials to succeed is typically in the tens of thousands. The fast success time is possible because `getaddrinfo()` can be invoked extremely quickly—most of the time `getaddrinfo()` returns in less than 2 milliseconds after 5 spoofed responses are received. As I mentioned in Section §3.3.1, in theory, the two metrics should not be dependent on RTT and bandwidth so long as at least 5 spoofed responses always arrive

before the authentic response. In practice, however, it is observed that rarely ($< 0.1\%$ of the trials) the authentic response arrives within the first 5 responses due to network jitter, especially as RTT gets larger. In addition, I observe that in some cases the authentic responses are sent in bursts (because of the DNS query gets retransmitted quickly under attack, forcing the resolver to respond with authentic responses quickly as well). Such authentic responses in a burst can sometimes take up the 5 spots of the next invocation of `getaddrinfo()`, thereby reducing the attacker's chances. For example, the `getaddrinfo()` may receive two authentic responses from the previous invocation (both with the same TXID), and it will only process the next three spoofed responses. Furthermore, the results show that the likelihood of a successful attack depends significantly on the two factors (RTT and bandwidth): the more resources the attacker has the faster the attack can succeed.

Ubuntu Linux

Figure 3.4-b shows the attack against Ubuntu Since the malware does not have access to flush the DNS cache (as in the Windows attacks), I consider an attack on a website with a TTL of 30 seconds (which is in the common range experimentally measured in Section §3.3.1). Thus, for each failed trial, the attacker waits for 30 seconds before the next trial. As shown in the figure, the attack time-to-success also depends on RTT and bandwidth.

Note that in this scenario (as well as MacOS), every trial waits until the TTL expires before it is initiated. Thus, the response to the query will miss the caches and likely go through the full resolution step, or hit a cache upstream, resulting in a large delay until the authentic response is received. This delay is measured to be on average 92msec (See



(a) Mac OS (without DoS)

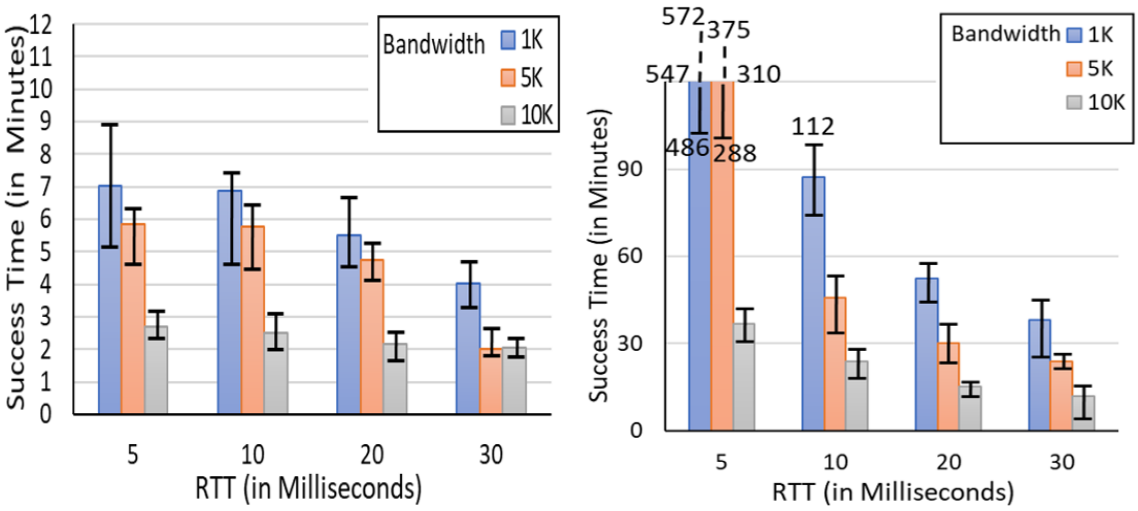
(b) Mac OS (with DoS)

Figure 3.6: Median time to success on Mac OS without NAT

Table 3.2). Thus, for the attack on Ubuntu and Mac OS, I start with RTT of 10msec (which is still quite conservative). The cases where $bandwidth = 1K/sec$ and $RTT = 5msecs$ are excluded because it is extremely challenging to a get a successful attack. For instance, I ran an experiment for 1 day using $bandwidth = 15K/sec$ and $RTT = 5msecs$ and I could never succeed. Furthermore, in an experiment where $bandwidth = 1K/sec$ and $RTT = 30msecs$, I got a successful attack after 15Hrs. Likewise, I excluded the same settings on Mac OS since this behavior is encountered while conducting the experiments.

Mac OS

Figure 3.6-a shows the results of the attack against Mac OS. Similar to the Linux attacks, I assume a TTL of $30secs$ for the resource record in the OS-wide DNS cache. As we can see, overall time to success improves with increased bandwidth or RTT.



(a) Windows

(b) Ubuntu Linux

Figure 3.7: Median time to success when client is behind NAT

Although the time to success is not prohibitive, the attack is slow. Thus, as I discussed earlier, I consider a case where the attacker also performs a DoS attack against the resolver with 15K packets per second (could be from the same attack machine or an external one). For ethical reasons, I set up my own resolver for this experiment so that I do not carry out a DoS attack on part of the DNS infrastructure. The time to success improves dramatically, to a few minutes or lower under most configurations, as shown in Figure 3.6-b. The results show that more than 60% of the attacks succeeded in less than a minute for the configuration shown in Figure 3.5). With the DoS attack, the average number of trials to succeed drops dramatically to less than 5, since most of the real DNS requests are dropped by the resolver, providing a larger window to spoof responses.

3.4.2 Client behind NAT

Having established how the attack can bypass a NAT, the remaining experiments focus only on the scenario without NAT. In this section, I show the evaluation results of the attack against two operating systems: Windows and Ubuntu Linux (MacOS behaved similar to Linux). To avoid redundancy, I omit the experiments results for Mac OS since the systems characteristics are highly similar to those in Ubuntu Linux.

Figure 3.7-a and Figure 3.7-b show the attack results against Windows and Ubuntu Linux when the client is behind a NAT (Mac OS behaved similar to Linux), respectively. As shown in the figures, in addition to RTT and bandwidth, the success time is also affected when I add NAT to the network topology.

As stated earlier, network congestion is the main cause of packet loss which makes the attacks more challenging. The main factor affecting the packet loss rate are packet buffers within the network software stack. Specifically, when the socket receive queue of the NAT receives packets at a rate that exceeds the router's renaming and forwarding capacity, the NAT starts dropping packets. In my case, since the attacker sends malicious response packets aggressively, the results show that a fraction of these packets is not delivered to the client machine. Moreover, on the client side, before a response packet gets processed by the DNS API `getaddrinfo()`, it is transmitted through different layers of network queues until it reaches the OS-wide UDP socket receive queue. Since the IP stack that is filling the queue and the network driver that is draining the queue run asynchronously, there is a high probability that the packet might be dropped before it is even processed by the DNS API which can cause starvation. This problem is also apparent in Mac attack results as

shown in Figure 3.6-a since the queue size is small (only 9216 Bytes) compared to its size in Ubuntu Linux (212992 bytes).

3.4.3 Analytical Model and Validation

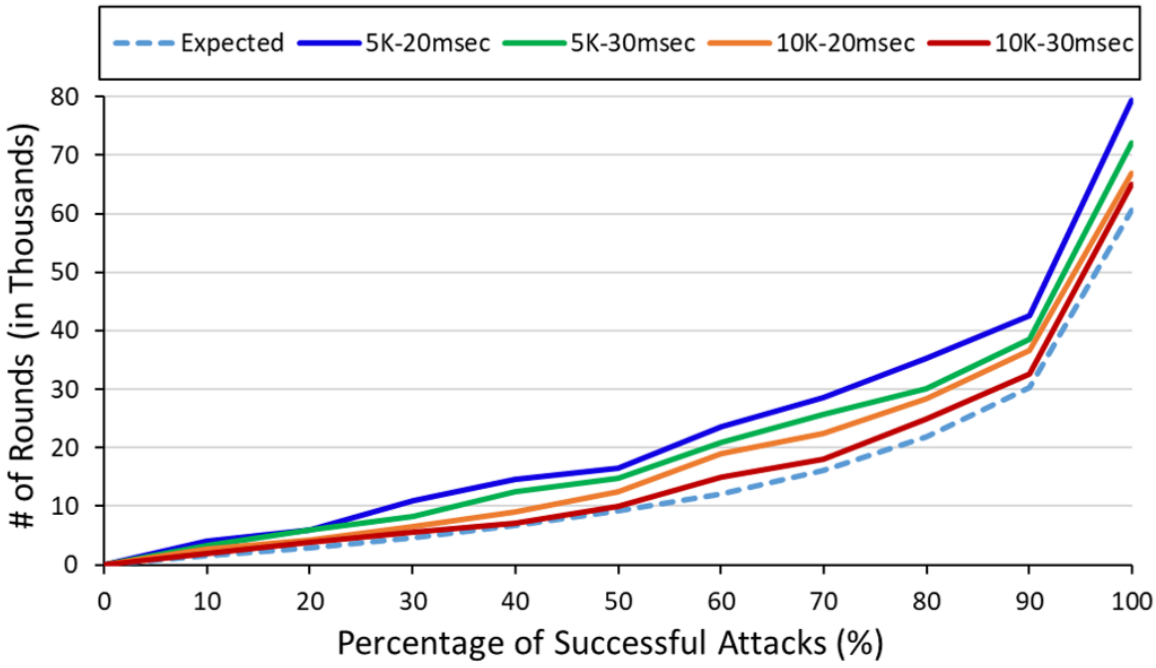
To provide deeper understanding on the attack behavior, I model the probability of success as a geometric process with the constraints imposed by each Operating System (e.g., the 5 response limit in Windows). These results are compared to measurement results regarding the number of rounds before a successful attack. The analytical model is approximate because it does not capture network effects or other effects such as the legitimate responses competing with the attack responses particularly in Windows.

Analytical Model: For a single spoofed reply, the probability of making a correct guess for the TXID is 1 out of $2^{16} = 65,536$ (2^{16} is the TXID field range). The probability of failure of the attack in response to a single invocation of DNS API (*e.g.,* `getaddrinfo()`) is determined as follows: $P(failure) = \frac{2^{16}-z}{2^{16}}$, where z is the number of guesses up to a maximum of 2^{16}. We find that the value of z differs based on the DNS API `getaddrinfo()` of the different operating systems. For Windows, z is 5 since the `getaddrinfo()` fails after 5 tries, leading to a success rate of 0.0076% for each try. The overall success of the attack over x invocations of `getaddrinfo()` is determined by the cumulative distribution function of the geometric distribution and is: $P(X \leq x) = 1 - P(failure)^x$.

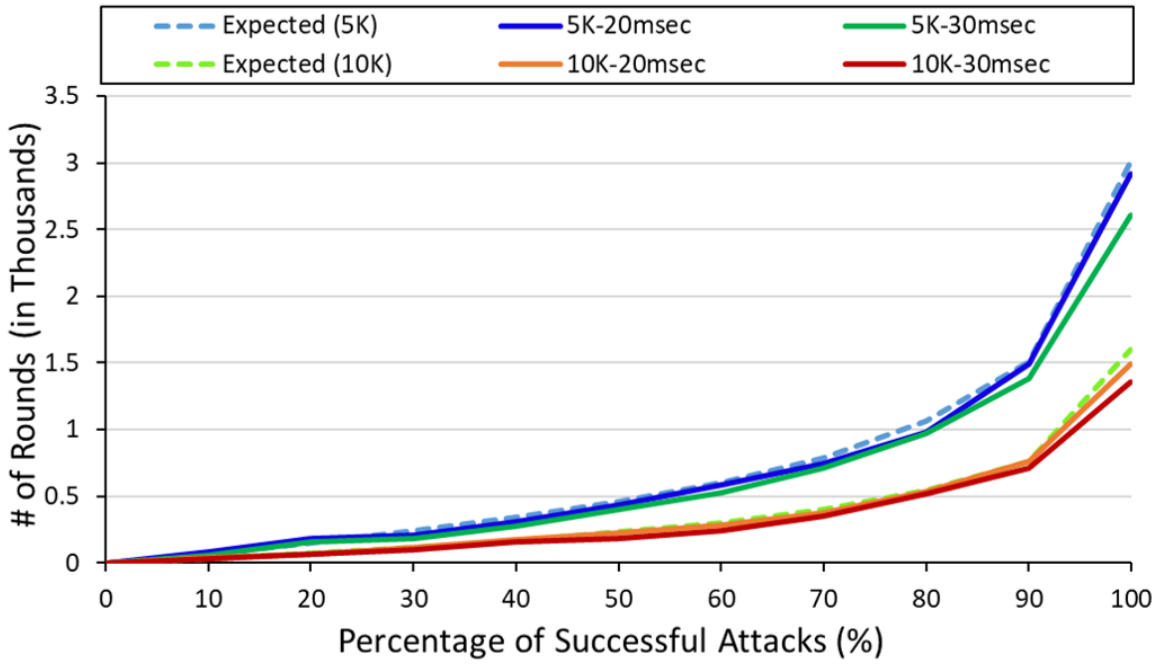
The average number of rounds before a success is determined by the geometric distribution and is $\frac{1}{1-P(failure)}$. For Windows, this comes out to be a little over 13000 tries. Since Windows does not rate limit `getaddrinfo()`, each attempt takes as short as 2msec, which means that the average time to succeed will be as short as $2msec * 13K = 26$ seconds.

For Mac and Linux, z is determined by the bandwidth of the off-path attacker to the client and the RTT of the DNS resolution (which determines when the legitimate response is received). For example, pessimistically assuming a low RTT of 5msec, and an attacker bandwidth of 10K spoofed messages per second, z is 50 packets, leading to a success rate of approximately 0.076% (about 10 times that of Windows), leading to the average number of rounds before the success of just over 1300. However, since each retry has to wait for the value to expire from the DNS cache (*e.g.,* 30 seconds), the time until success can be long (close to 11 hours for this example). Note that this is highly pessimistic since the uncached RTT is much higher than 5msec. For instance, using the average value in Table 3.2 which is 92msec, z is 920 packets, leading to a success rate of approximately 1.4% and an average number of rounds before the success of 70. In this case, the time until success is dropped to 35 minutes. The analysis does not take into account network effects (e.g., dropped packets due to overrunning buffers) which is found to have significant (even dominating) effects in some scenarios.

Model Validation: To validate the analytical model, I conducted a measurement study of the attack on Windows and Ubuntu Operating Systems. For each Operating System, I show measurement results for different cases (*i.e.,* when bandwidth from attacker to client is 5K and 10K and when RTT between DNS resolver and client is 20msec and 30msec); I use these parameters in the analytical model. For each experiment, 200 successful attacks are conducted to provide the measurement data. A histogram of the number of rounds to succeed for individual experiments is shown to provide and compare that to the predicted percentage of successful attacks at each point.



(a) Windows



(b) Ubuntu

Figure 3.8: Experimental and Analytical Results (# of Rounds)

Fig. 3.8-a and Fig. 3.8-b show that the results of the analytical model conform with the experimental data. For Windows, as shown in Fig. 3.8-a, we find that we need tens of thousands of rounds to obtain a probability of success higher than 50%. The analytical model underestimates the number of required rounds due to network effects such as congestion when the attack traffic is present, which especially affect Windows due to the 5 packet response limit for each request. For Ubuntu, as shown in Fig. 3.8-b, we see that the number of rounds in the measurements follow the analytical model closely. The model follows the analytical model more closely because the rate of retries is slower (because we have to wait for the DNS cache value to expire after the legitimate response is received), leading to less congestion.

3.4.4 Discussion

I have shown that the client side attack is practical against a range of operating systems: Micrososft Windows, Apple macOS, and Linux Ubuntu. The results also show that the attack can succeed with/without the presence of NAT. This dangerous attack is not mitigated by proposed solutions against traditional DNS poisoning attacks. The attack also shows that protecting the core DNS system alone (the name servers and resolvers) does not prevent DNS poisoning attacks that occur at the clients. Left unprotected, these caches can completely subvert DNS security. Indeed, in the measurement and analysis, it is clear that a system is only as secure as its weakest link — even though the DNS traffic between the resolver to authentic name servers can be secure, I show that the weak link can be between the client and the resolver. In this case, the attack target has changed from the DNS resolver to clients, the assumption has changed.

Existing defenses fail to protect against this vulnerability for two reasons: (1) the clients are not part of the core DNS systems; and (2) the threat model, with a malware executing on the client system, provides the malware information that is assumed to be unknown in traditional threat models. Specifically, the port randomization is no longer effective as they can be "derandomized" by the unprivileged malware. In addition, other defenses such as 0x20 [104] and even DNSSEC [46] currently are applied to only the resolvers instead of clients. Recent defenses such as DNS over HTTPS [163], DNS over TLS [165], and DNScrypt [109], have been proposed primarily to preserve the privacy of DNS traffic. These proposals can also have the side effect of hardening DNS traffic against injection attacks. Although there are standardization efforts behind these proposals, they are not yet widely deployed. It is also unclear if they will be used only by browsers, and whether their implementations will be secure.

3.5 Concluding Remarks

This research direction identifies and evaluates a new client-side OS-wide DNS cache poisoning attack against Windows, Mac OS, and Linux operating systems. The attack targets the OS level DNS cache, a client-side cache supported by most modern operating systems. I tailor the attacks to work against each OS individually taking into account the specifics of each implementation. The attack succeeds in as little as tens of seconds under realistic conditions. I also build an analytical model of the expected number of rounds for the attack to succeed for both Windows and Ubuntu and validate the model against the experimental observations. In the next chapter, I propose a defense that requires only a

patch of the OS of the client device and show that it can mitigate the attack. I hope that the lessons learned can help improve the future design and implementation of DNS and even other OS-wide caching systems.

Chapter 4

Ad Hoc System-Level Defense

The research in this dissertation also contributes a lightweight client-side defense strategy to mitigate this vulnerability. I detail the defense proposal against this type of poisoning attack and show that the attack can be fully mitigated using different strategies which can be deployed immediately through an OS patch. In particular, although the attack discussed in Chapter §3 significantly reduces the entropy by removing the uncertainty regarding the source UDP port number, it still relies on guessing the transaction ID (TXID) field, which has a range of 2^{16}. The attack is successful because today's DNS clients simply discard illegal DNS responses that do not match the port and TXID of a pending request. Here I focus on client side defenses since they require only changes to the client software, and can therefore be deployed immediately through an OS patch. The defense first detects an attack using its unique signature (multiple DNS replies with wrong TXID), and then takes corresponding measures for mitigation.

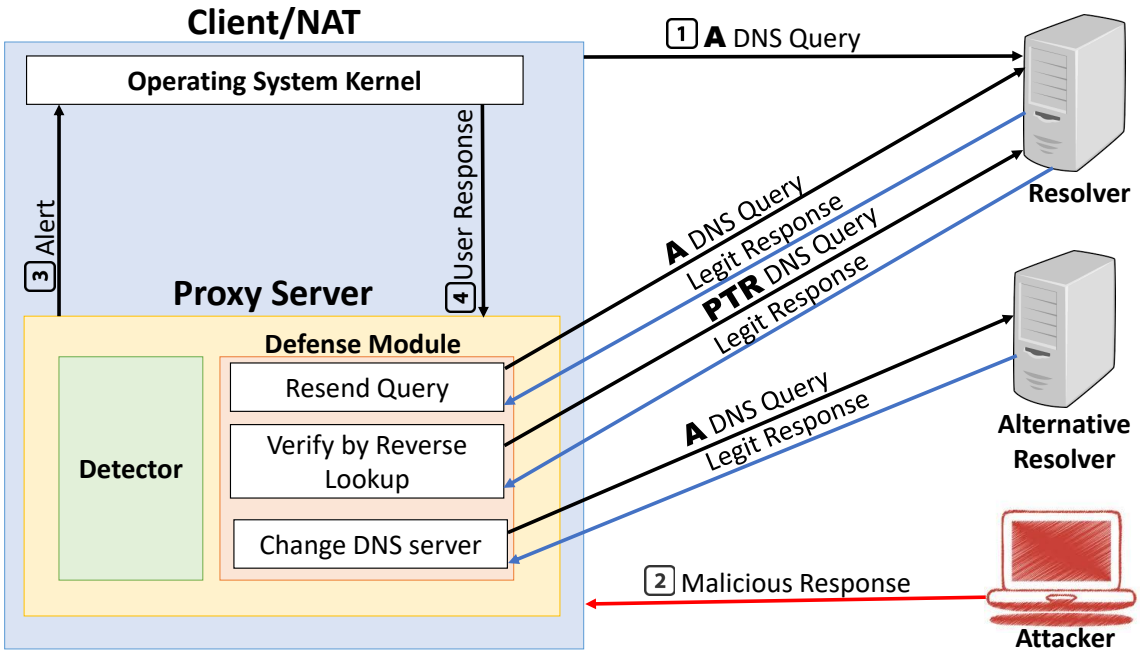


Figure 4.1: Defense Model

4.1 Attack Detection

An overview of the proposed defense is presented in Figure 4.1. The detection module is implemented as a proxy server for flexibility of evaluation against different Operating Systems; in a real deployment it can be integrated directly with the DNS service. The module sniffs on UDP port 53 for all DNS traffic. After a DNS query is sent in (1), an attack is detected in step (2) when observing responses with incorrect TXID. Once an attack is detected, we send a signal to the OS in step 3 on the figure, triggering mitigations. We can optionally implement a local sanity check on the system to see if there is suspicious behavior; for example, we check if there are processes reserving a suspiciously large number of ports and clean those up. This step is optional because it may lead to collateral damage if a legitimate process uses a large number of ports.

At the DNS service level (step 4), different reaction strategies are possible (shown under the defense module in the figure). A paranoid strategy that simply drops all responses can lead to Denial of Service as the attacker prevents legitimate responses; thus, it is important to find reaction strategies that can protect the system, but also allow to continue to function. We describe the strategies in the remainder of this section.

4.2 Attack Mitigation Strategies

The proposed mitigation strategies leverage the optional step of releasing the port reservations of the suspected malicious processes. They are three fold:

1. Repeat query: The first mitigation strategy is to resend the same DNS query. Specifically, the defense module uses a DNS API (*e.g.,* `getaddrinfo()`) to send a new DNS request for the same domain name to the same DNS resolver (the OS uses a random UDP port). If the ports are released, there is no way that the attacker can learn the source UDP port. Without releasing the ports, its unlikely for this defense to be effective (essentially this becomes similar to the Windows implementation where the request resets after 5 erroneous replies).

2. Verify Response to reverse lookup: The module verifies the IP address in the response DNS packet by sending a Pointer (PTR) DNS query. This query type is used to resolve an IP address to a FQDN. If the FQDN and the query name of the pending query do not match, then we can be more confident that an on-going attack is present (we notice that the PTR reply itself can also be spoofed though, but the probability of success on both the reply to both the A and PTR queries is very small). We probed Alexa top 500

global websites by sending PTR DNS requests, and found that PTR queries have moderate support on today's Internet. We found that nearly 52% of the sites have PTR records and only 24% return an FQDN that contains the domain name in the query. Thus, this defense will work only opportunistically, or with support from websites and public servers to maintain accurate PTR resolutions.

3. Switch DNS server: The last class of defense we explore is to choose an alternative DNS server, accomplishing the resolution before the malware can inform the off-path attacker of the change of server.

On top of these mechanisms, we suggest a less intrusive action (but potentially risky action) of accepting the returned DNS response with the correct TXID but not cache it until/unless it is verified. By not caching the DNS response, at least we limit the damage by making sure that the DNS cache is not poisoned, allowing the cache to be used only when no foul play is suspected. In our scenario with the malware, the malware receives the spoofed response, but is unable to poison the entry for any other applications.

4.3 Empirical Defense Evaluation

We tested the above defense on a number of operating systems including: Microsoft Windows 10, MacOS Sierra, and Ubuntu 17.04 without a NAT box. However, we also tested the defense on Linksys WRT3200ACM router running DD-WRT firmware and confirmed that the defense works efficiently. The attacker's machine runs Ubuntu Linux 16.04 LTS. We used RTT = 20msec and the attacker's throughput is 10K/s (*i.e.,* 7.4Mbps when malicious packet size is 97Bytes).

Table 4.1: Defense Experiments

Operating System	# of Attacks (without defense)	# of Attacks (with defense)
Windows	1705	0
Ubuntu Linux	152	0
Mac OS	18	0

We ran the attack continuously over a twenty-four hour period collecting results for the cases with and without the defense. As shown in Table 4.1, the defense mitigates attacks (*i.e.,* the success probability of the attack is 0%). On the other hand, we recorded 1705, 152, and 18 successful attacks on Windows, Ubuntu Linux, and Mac OS, respectively, when the defense is not deployed.

4.4 Other Recommendations

Additional mechanisms to hinder the attack include having the OS restrict the total number of open sockets to avoid the port reservation attack. Even though `ulimit` is supposed to limit the file descriptors of each user to 1024 or 4096, it does not seem to be enforced correctly on Mac or Linux at the moment.

A second recommendation is to have the OS provide isolation among users of the same machine with each user having its own *dedicated DNS cache*. Isolation prevents a malicious user from poisoning the cache for other users as in the attack scenario with a public machine.

Chapter 5

CSProp: Ciphertext and Signature Propagation

Critical infrastructure on the Internet relies on the distribution of roles and responsibilities over several nodes. The interaction between nodes often occurs over secure channels to provide the required level and type of security (*i.e.,* confidentiality, integrity, availability – the CIA triad). Operating securely in constrained environments is one of the primary challenges facing the wide-scale deployment of Internet of Things (IoT) and other embedded systems on the edge of the Internet. The problem is that the cryptographic algorithms used to secure interactions between well-provisioned desktop and server environments are computationally prohibitive for resource-poor, battery operated devices. This could be a major issue since the number of interconnected IoT devices is increasing dramatically. By the year 2025, it is estimated that the number of IoT devices will be over 75 billion[111]; thus, it is essential to develop security solutions for them.

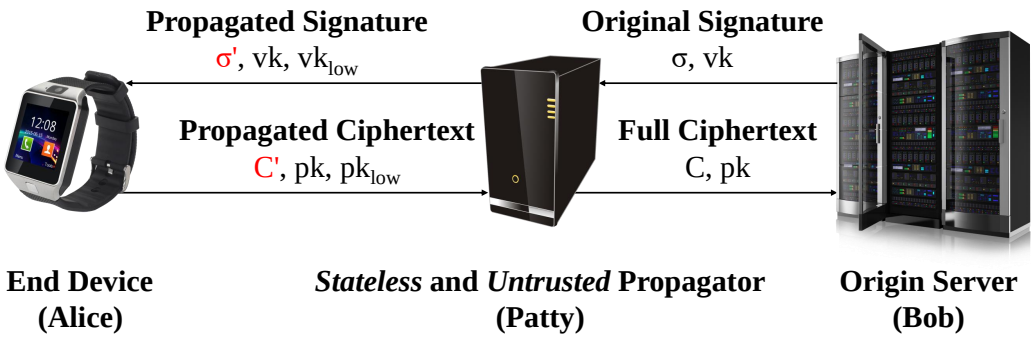


Figure 5.1: High Level Overview of CSProp

Consider a security problem which arises when resource-constrained devices are added to a secure network of more capable machines. If the security subroutines used by the network are too computationally intensive for the small device, then either (1) performance will suffer; (2) security will suffer; or (3) security for the network must be overhauled so the new device can participate. In standardized large-scale networks, (3) is likely not an option and so (2) will be chosen to avoid a performance hit.

This research contributes a new cryptographic primitive: *Ciphertext and Signature Propagation* (CSProp). When used for signature propagation, CSProp allows a capable machine (even one that is *stateless and untrusted* (*e.g.,* a certificate is not required to authenticate it) sitting upstream of a lightweight device to modify and forward (propagate) an authenticated message so it can be efficiently verified by a lesser machine. In Figure 5.1, the capable machine is Patty—which can be untrusted and stateless—who wants to propagate Bob's signature to the lightweight device Alice. The trivial solution where Patty simply

Table 5.1: Glossary

Acronym	Definition	Acronym	Definition
sk	Secret key	pk	Public key
pk_{low}	Low public key	vk	Verification key
vk_{low}	Low verification key	N	Public modulus
e	Public exponent	e_{low}	Low public exponent
d	Private exponent	M	Plaintext message
C	Ciphertext	C'	Partial decrypted ciphertext
h	Message digest	σ	Digital signature
σ'	Partial verified digital signature	K	Pre-master key
H	Hash function	$\mathcal{A}$	Adversary
$\mathcal{C}$	Challenger	$\mathcal{P}$	Computational problem
ϕ	Totient function	A	Address record
DS	Delegation signer record	DNSKEY	DNS Key record
RRset	A set of DNS records of same type	RRSIG	DNSSEC signature
KSK	Zone's key signing key	ZSK	Zone's zone signing key
RRsetA	RRset of A record(s) type	$\mathsf{RRsetDS}$	RRset of DS record(s) type
$\mathsf{RRsetDNSKEY}$	RRset of DNSKEY records type		

forwards Bob's (data, signature) pair directly to Alice puts unacceptable strain on Alice's resources. Another trivial solution where Patty simply verifies Bob's signature herself and forwards only the data to Alice is undesirable from a security point of view as it requires Alice to be trusted, and also opens the door for an attacker who targets the link between Alice and Patty.

CSProp offers a third solution, illustrated in Figure 5.1 (see Table 5.1 for the terms). Using CSProp , Patty can modify Bob's (message, signature) pair, and produce a propagated signature whose verification requires a computation of lower complexity than the original. CSProp construction provides security guarantees that there is no way for Patty to produce a valid lightweight propagated signature, except by propagating an original valid signature from Bob. Thus, CSProp securely implements a lightweight channel between Alice and Bob, without requiring any modifications to the protocol at Bob (i.e., providing backward compatibility at the server). If, for example, Alice is a low-powered wearable device who wants to connect to Bob (*e.g.,* a large web server), the propagation role of Patty might be played by a DNS resolver (or default gateway, or fog/edge server) which has capability to perform cryptographic operations, too expensive for Alice. CSProp can also be used for ciphertext propagation: a similar technique for RSA is provided which allows a weak device to use an efficient encryption procedure to produce a ciphertext with low overhead, that can then be propagated forward by Patty into a standard RSA ciphertext of the same message. I provide related background and preliminaries in Section §5.1 and present a formal definition of the new primitive, as well as an instantiation based on RSA in Section §5.2.

The design of CSProp considers public-key operations (*i.e.,* ciphertext encryption and signature verification). Such operations are typically executed at the client's end specially when using Internet protocols such as the Domain Name System SECurity extension (DNSSEC) and the Transport Layer Security (TLS) protocols. Public-key operations are known to be computationally expensive, and if executed more frequently, more burden is

added to the resource-constrained devices (*e.g.,* IoT and edge computing); see Chapter §6. Thus, developing innovative lightweight cryptosystems in such environments is important. The National Institute of Standards and Technology (NIST) initiated an effort to standardize lightweight cryptographic algorithms for small electronics [22]. However, most of this work focuses on symmetric cryptography, whereas my work focuses on Public Key Cryptography (PKC). There are also other lightweight cryptography proposals that usually require expensive computations [237, 39] or reported to be vulnerable to different types of attacks [279, 118, 32]. Introducing new protocols also challenges deployment within an Internet scale system such as those I consider. Also related to my work is a small body of work considering exploiting a proxy to reduce the complexity of cryptography [147, 71]. Unlike these efforts, CSProp does not require a trusted proxy, and I show experimentally that it outperforms this class of approaches.

The efficiency of CSProp relies on using a small public exponent for one of the public key factors. This approach bears some similarities to the use of small public exponents in RSA, but with some important differences due to the fact that *CSProp uses a small factor rather than the full key.* Readers might wonder why not use a full public exponent that is low (*e.g.,* $e = 3$) for the RSA cryptosystem, which is an idea that has been considered to accelerate RSA operations in the past. My rationale is two-fold: (1) Security: RSA with low public exponent has been demonstarted to be vulnerable to some types of attacks that could break RSA encryption and verification [152, 96, 69], although they can be prevented by avoiding implementation pitfalls that enable them. In contrast, CSProp is immune against these attacks since only a small factor is used, not the full exponent: I

elaborate on this security advantage against some known attacks on low public exponents in Section §5.1; and (2) Compatibility: in light of the known attacks on low public exponents, RFC recommendations [99] and vendor practice favor using larger public exponents, which presents a substantial barrier to using low public exponents throughout the system. Despite realization of the potential of using RSA with small public exponents [178] (assuming a secure padding scheme such as RSAES-OAEP [57] and RSASSA-PSS [59] is used), vendors and organizations continue to choose to restrict support to larger exponents [99]. In contrast, CSProp supports backward compatibility by choosing larger exponents, but requires that *only a factor of the public exponent is low* and with propagation allows us to speed up the RSA encryption and verification processes specifically for resource-constrained devices.

I further discuss various opportunities for realizing energy-efficient implementations of security protocols. Specifically, in Section §5.3, CSProp is applied to two important Internet protocols: DNSSEC and TLS. They are the core protocols in Internet communications. They rely on the use of public-key operations to safeguard connections between servers and clients. Typically such operations are offloaded to a third server (*e.g.,* a DNS resolver or a default gateway), thus shielding the end devices from overhead of encryption and verification. However, the last hop is left unprotected: for example, a recent attack[36] has shown that DNS cache poisoning can be performed between the end device and the resolver to directly poison the OS-wide DNS cache of the victim's system. CSProp can be used to secure the end devices by having the resolver propagate the signatures forward for efficient verification. Similarly, in case of TLS, instead of using the default gateway (which need to be trusted) to perform all the cryptographic operations, CSProp can instead allow

the default gateway to propagate the cryptographic material to and from the end device for efficient validation/encryption in order to maintain end-to-end security.

In Section §5.4, I evaluate the impact of CSProp by describing experiments on three generations of Raspberry PIs. Substantial savings are achieved in consumed energy and latency. More precisely, on Raspberry Pi Zero, the propagated signature verification in DNSSEC (vs. traditional DNSSEC validation) reduces latency by a factor of $78x$ and energy consumption by $47x$. For TLS handshake, the advantage to latency and energy by an average of $8x$ and $8x$, respectively (considering the full TLS handshake, which has substantial message delays that are unaffected by CSProp). I also compare CSProp with Elliptic Curve Cryptography (ECC) cipher suite and found that CSProp beats up ECC by 2.7 times.

This research direction makes the following main contributions:

1. I introduce a new cryptographic primitive which is named *Ciphertext and Signature Propagation (CSProp)* that can enable embedded and IoT devices to participate in Public Key Cryptography (PKC) at a much lower overhead than traditional implementations. A formal definition of the new primitive is presented, as well as an instantiation based on RSA. I further present its security proof under the RSA-based instantation (see Section §5.2).

2. I show the effectiveness of CSProp on two commonly used applications: DNSSEC and TLS (see Section §5.3). Accordingly, I evaluate the impact of CSProp by describing experiments on three generations of RaspberryPIs. The experiments show substantial performance and energy gains from using CSProp (see Section §5.4).

I discusses these and other related work in Chapter §2. I summarize the conclusions and discuss future work in Section §7.6.

5.1 Background and Preliminaries

In this section, I present some cryptographic preliminaries to provide the necessary background for describing CSProp. Specifically, I introduce the RSA problem and explain low public exponent RSA. Next, some special case attacks on RSA with low public exponents are discussed. I refer interested readers to [68] for an excellent survey of the subject.

5.1.1 The RSA Problem

All cryptographic primitives in this work have security based on the RSA problem, which is the following mathematical problem:

Given integers (N, e) where $N = p \cdot q$ is the product of two secret primes, find d such that $e \cdot d = 1 \pmod{\phi(N)}$, where $\phi(N) = (p-1)(q-1)$ is Euler's totient function.

If $e \cdot d = 1 \pmod{\phi(N)}$ then $x^{e \cdot d} = x \pmod{N}$ and so the modular exponentiation functions $x \mapsto x^e \pmod{N}$ and $x \mapsto x^d \pmod{N}$ are inverses of one another. In cryptography, the stronger assumption is often made that given (N, e) and a random $x \pmod{N}$, it is hard to compute $x^d \pmod{N}$. In cryptographic terminology, this ammounts to saying that $x \mapsto x^e \pmod{N}$ is a *trapdoor permutation*. Moving forward, when speaking about the RSA problem, this is the variant to which I am referring.

5.1.2 Low Public Exponent RSA

The computation time of RSA encryption and digital signature verification are dominated by the time required to compute the $e-$th power of the message and signature. To reduce computation time, e can be chosen to be a small number. The lowest possible exponent recommended is $\mathsf{e} = 3$ [68], but $e = 5$, $e = 17$, and $e = 2^{16} + 1 = 65,537$ are also common. For example, RFC3110 [121] recommends choosing $e = 3$ in order to optimize signature verification in DNSSEC, and Ferguson and Schneier [131] suggest using $e = 3$ for signatures and $e = 5$ for encryption.

The RSA problem when e is set to a public fixed small value (as supposed to e being chosen randomly in normal RSA) is known as the *low public exponent* variant of RSA[1].

5.1.3 Special Case Attacks on RSA with Low Public Exponents

The hardness of the RSA problem and its efficient cousin, RSA with low public exponent, is the subject of an extensive body of work. This is appropriate as the assumption that these problems are hard is used to justify the security of a large portion of today's internet traffic. CSProp is different from traditional low public exponent RSA in that only one of the public exponents is small (*i.e.,* not the full public exponent e), with important implications that make it *not vulnerable* for the attacks on low public exponents (please refer to Section §5.2.3 for the proof of security). CSProp's resilience to these attacks results from two factors: (1) CSProp uses a small public exponent, rather than a small public key, making its security properties equivalent to RSA before propagation; and (2) Low public exponent

[1] Choosing a low private exponent d is insecure and can completely break the cryptosystem[273, 159]

problems are primarily due to incorrect usage: since the propagation scheme is automated and not typically directly accessible to users, it does not have implementation issues. In fact, Section §5.2 shows that CSProp's security depends on the security of traditional RSA.

Partial Key/Message Exposure Attacks. Coppersmith [96] showed an attack on RSA with low public exponents when the attacker knows two-thirds of the bits of the message. While "message guessing" attacks can easily be avoided if proper padding is used, Boneh, Durfee and Frankel [69] extended Coppersmith's technique to give an attack on RSA with low public exponents when the adversary knows at least a quarter of the bits of the secret key. Other works [56, 135, 248] demonstrate that in some circumstances it is possible to recover bits of the key via side-channel attacks.

Broadcast Attacks. Håstad [152] described a factorization algorithm (thus breaking RSA) if the adversary gets access to 3 ciphertexts which encrypt the same message under 3 different low public exponent public keys $(N_1, 3), (N_2, 3), (N_3, 3)$. His technique generalizes to larger values of e_{low} and requires roughly e_{low} different encryptions. Other works [42, 97] generalize this method to attack RSA when related (as supposed to the exact same) messages are encrypted multiple times under different low exponent public keys.

Bad Implementation Attacks. Boneh et al. [70] addressed another class of low public exponent attacks which focuses on the implementation of RSA rather that the RSA function. The attack succeeds if a fraction of the bits of the private key d is exposed; consequently, an adversary would be able to construct d. Indeed, this attack in not only applicable when $e_{\mathsf{low}} = 3$, taking into consideration the computation overhead when $e > 3$. Interestingly, some other attacks are even more practical with large public exponents [258].

5.2 Ciphertext and Signature Propagation

In this section, I formally introduce CSProp. I further provide the instantiation of CSProp based on the RSA cryptosystem and the proof of security.

5.2.1 Definitions

Notation. Throughout this section, n denotes a security parameter, and $\mathcal{P}$ and $\mathcal{P}'$ are computational problems representing the cryptographic problems facing the adversary in the original and propagated signature domains respectively.

Definition (Signature Propagation). A *$\mathcal{P}-to-\mathcal{P}'$ signature propagation scheme of rate* R is a set of efficient algorithms:

$$(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Prop}, \mathsf{VerifyProp})$$

satisfying the following syntax, efficiency, correctness, and security requirements.

- • **Syntax:**

- $\mathsf{KeyGen}(1^n)$ outputs $(\mathsf{vk}, \mathsf{vk}', \mathsf{sk})$.

- $\mathsf{Sign}(\mathsf{M}, \mathsf{sk}, \mathsf{vk})$ outputs σ.

- $\mathsf{Verify}(\mathsf{M}, \mathsf{vk}, \sigma)$ outputs a bit.

- $\mathsf{Prop}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma)$ outputs σ'.

- $\mathsf{VerifyProp}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma')$ outputs a bit.

• **Efficiency:** I have $R \cdot T' = \mathcal{O}(T)$ where T and T' denote the running times of Verify and $\mathsf{VerifyProp}$, respectively.

• **Correctness:** Fix a message M arbitrarily. Consider the random procedure:

1) draw $(\mathsf{vk}, \mathsf{vk}', \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$;

2) draw $\sigma \leftarrow \mathsf{Sign}(\mathsf{M}, \mathsf{sk}, \mathsf{vk})$;

3) draw $\sigma' \leftarrow \mathsf{Prop}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma)$.

Then,

$$\mathsf{Verify}(\mathsf{M}, \mathsf{vk}, \sigma) = \mathsf{VerifyProp}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma') = 1$$

holds with probability 1.

• **Security:**

There are efficient reductions from an adversary who wins the standard existential unforgeability game for $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ (resp. G, below) to an adversary who solves $\mathcal{P}$ (resp. $\mathcal{P}'$). The game G is between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$ and works as follows:

The Signature Propagations Game G:

1. $\mathcal{C}$ draws $(\mathsf{vk}, \mathsf{vk}', \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$ and sends $(\mathsf{vk}, \mathsf{vk}')$ to $\mathcal{A}$.

2. For $i = 1, \ldots, \mathsf{poly}(n)$: $\mathcal{A}$ sends query messages, M_i, to $\mathcal{C}$; $\mathcal{C}$ computes $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{M}_i, \mathsf{sk}, \mathsf{vk})$ and sends σ_i back to $\mathcal{A}$.

3. Finally, $\mathcal{A}$ sends a pair (M^*, σ^*) and wins if:

$$\mathsf{VerifyProp}(\mathsf{M}^*, \mathsf{vk}, \mathsf{vk}', \sigma^*) = 1 \text{ and } \mathsf{M}^* \neq \mathsf{M}_i \ \forall \ i.$$

Remark. So in a $\mathcal{P}-$to$-\mathcal{P}'$ signature propagation scheme, $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is a standard signature scheme assuming the hardness of the problem $\mathcal{P}$; and $(\mathsf{KeyGen}, \mathsf{Prop} \circ \mathsf{Sign}, \mathsf{VerifyProp})$ is a signature scheme assuming hardness of $\mathcal{P}'$; moreover, $\mathsf{VerifyProp}$ is $R-$times faster than Verify. Thus, signature propagation gives a way to improve verification

efficiency while still maintaining security assuming hardness of $\mathcal{P}'$ (a possibly stronger assumption, which I will demonstrate for RSA).

Propagation for ciphertexts is defined similarly.

Definition (Ciphertext Propagation). A *$\mathcal{P}-to-\mathcal{P}'$ ciphertext propagation scheme of rate R* is a set of efficient algorithms:

$$(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prop}, \mathsf{DecProp})$$

satisfying the following syntax, efficiency, correctness, and security requirements.

• **Syntax:**

- $\mathsf{KeyGen}(1^n)$ outputs $(\mathsf{pk}, \mathsf{pk}', \mathsf{sk})$.

- $\mathsf{Enc}(\mathsf{M}, \mathsf{pk})$ outputs C.

- $\mathsf{Dec}(\mathsf{C}, \mathsf{sk})$ outputs a message M'.

- $\mathsf{Prop}(\mathsf{C}, \mathsf{pk}, \mathsf{pk}')$ outputs C'.

- $\mathsf{DecProp}(\mathsf{C}', \mathsf{sk})$ outputs a message M'.

• **Efficiency:** I have $R \cdot T = \mathcal{O}(T')$ where T and T' denote the running times of Enc and Prop, respectively.

• **Correctness and Security:** $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a standard encryption scheme assuming the hardness of $\mathcal{P}'$; $(\mathsf{KeyGen}, \mathsf{Prop} \circ \mathsf{Enc}, \mathsf{DecProp})$ is an encryption scheme assuming the hardness of $\mathcal{P}$; correctness and security are inherited.

5.2.2 Propagating RSA Signatures

In this section I instantiate a $\mathcal{P}-\text{to}-\mathcal{P}'$ signature propagation scheme where $\mathcal{P}$ is standard RSA and $\mathcal{P}'$ is RSA with low public exponent, specifically with exponent e_{low}. The construction uses a hash function H, modeled as a random oracle.

- $\mathsf{KeyGen}(1^n)$ generates an RSA modulus $N = p \cdot q$ for secret primes p and q and draws a random e such that $e_{\mathsf{low}}|e$; find d such that $e \cdot d = 1 \pmod{\phi(N)}$. Output:

$$(\mathsf{vk}, \mathsf{vk}', \mathsf{sk}) = \big((N, e), (N, e_{\mathsf{low}}), (N, d)\big).$$

- $\mathsf{Sign}(\mathsf{M}, \mathsf{sk}, \mathsf{vk})$ computes $h = \mathsf{H}(\mathsf{M}, \mathsf{vk})$, and outputs $\sigma = h^d \pmod{N}$.

- $\mathsf{Verify}(\mathsf{M}, \mathsf{vk}, \sigma)$ computes $h = \mathsf{H}(\mathsf{M}, \mathsf{vk})$ and outputs 1 if $\sigma^e = h \pmod{N}$, 0 otherwise.

- $\mathsf{Prop}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma)$ outputs $\sigma' = \sigma^{e/e_{\mathsf{low}}} \pmod{N}$.

- $\mathsf{VerifyProp}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma')$ computes $h = \mathsf{H}(\mathsf{M}, \mathsf{vk})$ and outputs 1 if $(\sigma')^{e_{\mathsf{low}}} = h \pmod{N}$, 0 otherwise.

Theorem. Let $R = |e|/|e_{\mathsf{low}}|$, where $|e|$ and $|e_{\mathsf{low}}|$ are the bit-lengths of e and e_{low}, respectively. Then,

$$(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Prop}, \mathsf{VerifyProp})$$

is a $\mathcal{P}-\text{to}-\mathcal{P}'$ signature propagation scheme with rate R, in the random oracle model [58]. The random oracle is a standard strong assumption on perfectly random hash functions supporting the collision resistance property, which I inherit from the use of RSA. Such hash functions require that for every unique input the function generates a unique output chosen with equal probability from the output domain.

5.2.3 Security Proof

In this section, I present the security proof of the $\mathcal{P}-$to$-\mathcal{P}'$ signature propagation scheme under the RSA-based instantation. I first prove that the existential unforgeability property of $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ holds, which implies security with respect to signatures. I also discuss the security of the scheme relative to attacks on low public exponents. Finally, using cryptographic game theory, I show that the security of the signature propagation game depends on the security of the standard RSA game only (i.e., CSProp is secure if RSA is secure). Note that the proof is also applied to the instantiation of $\mathcal{P}-$to$-\mathcal{P}'$ ciphertext propagation scheme using RSA.

Proof. Correctness follows from verifying the equation:

$$\left((h^d)^{e/e_{\mathsf{low}}}\right)^{e_{\mathsf{low}}} = (h^{d\cdot(e/e_{\mathsf{low}})\cdot e_{\mathsf{low}}} = h^{d\cdot e} = h \pmod N$$

using $d \cdot e = 1 \pmod{\phi(N)}$. The subroutines Verify and $\mathsf{VerifyProp}$ are dominated by computing $e-$th and $e_{\mathsf{low}}-$th powers mod N, which require executing the "square mod N" function $\mathcal{O}(|e|)$ and $\mathcal{O}(|e_{\mathsf{low}}|)$ times, respectively; thus, the efficiency property holds. Note that $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is the standard RSA signature scheme, except that the exponent e is chosen randomly subject to the condition that $e_{\mathsf{low}}|e$. This event naturally occurs with probability roughly $1/e_{\mathsf{low}}$ in the plain RSA scheme, and so existential unforgeability of $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ holds since the standard RSA problem is hard [68].

Similarly, attacks on RSA with low public exponent do not apply to the propagation scheme. In particular, because the strength of $\mathsf{VerifyProp}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma')$ holds depending on the hardness of the standard RSA signature verification procedure: $\mathsf{Verify}(\mathsf{M}, \mathsf{vk}, \sigma)$ the attacks do not apply to CSProp. In other words, the original signature σ is verified using

both exponents e/e_{low} and e_{low}, and so $\mathsf{VerifyProp}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma')$ holds *iff* σ' is generated using the propagation procedure: $\mathsf{Prop}(\mathsf{M}, \mathsf{vk}, \mathsf{vk}', \sigma)$. Technically, this means that a malicious proxy that attempts to forge a propagated signature fails by the construction of VerifyProp (Section 3.2) assuming the standard RSA problem to solve subroutine Verify is hard. Note that the proof of security also applies to the RSA ciphertext propagation scheme. Thus, a malicious proxy can cause denial of service but cannot forge a signature on falsified or incorrect data. I elaborate on this case to show how to use an adversary who wins the signature propagation game to solve the RSA problem with public exponent e_{low}, implying the impossibility of this attack provided RSA is secure. The security proof is as follows:

So suppose $\mathcal{A}$ is an efficient adversary who wins the signature propagation game with probability $\epsilon > 0$. I design another adversary $\mathcal{A}'$ which, given (N, e_{low}) and a random $h^* \pmod N$, outputs $(h^*)^{d_{\mathsf{low}}} \pmod N$ also with probability ϵ, where d_{low} is such that $d_{\mathsf{low}} \cdot e_{\mathsf{low}} = 1 \pmod{\phi(N)}$. $\mathcal{A}'$ works as follows:

- Upon receiving $(N, e_{\mathsf{low}}, h^*)$, $\mathcal{A}'$ chooses a large random integer e' and sends (N, e, e_{low}) to $\mathcal{A}$ where $e = e_{\mathsf{low}} \cdot e'$.
- Instantiate Q, a set of queries of $\mathcal{A}$ to $Q = \{\ \}$. Each time $\mathcal{A}'$ queries for a signature of a message M do the following:
  - check if M has already been asked by $\mathcal{A}$; if so return σ to $\mathcal{A}'$ where (M, σ) is the pair appearing in Q;
  - otherwise, choose a random number $\sigma \pmod N$ and return σ to $\mathcal{A}'$;
  - add (M, σ) to Q;

- set $h = \sigma^e \pmod N$ and program the input/output pair $\big((\mathsf{M}, N, e), h\big)$ into H, so that if $\mathsf{H}(\mathsf{M}, N, e)$ is computed again at any point in the experiment, h will be returned.

- Finally, when $\mathcal{A}$ is ready to return its forgery of the message M^*, $\mathcal{A}'$ works as follows:
  - if M^* appears as the first coordinate of some pair in Q, $\mathcal{A}'$ aborts giving no output;
  - otherwise, when $\mathcal{A}$ queries H on the input (M^*, N, e), $\mathcal{A}'$ returns h^*;
  - finally when $\mathcal{A}$ sends (M^*, σ^*), $\mathcal{A}'$ outputs σ^* and halts.

Notice that $\mathcal{A}'$ answers the queries of $\mathcal{A}$ correctly because $\sigma^e = \mathsf{H}(\mathsf{M}, N, e)$ holds for them all. Furthermore, if h^* is a random number (mod N) then the response to $\mathcal{A}$'s hash query $\mathsf{H}(\mathsf{M}^*, N, e)$ is uniformly distributed. These two observations mean that $\mathcal{A}'$ properly simulates the signature propagation game for $\mathcal{A}$, and so by assumption, $\mathcal{A}$ wins this game with probability ϵ. Finally, note that whenever $\mathcal{A}$ wins the signature propagation game, $(\sigma^*)^{e_{\mathsf{low}}} = h^* \pmod N$ holds, which implies $\sigma^* = (h^*)^{d_{\mathsf{low}}} \pmod N$, and so $\mathcal{A}'$ breaks low public exponent RSA. ■

5.3 Applications of CSProp

I illustrate the use and advantages of CSProp on two important Internet protocols: DNSSEC and TLS, which are core protocols with respect to securely connecting end devices to the Internet. Both DNSSEC and TLS are used to provide integrity, confidentiality, and/or authentication for critical data. Often real-world configurations force end devices

to rely on third parties (*e.g.,* DNS resolver and default gateway) to perform cryptographic functionality such as decryption or verification on their behalf. Although such a setup reduces the requirement on the energy-constrained end devices, it compromises security: if the third party is compromised or spoofed, the end devices are completely compromised. Moreover, the last hop between the third party and the end devices becomes vulnerable to attacks (e.g., a recent client-side attack on DNS bypasses DNSSEC [36]). In this section, I show how we can use CSProp to extend DNSSEC and TLS verification to the end devices, providing security with acceptable overhead.

5.3.1 CSProp over DNSSEC

The Domain Name System (DNS) is an essential networking protocol. It is responsible for mapping Fully Qualified Domain Names (FQDNs) to their corresponding IP addresses. To defeat certain DNS attacks (*e.g.,* cache poisoning[181] and amplification[13] attacks), DNS SECurity Extension (DNSSEC)[47] is proposed as a form of cryptographic defense to authenticate DNS responses with digital signatures. DNSSEC is standardized by the Internet Engineering Task Force (IETF). Without DNSSEC, DNS becomes vulnerable to different classes of attacks where an attacker attempts to provide false responses to queries [181]. DNSSEC operates by adding cryptographic signatures to existing DNS records to prove that they are legitimate responses from trusted servers. Specifically, these signatures provide DNS clients origin authentication and integrity of data (but not confidentiality). Typically, verification of the signatures is implemented by resolvers, rather than the end devices themselves, to reduce the overhead on the end devices. When an end device performs a DNS query, it sends the query to its resolver. If the data is not present in

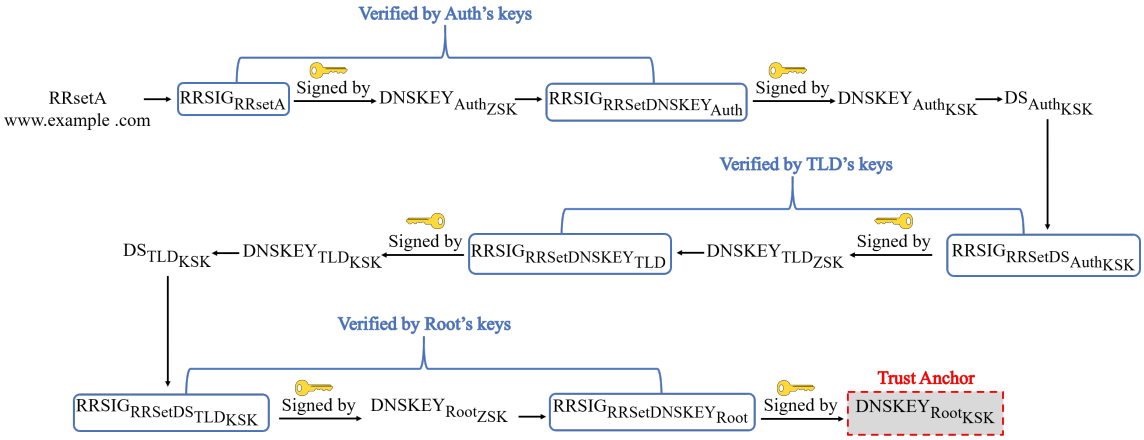


Figure 5.2: DNSSEC Chain of Trust

the resolver's DNS cache, the resolver starts the resolution process by traversing the DNS hierarchy from the root server and down to the corresponding authoritative name server.

Unfortunately, to shield the end devices from these expensive operations, this design leaves opportunities for attackers on the last hop between the resolver and the end device. For example, a resolver that is compromised can arbitrarily falsify information. Moreover, an attacker can spoof the resolver or otherwise inject responses to attack the end devices [36].

Without end-to-end authentication, DNS security cannot be guaranteed. A trivial solution is to ask the end devices to carry out the authentication, but this requires multiple expensive cryptographic operations as discussed in the next section. To secure DNS against attacks [36], CSProp is used to provide low overhead end-to-end DNSSEC validation.

DNSSEC Chain of Trust

Each DNS query traverses a hierarchy of DNS servers, starting from the root server, to the TLD server, and so on through the chain of the authoritative DNS servers. The validity of the resolution depends on trusting the full sequence of DNS resolution, forming a chain of trust that is to be validated by the resolver (or in my case, the end devices). DNSSEC has been proposed to establish secure DNS resolution using a chain of trust [47] that follows the hierarchical structure of DNS. DNSSEC establishes the trust by validating signatures from an authoritative name server up to the root in a child-parent manner. For example, as illustrated in Figure 5.2 (see Table 5.1 for definitions), when a DNS resolver receives RRsets of types other than DNSKEY (*e.g.,* the record of type A) returned by the `example` authoritative name server, the `example` helps verify these records. The `.com` name server helps verify RRsets of type DNSKEY returned by `example.`, and the root server helps verify `.com`. The trust anchor of the chain is the KSK of the root server, and all data published by the root is vetted by a thorough security procedure called the Root Signing Ceremony[2]. In the ceremony, several individuals from around the world collaborate to sign the root DNSKEY RRset in a very public and highly audited way.

DNSSEC Signing Algorithm

There has been no standardization of a specific zone signing algorithm. The usable algorithms usually appear in DNSKEY, RRSIG, and DS RRsets [243, 162, 271, 48]. In practice, root servers always use **Algorithm 8** (which is RSA/SHA256) [243]. However,

[2]Root Signing Ceremony is explained in more detail at the following resource: `https://www.cloudflare.com/dns/dnssec/root-signing-ceremony/`

Table 5.2: DNSSEC Algorithm Use Statistics

Algorithm		**# of DS records Signed**	
Code	*Name*	*TLDs*	*Alexa*
3	DSA/SHA1	0	7
5	RSA/SHA-1	163	1305
7	RSASHA1-NSEC3-SHA1	539	5669
8	RSA/SHA-256	2157	10962
10	RSA/SHA-512	37	758
12	ECC-GOST	0	3
13	ECDSAP256SHA256	5	6017
14	ECDSAP384SHA384	0	202

to the best of my knowledge, there is no documentation of the algorithm used to sign the zones of TLDs and authoritative name servers. For that, I conducted a measurement study to analyze the DS records of the TLDs by examining the root DNS zone[3]. Similarly, the measurement study is conducted on the top 1 million sites based on Alexa Traffic Rank[4]. As shown in Table 5.2, the findings are confirmed in [243] that **Algorithm 8** is indeed the

[3]The dataset is available online at: https://www.internic.net/domain/root.zone and managed by the Internet Corporation for Assigned Names and Numbers (ICANN)

[4]Alexa Top Sites (ATS) web service: https://aws.amazon.com/alexa-top-sites/

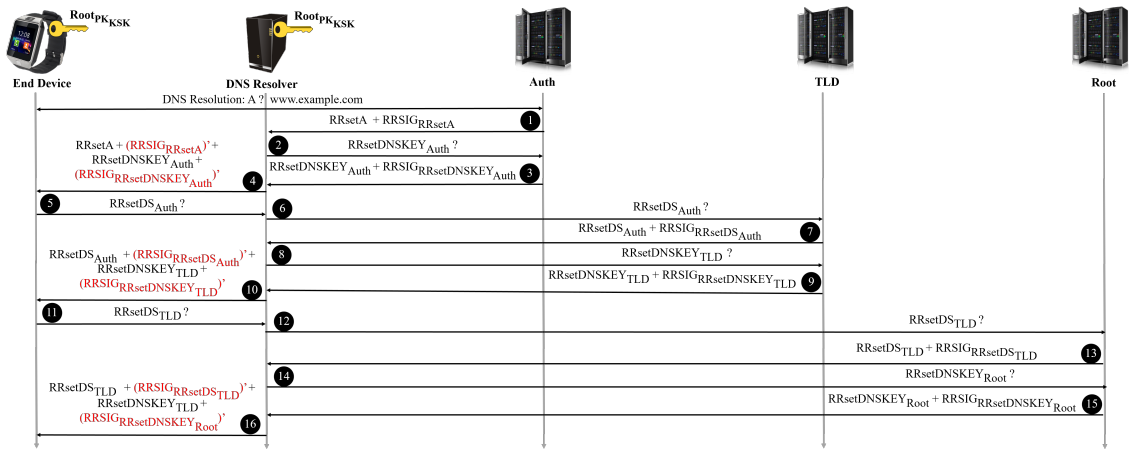


Figure 5.3: CSProp over DNSSEC — Design

most widely used algorithm in DNSSEC. The CSProp protocol supports this algorithm, making it straightforward to deply within the current ecosystem.

Design of DNSSEC with CSProp

DNSSEC using CSProp provides efficient *end-to-end* authentication from the origin server to the end device. CSProp provides signature validation over the entire chain of trust of DNSSEC. The design is illustrated in Figure 5.3. The components in red represent additions for CSProp. Moreover, in this figure, the end device takes charge of the authentication, whereas in a conventional implementation, the verification traffic is initiated by the resolver. I assume the records are not present at the resolver's cache. I explain the steps in detail as follows:

After the DNS resolution process is completed and before the legitimate response of the requested query (*e.g.,* A record of `www.example.com`) is forwarded to the end device,

in step ❶ the DNS resolver receives an RRset of type A along with the corresponding RRSIG record from the authoritative name server ($Auth$). To compute the partial validated signature (RRSIG′) of the above RRsetDNSKEY, the resolver needs the DNSKEY record of $Auth_{\mathsf{ZSK}}$ and sends a query to $Auth$ as shown in ❷. In step ❸, $Auth$ responds back and sends both $\mathsf{RRsetDNSKEY}_{Auth}$ and the corresponding $\mathsf{RRSIG}_{\mathsf{RRsetA}_{Auth}}$. In step ❹, the resolver computes $(\mathsf{RRSIG}_{\mathsf{RRsetA}_{Auth}})'$ using vk and vk′ of $Auth_{\mathsf{ZSK}}$ and $(\mathsf{RRSIG}_{\mathsf{RRsetDNSKEY}_{Auth}})'$ using vk and vk′ of $Auth_{\mathsf{KSK}}$ and forwards them to the end device along with RRsetA and $\mathsf{RRsetDNSKEY}_{Auth}$. The end device completes the validation process of $(\mathsf{RRSIG}_{\mathsf{RRsetA}_{Auth}})'$ and $(\mathsf{RRSIG}_{\mathsf{RRsetDNSKEY}_{Auth}})'$ using vk′ of $Auth_{\mathsf{ZSK}}$ and $Auth_{\mathsf{KSK}}$, respectively. Then, the end device needs to verify the RRSIG of the DNSKEY record of $Auth_{\mathsf{KSK}}$. As shown in step ❺, it sends a query to the resolver and requests $\mathsf{RRsetDS}_{Auth}$. In step ❻, the resolver forwards the query to the `.com` TLD server which responds with $\mathsf{RRsetDS}_{Auth}$ and $\mathsf{RRSIG}_{\mathsf{RRsetDS}_{Auth}}$ as shown in step ❼. To partially verify $\mathsf{RRSIG}_{\mathsf{RRsetDS}_{Auth}}$, the resolver in step ❽ sends a query to the `.com` TLD server for the TLD's DNSKEY records. Then, the TLD server responds in step ❾ with $\mathsf{RRsetDNSKEY}_{TLD}$ and $\mathsf{RRSIG}_{\mathsf{RRsetDNSKEY}_{TLD}}$.

As in step ❸, the resolver computes in step ❿ $(\mathsf{RRSIG}_{\mathsf{RRsetDS}_{Auth}})'$ using vk and vk′ of TLD_{ZSK} and $(\mathsf{RRSIG}_{\mathsf{RRsetDNSKEY}_{TLD}})'$ using vk and vk′ of TLD_{KSK} and forwards the partial verified RRSIGs to the end device along with $\mathsf{RRsetDS}_{Auth}$ and $\mathsf{RRsetDNSKEY}_{TLD}$. Steps ⓫-⓰ are similar to steps ❺-❿ to verify $\mathsf{RRsetDS}_{TLD}$ and $\mathsf{RRsetDNSKEY}_{Root}$. Finally, the end device compares the $\mathsf{DNSKEY}_{Root_{\mathsf{KSK}}}$ record with the publicly available version, and this completes the DNSSEC validation process.

The ability to establish trust between child and parent zones is an integral part of DNSSEC.

We cannot trust any of the DNS records if part of the chain is broken. CSProp over DNSSEC provides complete *end-to-end* protection and secures DNS records from being altered by MitM attackers. Furthermore, the steps of the protocol, and the number of packets exchanged between the parties is the same as in regular DNSSEC with changes isolated to the last hop between the DNS resolver and the end device (in addition to the choice of the public key). These properties make it practical to deploy the design.

5.3.2 Optimizing TLS handshakes with CSProp

In the second application, I consider using CSProp to optimize the operation of TLS. The underlying security of TLS protocol relies on the implementation of the cryptographic algorithms during the handshake phase. The cryptographic algorithms provide authentication and integrity between the communicating entities (*i.e.,* in my case, the web server and the end device). To offer these security services, the end device has to handle complex cryptographic operations for validation which are computationally expensive. By using CSProp, we can substantially reduce the computational cost incurred by the handshake phase without compromising security.

TLS Handshake

TLS is a core security protocol on the Internet and has undergone several revisions over the years to address security and performance flaws specifically in the handshake protocol [240]. CSProp is designed to work with TLS 1.3, which is the latest version improving both the performance and security of TLS 1.2. Authenticating the communicating parties to each other is typically done by validating their PKI certificates. The most commonly

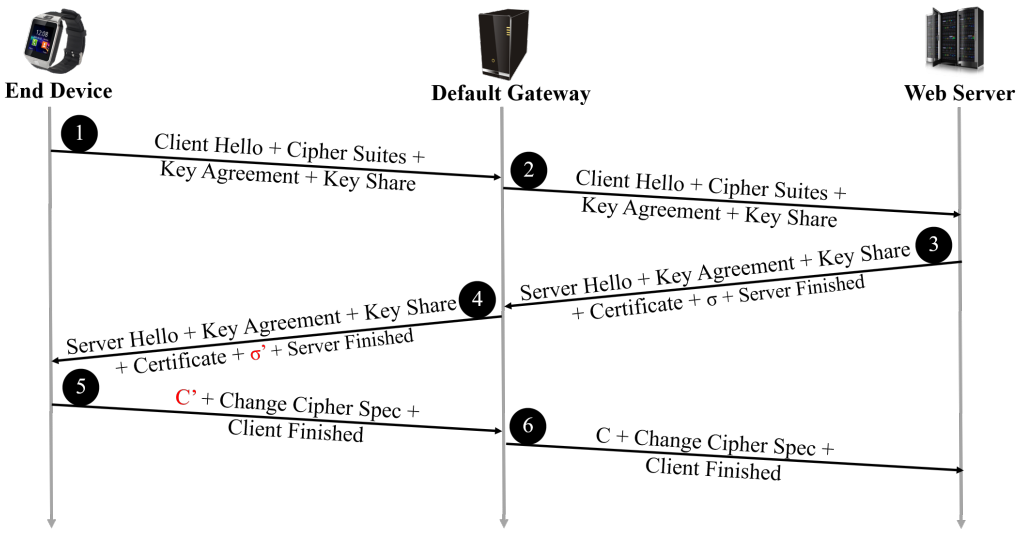


Figure 5.4: CSProp over TLS — Design

used certificate is X.509 which is based on the RSA cryptosystem [38]. In common cases, only the web server needs to be authenticated by the client (unless client authentication is required by the server).

Design

CSProp can help reduce the computation cost needed for TLS on the end device by *securely offloading* a considerable part of the encryption and validation processes to the default gateway. Initially, the communication is between the web server and the end device; however, the default gateway is present in typical scenarios of constrained environments (*e.g.,* IoT environment) as shown in Figure 5.4. The protocol is described in detail as follows:

In step ❶, the end device commences the handshake and sends the "Client Hello" message followed by the cipher suite, key agreement and key share messages to the default

gateway in which the latter forwards the messages to the designated web server. In reply, the web server sends in step ❷ the "Server Hello" message comprised of the chosen key agreement, server's X.509 certificate and its associated signature σ, and the server's key share associated with the "Server Finished" message. Then in step ❸, the default gateway forwards all messages received in step 2 to the end device — with one significant change. It substitutes σ with σ' which is the partial verified signature of the server's certificate and σ' is computed using vk and vk'. Now, the end device partially verifies σ' using vk' as shown in step ❹. In step ❺, the end device generates the pre-master secret key K using the web server's key share. K is encrypted using pk' to generate the partial ciphertext C'. The end device sends C', the cipher suite change (if it is applicable) along with the "Client Finished" message to the default gateway. Finally in step ❻, the default gateway completely encrypts K using pk and pk' and forwards messages received in step 5 to the web server along with the full ciphertext C. Upon receiving the messages, the web server using its secret key sk decrypts C to retrieve K, and this concludes the handshake. From here on, all the messages are securely exchanged between the entities.

Similar to CSProp over DNSSEC, CSProp over TLS does not require any additional messages to be exchanged between the three parties that are involved in the handshake phase. This ensures a zero-round trip handshake as in TLS 1.3.

5.4 Evaluation

In this section, I experimentally assess the effectiveness of CSProp over DNSSEC and TLS. The protocols are compared under realistic settings and with respect to a number

Table 5.3: Experimental Setup: The following platforms are used in the experiments.

Device	Model	Role	Processor (CPU)	CPU Clock (GHz)	RAM	Cores
Dell	XPS 8700	Origin Server	Intel(R) Core(TM) i7-4790	3.6	16GB	4
Sony VAIO	VPCEA390X	DNS resolver/Default gateway	Intel(R) Core(TM) i5	2.53	8GB	2
Microsoft Surface	Pro 6	End device	Intel(R) Core(TM) i7-8650U	1.9	16GB	4
Raspberry Pi	Zero W	End device (IoT)	ARM11 Broadcom	1	512MB	1
	3 Model B	End device (IoT)	Arm Cortex-A53 (ARMv8)	1.2	1GB	4
	3 Model B+	End device (IoT)	Cortex-A53 (ARMv8)	1.4	1GB	4

of end devices representative of IoT and embedded devices. I also compare CSProp with ECC cipher suite.

Selected Software. A prototype of CSProp is implemented over DNSSEC and TLS, based on the security library, dnsjava[5], and the Bouncy Castle [200] crossed-platform libraries. Bouncy Castle is a standard library for cryptography and contains APIs that are supported for the embedded devices. The dnsjava library is an implementation of DNS in Java and is used by a number of major android applications, such as Netflix, Skype, Samsung Email, and Dailyhunt [115].

Selected Hardware. I evaluate the prototype on four popular end devices: (1) Microsoft Surface Pro 6; (2) Raspberry Pi Zero W; (3) Raspberry Pi 3 Model B; and (4) Raspberry Pi 3 Model B+. As shown in Table 5.3, each device has different system specifications. These devices provide a range of embedded/mobile platforms typical of those used in constrained environments such as IoT applications [245]. All three Raspberry Pi devices run Raspbian operating system, while Surface Pro 6 runs Windows 10 Home edition operating system.

[5]Available at: http://www.xbill.org/dnsjava/

Network Connectivity. An Arris router [50] is used as the central communication device. The PCs and the Raspberry pis, except Raspberry Pi Zero W, are connected using 1Gbps Ethernet cables Ethernet.

Energy Consumption. To obtain the energy consumption values, the Watts Up? Pro AC meter [233] is used. This power meter supports several displays, computer software, and PC interfaces. It can automatically generate data graphing, and its data logger function records all data into non-volatile memory.

Testbed Setup. I configure the testbed as follows. I use a Dell PC running Ubuntu 16.04.6 LTS operating system as an origin server (*i.e.,* DNS servers in DNSSEC, web servers in TLS, and default gateway in IoT environments). For reasons of backward compatibility with middleboxes, I use the recommended key size of 2048-bit and hashing algorithm SHA-256 [240, 194]. The origin server is responsible for generating the RSA keys needed to perform all the cryptographic operations. The propagator is a Dell PC running Windows 10 Home edition operating system.

CSProp is compared with traditional implementations of DNSSEC validation, TLS handshakes, and RSA public-key operations. CSProp is also compared with current real-world configurations where the public exponent is $2^{16} + 1 = 65537$.

5.4.1 CSProp over DNSSEC

In this section, I show measurement results of CSProp over DNSSEC based on two metrics: (1) Latency; and (2) Energy consumption. A private network (simulating the topology in Figure 5.1) is configured to represent the DNS hierarchy. More precisely, I

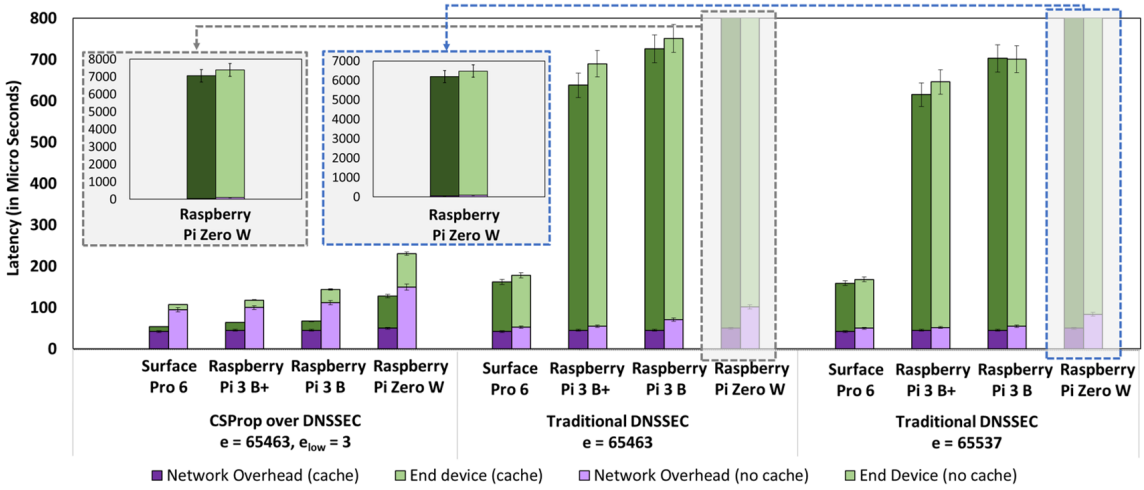


Figure 5.5: CSProp over DNSSEC — Latency

configure the *Root* (.), the *TLD* (`.com`), and the *Auth* (`example`) name servers internally in the origin server. I use `www.example.com` as the target domain name in the experiments. In addition, the $\mathsf{DNSKEY}_{Root_{\mathsf{KSK}}}$ record (*i.e.,* the trust anchor) is pre-installed at the DNS resolver and all four end devices used in the prototype. To optimize DNSSEC resolution process, the DNS resolver supports the caching property.

I collected 10 data points for each experiment to get statistically meaningful results. I calculated the 95% confidence intervals of the mean and show those on the figures. The measurements are performed when caching is enabled and disabled at the DNS resolver to get an insight of the impact of caching on the protocol. Dark colors in Figure 5.5 represent results when caching is enabled, while the light colors represent disabled-cache results. Since the measurements are conducted on different devices with different specifications, the y-axis is scaled to fit the maximum readings. To make figures more expressive and easier to understand, I use transparent gray boxes to zoom in the details in Figure 5.5 that are not visible at scale.

Figure 5.5 shows a break down of the latency incurred by CSProp over DNSSEC. The latency is broken down into the time consumed by end devices and by the network. The latter time includes: (1) the network overhead caused by sending and receiving packets between the communicated parties; and (2) the time required by the DNS resolver to compute the propagated signature. The results show a significant reduction in latency compared to traditional DNSSEC validation, with a minor impact on latency when the cache is disabled at the DNS resolver. Additionally, we see how device specifications affect performance. For example, in case $e = 65463$ and cache is disabled, the results show that CSProp reduces latency by $91x$, $21x$, $35x$, and $10x$ on Raspberry Pi Zero W, Raspberry Pi 3 Model B, Raspberry Pi 3 Model B+, and Surface Pro 6, respectively, compared to traditional DNSSEC validation. Note that the reductions are approximately the same when DNS cache is enabled. CSProp is also compared with current DNSSEC implementations where the used public exponent is 65537. The results show that CSProp outperforms this setting: for example, CSProp reduces latency by $78x$ on Raspberry Pi Zero compared to conventional DNSSEC when $e = 65537$. Apparently, the results are to some extent better than those when $e = 65463$ in the case of traditional DNSSEC validation. The reason is that 65537 is a Fermat number. Fermat numbers are $2^n + 1$ primes, and they are recommended [243] since only the first and last bits of their binary representation are ones (100...001); a feature expedites computations on computers.

Since there is no tangible difference in results with and without caching, I combine energy consumption measurements and take the average as shown in Figure 5.6. In general, the results show a significant reduction in energy consumption when CSProp is used. Indeed,

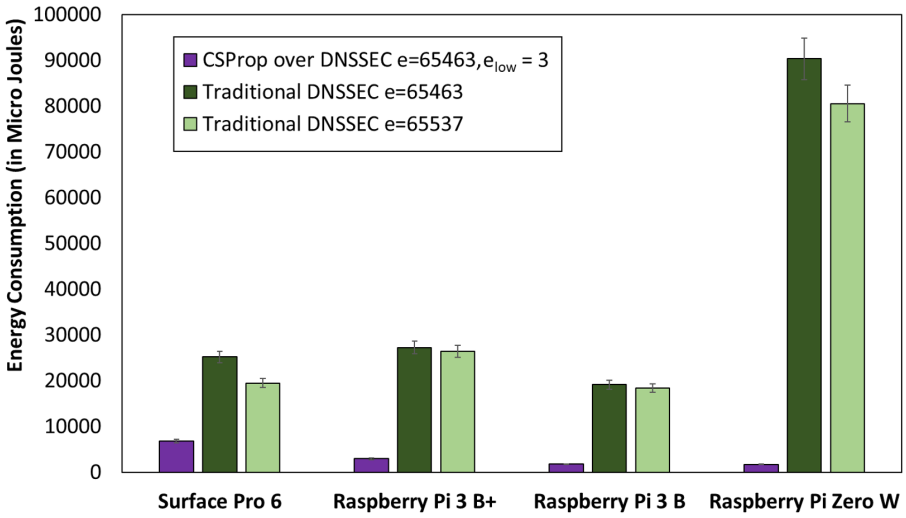


Figure 5.6: CSProp over DNSSEC — Energy Consumption

the protocol provides energy reductions by $53x$, $10x$, $9x$, and $4x$ on Raspberry Pi Zero W, Raspberry Pi 3 Model B, Raspberry Pi 3 Model B+, and Surface Pro 6, respectively.

5.4.2 CSProp over TLS

Similar to the setup phase of CSProp over DNSSEC, a private network is configured where the origin server is the web server of `www.example.com` domain name. In the measurements, I consider the handshake phase and implement it based on TLS 1.3. Note that my web server does not contain data objects since the CSProp protocol does not optimize HTTP connections after the TLS handshake is successfully accomplished. The web server's certificate is of type X.509 and is signed by a root CA which its certificate is already pre-installed at the default gateway and end devices. The pre-master secret key (K) is generated using the Advanced Encryption Standard (AES) algorithm as recommended in [240] with 128-bit as the key size.

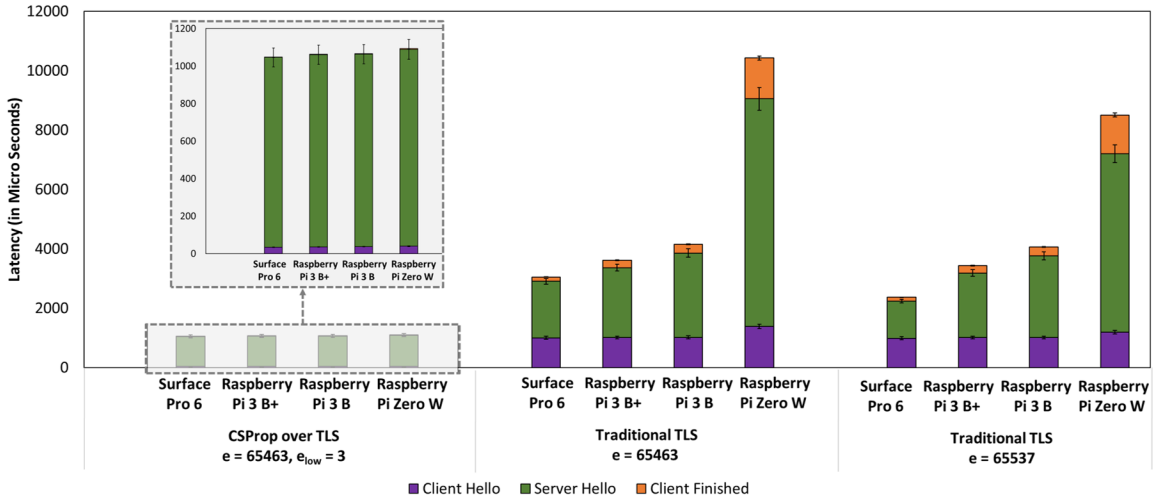


Figure 5.7: CSProp over TLS — Latency

The latency of the TLS operations are shown in Figure 5.7. I show the latency incurred by CSProp over TLS but based on the handshake messages: "Client Hello", "Server Hello", and "Client Finished". This approach is used to clearly understand the advantage of the protocol over the existing implementations; specifically when $e = 65537$. CSProp provides $8x$, $4x$, $3x$, and $2x$ reductions in latency (vs. traditional TLS handshake) on Raspberry pi Zero W, Raspberry Pi 3 Model B, Raspberry Pi 3 Model B+, and Surface Pro 6, respectively. Note that these numbers are the full handshake numbers, including the network delays (which are not helped by CSProp).

For energy consumption measurements, I measured the rate at which power is being used at a specific moment in watts (as shown in Figure 5.8). The results show that CSProp, on average, reduces the consumed energy by a factor of $8x$, $3x$, $3x$, and $2x$ on Raspberry Pi Zero W, Raspberry Pi 3 Model B, Raspberry Pi 3 Model B+, and Surface Pro 6, respectively. Again, these numbers include the energy consumed across the full

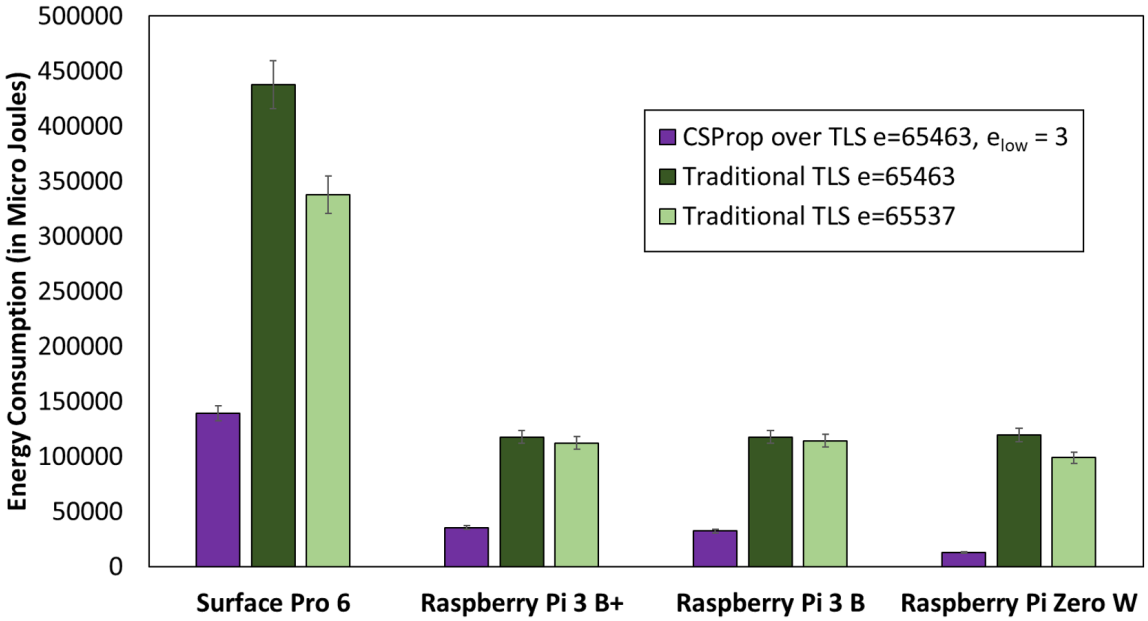


Figure 5.8: CSProp over TLS — Energy Consumption

handshake, with long periods of time taken up for network communication in which the energy consumed is not affected by CSProp.

It is interesting to note that the less resources the embedded device has, the larger the advantage from CSProp. I believe that this occurs since deeply embedded devices are likely not to have sophisticated energy saving features such as Dynamic Voltage and Frequency Scaling (DVFS) [251], which can help more sophisticated devices adapt their energy usage, for example, to use less power while waiting for responses from the network.

5.4.3 Comparison with Elliptic Curve Cryptography (ECC) Cipher Suites

When power and latency are a consideration, Elliptic Curve Cryptography (ECC) is often considered: it has an approximate equivalent strength to RSA and, in fact, has some

Table 5.4: Comparing CSProp with Elliptic Curve Cryptography (ECC) cipher suites for TLS handshake latency (*in Micro Seconds*)

	Raspberry Pi 3 B+	**Raspberry Pi B**	**Raspberry Pi Zero W**
CSProp $e = 65463$**,** $e_{low} = 3$	1063.24	1066	1093.56
ECDH-ECDSA P-256	2658.18	2984.66	3171.32

advantages relative to using RSA. In particular, key sizes are much shorter: *e.g.,* Elliptic Curve Digital Signature Algorithm (ECDSA) with curve P-256 has a key size of 256 bits, whereas RSA commonly uses key sizes of 1024 or 2048 bits. Additionally, ECC signatures are much shorter than RSA signatures. However, as mentioned by RFC 6605 [164], even though singing is significantly faster when using ECC than RSA, signatures validation is significantly faster ($\simeq$ 5 times faster in some implementations) when using RSA. For DNSSEC, this is apparently the most serious challenge when using ECC due to the latency of signature validation. Interestingly, Rijswijk-Deij et al. [265] show that even when using the optimized version of OpenSSL by CloudFlare[6] (in which ECDSA and RSA are sped up by a factor of 8 and 2, respectively), ECDSA is still 6.6 and 3.4 times slower than 1024-bit RSA and 2048-bit RSA, receptively, in terms of signatures validation. More importantly, the actual adoption of ECC by DNSSEC operators is very low [168, 264]; raising concerns in regards to backward compatibility if ECC were to be proposed for IoT devices.

For TLS, Gupta et al. [150] conducted a study to analyze the performance of ECC and RSA for SSL (Secure Socket Layer) on resource constrained devices. Their experiments

[6]https://ripe70.ripe.net/presentations/85-Alg-13-support.pdf

show that TLS handshake using RSA outperform ECC. For completeness, experimental measurements are conducted to compare ECC with CSProp. In the experiments, I used ECDHE-ECDSA (Ephemeral Elliptic Curve Diffie-Hellman key agreement with ECDSA signatures) [223] cipher suite with curve P-256. I run the experiments on three different IoT devices: Raspberry Pi 3 B+, Raspberry Pi B, and Raspberry Pi Zero W (see Table 5.3 for devices specifications). As shown in Table 5.4, TLS handshake using CSProp is faster by a factor of $\simeq$ 2.7x than when using ECC. This will impose an additional burden on end devices with the increased CPU load, especially if deployment of ECC-based TLS handshake accelerates. In 2014, Bos et al. [72] surveyed the adoption of ECC and found that only 10% of hosts supported ECC-based TLS. On a larger-scale study, the International Computer Science Institute (ICSI) Certificate Notary [171] reported that 11.5% and 2.4% of observed SSL\TLS connections used ECDHE-ECDSA with curves P-256 and P-384, respectively, in June/July 2018. We note also that a variety of attacks on ECC cipher exist [263].

5.5 Concluding Remarks

IoT and embedded devices, in general, are resources-constrained forcing designers to choose either security (e.g., by offloading security to gateway nodes) or performance (performing the expensive cryptographic operations required for end to end security). This research direction contributes a new cryptographic primitive, CSProp, that uses a low public exponent to reduce the computational load required by the end devices. Specifically, CSProp breaks down a validation operation into two phases. The first phase – which uses the traditional expensive cryptographic computations – is performed by a proxy servers (or a

propagator) (*e.g.,* DNS resolver, default gateway, or fog/edge server) that does not have to be trusted, meaning that the propagator does need need a PKI certificate and could be even be a role that changes within a peer-to-peer system. The second phase relies on the propagated low public exponent of RSA that offers a cheaper computational cost carried out by the constrained devices, while maintaining strong security guarantees. I reason about the security of CSProp and show that breaking it is of the same order of difficulty as breaking RSA.

The performance of CSProp is tested on resource-constrained devices to optimize the operation of two core security protocols on the Internet: DNSSEC, which protects the DNS system from attacks such as cache poisoning attacks, and TLS which is an essential protocol used to establish secure connections on the Internet. I show that the complexity (in terms of latency and energy consumption) of both the DNSSEC validation and the TLS handshake operations at the end devices is substantially reduced to the level where it is believed that it is practical to consider carrying them out at the end devices. The results also show that CSProp has advantages relative to using ECC cipher suite showing that CSProp outperforms ECC by a factor of 2.7.

Limitation and Future Work. A limitation of CSProp is that it helps only with operations that use the public key (signature verification, authentication, as well as encryption). Operations on the private key such as generating signatures and ciphertext encryption cannot use low exponents without compromising security [68]. Luckily, IoT and edge computing devices are clients of services requiring operations on public keys most of the time. For future work, a good direction would be to consider extending the benefits of CSProp to other

protocols, and considering alternative solutions that lower the overhead of private key based operations to make them more suitable for such resource-constrained environments.

Chapter 6

Analysing Cryptographic Overhead on IoT Devices

This chapter describes a comprehensive measurement study to characterize IoT devices with two components: (1) profiling existing devices to understand the cryptographic demands on IoTs; and (2) evaluating their performance on the new proposed primitive, CSProp (see Chapter §5 for more details), and compare the results with a widely used conventional cryptographic primitive which is RSA. Specifically, an empirical study is presented to measure the cryptographic overhead when using IoTs in realistic environments. I also discusses a potential deployment model for constrained environments. The results show that with a minimal amount of configuration, CSProp can be considered a natural fit with the typical communication practices in smart object networking environments.

6.1 Understanding Cryptographic Demands on IoTs

6.1.1 Overhead of Conventional Cryptography

Several studies have been conducted presenting the computational and energy costs of conventional cryptographic protocols on embedded IoT processors. For instance, Potlapally *et al.* [232] shows how fast the battery gets drained (more than twice) in the presence of encryption comparing to no encryption. They also show energy analysis of common cryptographic algorithms (such as RSA, DSA, and ECDSA) on a client device running Compaq iPAQ H3670 whose processor is clocked at 206MHz. In addition, a good body of work analyzes the performance of specific applications used in resource-constrained environments. For instance, Miranda *et al.* [213] analyzes the energy consumption of the Transport Layer Security (TLS) protocol transactions on a mobile device and found that more than 60% of total energy is consumed by TLS overhead. In standardized large-scale networks, developers often choose to sacrifice security to retain performance. For instance, in a typical IoT ecosystem, users communicate with IoT devices (*e.g.,* Nest temperature sensor or smartwatchess) through smartphone applications. Unfortunately, there are significant amount of efforts that have uncovered different types of vulnerabilities of various IoT devices due to lack of security [280, 269, 177, 161, 132, 107, 103, 101, 81, 73]. More recently, Zuo *et al.* [283] and Junior *et al.* [179] analyze the channels between several IoT devices and their corresponding smartphone applications. They found that these channels are often not encrypted/authenticated (presumably to avoid pitfall(2) above); thus, construction of exploits to remotely control the devices can *easily* succeed.

Table 6.1: Profile of the Data Exchanged Between an Iot Device and a Client in a Wireless Home Network

IoT Device	Operation	Total # of Captured Packets	DNS RRsets	RSA					AES 128-bit	
				TLS Handshake	Encryption	Decryption	Signing	Verifying	Encryption	Decryption
WYZE CAM V2 (camera)	Setup	897	68	32	168	0	104	90	232	180
	Pairing	25	2	3	9	0	0	2	5	7
	Live Streaming (2 Hrs)	1460282	108933	29535	30191	0	318	31239	964083	313664

6.1.2 Limitations of Existing Defenses

There are defenses against the emerging security problems in the IoT area. Some are proposed specifically to enhance the security of particular applications, such as the Bluetooth Low Energy (BLE) protocol [179, 130] and IoT apps [261, 133]. Others are classified as lightweight cryptography proposals which usually involve expensive cryptographic processing [237, 39] or proven to be vulnerable to different attack surfaces [279, 118, 32]. There is also extensive body of work considering proxy-based defenses for IoT environments such as ALPKA [74], AKAPR [222], proxy-based end-to-end key establishment protocol [231], and many others [119, 66, 52, 178, 114]. However, all of them are constructed using a trusted proxy; an assumption that raises many scalability and security issues [277, 36]. To tackle this problem, the concept of proxy re-encryption was introduced [66], but it still relies on a semi-trusted proxy to transfer ciphertext from sender to receiver.

6.2 An Empirical Study

To increase the motivation, the cryptographic overhead occured is analyzed when an IoT device is used in a home-based environment. The testbed consists of Wyze Cam

V2 (an Amazon choice smart home camera [24]) connected to a wireless home network via an Arris router [50]. The wireless network uses WPA2-AES-128-bit protocol [199] (know as WPA2-Personal) for encryption and a Pre Shared Key (*e.g.,* an 8-character password) for authentication. The client is a Wyze app downloaded to an iPhone X running ios 13.3.1. The results show that the camera uses cryptographic operations intensively. For instance, as shown in Table 6.1, live-streaming for a period of 2-hours required more than 60K (*i.e.,* $\approx$ 4.2% of packets exchanges) of RSA public-key operations. Furthermore, to support end-to-end data protection, the results show that transmitted data packets between the camera and the app were frequently encrypted and decrypted using the AES protocol. However, since it is a WPA2-Personal network, this setting secures the network only against outsiders. In particular, this network is vulnerable to Man-in-the-Middle (MitM) attacks if an adversary is an insider who already knows the PSK key. Consequently, she/he would be able to derive the same secret keys —*i.e.,* Paiwise Transient Key (PTK) and Group Temporal Key (GTK) used to encrypt/decrypt unicast and multicast data packets, respectively, between clients and their associated access point (AP)—that are shared among all users and generated during the 4-Way Handshake protocol [146][1]. The results show that $\approx$ 87.5 of data packets are vulnerable to this type of attack. What is worse, in case Domain Name System SECurity Extension (DNSSEC) [47] validation is enabled, more cryptographic operations are required; increasing the computational burden on the IoT camera since chains of DNS RRsets signatures need validation.

[1]Note that using 802.1X [95] for authentication, which is used in WPA2-Enterprise networks, closes this vulnerability. This is because each user is assigned a unique PSK key.

6.3 Comparison Between CSProp and Conventional Cryptography: Arduino Measurements

This section describes a comprehensive measurement study of CSProp to secure a resource-constrained smart object device. We further experimentally compare CSProp with conventional cryptography on a special type of IoTs: a microcontroller.

6.3.1 Experimental Setup

Selected Hardware. Due to the resource-constrained nature of popular microcontroller-based IoT hardware development boards (*e.g.,* LightBlue Bean+ [28] and Arduino Uno [26]), it is usually challenging to execute the necessary RSA cryptographic operations on the main processor. For instance, Sethi *et al.* [49] shows that generating RSA signatures on Arduino Uno takes 25.08 secs, 3.33 mins, and 26.4594 5mins when RSA key sizes are 512, 1024, and 2048 bits, respectively. Furthermore, to maintain acceptable throughput and energy consumption, such devices often require hardware acceleration.

For implementing CSProp, I choose Arduino MKR WiFi 1010 [25] as the test platform since it meets the throughput and energy consumption requirements of current cryptographic primitives and schemes including RSA [207]. This Arduino is based on the SAMD21 microcontroller, a 32-bit low power ARM MCU processor with a clock speed of 48 MHz, 32 KB of SRAM, and 256 KB of flash memory. The hardware acceleration engine for cryptographic algorithms supports the hashing algorithm SHA-256 which is used in the prototype since it is recommended by RFC8446 [240] for TLS and by RFC6781 [194] for DNSSEC. Using an 8-bit platform with sufficient RAM size is also viable to generate good

performance results without hardware acceleration as has been shown by Gura et al [151]. However, 32-bit microcontrollers are much more easily available, at same lower costs, and; more importantly, are more power efficient to run RSA cryptographic operations.

Network Connectivity. Arduino MKR WiFi 1010 features Wi-Fi, Bluetooth®, and Bluetooth Low Energy (BLE) interfaces. The Wi-Fi interface uses the WiFiNINA library that supports IEEE 801.11 b/g/n modes of operation and WEB, WPA, WPA2 Personal, and WPA Enterprise encryptions. It provides several buses and communication interfaces that fit IoT sensor nodes, such as UART, SPI and I2C. In addition, since this Arduino is a dual processor device, all interfaces can be used at once on the board.

Energy Consumption. To obtain the energy consumption values, the Watts Up? Pro AC meter [233] is used. This power meter supports several displays, computer software, and PC interfaces. It can automatically generate data graphing, and its data logger function records all data into non-volatile memory.

Software. For programming the Arduino MKR WiFi 1010 modules, the Arduino-IDE [45] release/v1.8.12 is used. The Arduino-IDE is the official development framework for Arduino devices. It is installed on a Microsoft Windows 10 operating system. For selecting a potential cryptographic library, I surveyed a set of possible source codes and performed an initial analysis of how well they fit the Arduino environment. Based on the analysis, I found three libraries that provide RSA as a PKI algorithm. The first is the AVRCryptoLib [27] library, and it is written in an assembly language to reduce the size and optimize the performance. The other two libraries are Relic-toolkit [44] and Cryptographic-Protocols-Arduino-and-PC [80], and they are written entirely in C. I choose the Cryptographic-Protocols-Arduino-

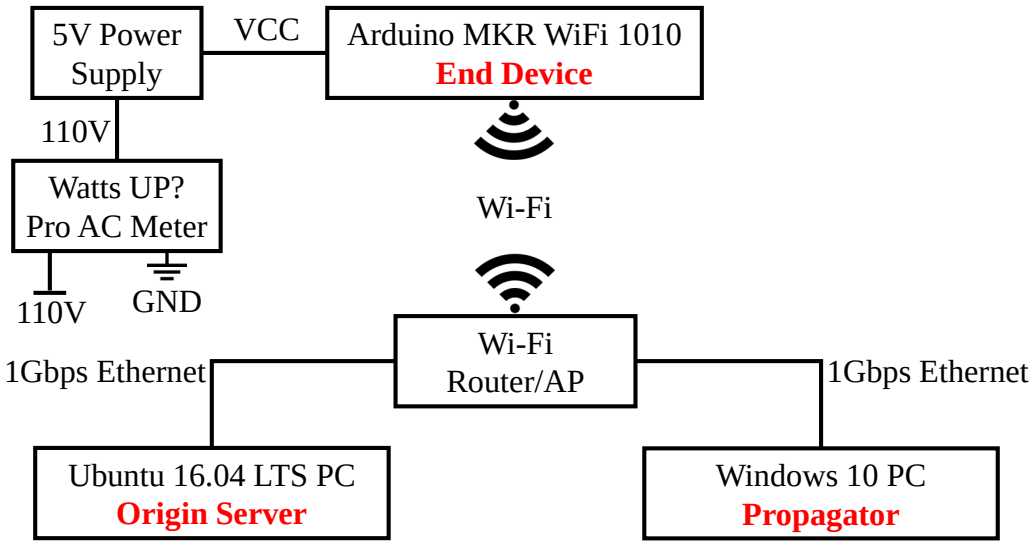


Figure 6.1: Testbed Architecture Configurations

and-PC library since it is more flexible to be customized and implemented than the other two libraries.

Latency. One of the main criterias to measure the performance of a cryptosystem is to measure the latency (*i.e.,* execution time) of its cryptographic operations. The `millis()` function [3] is used which is included in Arduino Language. This function returns the number of milliseconds passed since the Arduino board began running the current program.

Memory Usage. Arduino MKR WiFi 1010 has two types of memory: a ROM (flash) memory and an SRAM. The compiler of the Arduino IDE automatically calculates and displays both the amount of ROM and SRAM memory that a sketch will use after compiling. However, the memory footprint of the SRAM does not include local variables, but the compiler does display the amount of SRAM remained for local variables.

Testbed Setup. To evaluate the performance of CSProp on the selected device, the testbed is configured as follows. I use a Dell PC running Ubuntu 16.04.6 LTS operating system as

an origin server. This server is responsible for generating the RSA keys needed to perform all the cryptographic operations. The propagator is a Dell PC running Windows 10 Home edition operating system.

An Arris router [50] is used as the central communication device. The PCs are connected using 1Gbps Ethernet cables Ethernet, while the Arduino is connected through its Wi-Fi interface. For powering up the Arduino, a dedicated 5 V/2 A power source is used. Since the Arduino has a Li-Po charging circuit, it is possible to run it on a USB port or batteries. However, in order to provide a stable and constant energy source, a dedicated power supply is used instead. Figure 6.1 shows the main components of the testbed.

6.3.2 Performance Analysis

For evaluating CSProp, we are particularly interested in two RSA public-key operations: verification for signature propagation and encryption for ciphertext propagation. The performance measurements for CSProp and traditional RSA public-key operations are provided. The measurements reflect the performance of three different RSA key lengths: 512, 1024, and 2048 bits. The exponents $e = 65463$ and $e_{\mathsf{low}} = 3$ are used as the public exponents for CSProp; the full key e is used between the origin server and the propagator, and then e_{low} is used after signature propagation. I compare CSProp with traditional implementations of RSA public-key operations. CSProp is also compared with current real-world configurations where the public exponent is $2^{16}+1 = 65537$. As mentioned earlier, SHA-256 is used in the experiments since it the recommended hashing algorithm for different networking protocols. The code size is ≈ 8 KB and the message size is 128 Bytes for all the test cases. It is also worth noting that in the implementation I considered basic mathematical

operations (*e.g.,* exponentiation and multiplication) without using any optimizations (*e.g.,* montgomery multiplication and optimized squaring as described in [192]). All results are from 50 runs (the values are almost identical) for each scenario, then the average is taken to represent each data point.

6.3.3 Results

The results are summarized in Table 6.2. For all RSA key sizes, CSProp outperforms the traditional RSA public-key operations in all scenarios. For instance, the execution time for CSProp-encryption is more than 57 and 81 times faster than traditional RSA encryption when $e = 65537$ and $e = 65463$, respectively. For CSProp-verification, the conclusions are very similar. The results show huge differences for the same security level when comparing CSProp and traditional RSA public-key operations, being CSProp a better alternative for resource-constrained devices since it presents better energy consumption. For instance, CSProp provides efficient reductions by $36x$ and $42x$ for encryption and verification,respectively, when $e = 65537$. Comparing the memory results, it is interesting to note that the modular exponentiation of CSProp requires little memory (a crucial design decision in designing lightweight cryptosystems) compared to a traditional RSA implementation as are the cases when $e = 65537$ and $e = 65463$ which significantly increase the memory usage.

More importantly, the results also present interesting findings when different key sizes of the same algorithm are compared. The most interesting finding is that PKI cryptography is possible on resource-constrained devices.

Table 6.2: This table shows a Comparison of CSProp Vs. two typical implementations of traditional RSA public-key operations. The performance is measured based on latency (in ms), memory footprints (in bytes), and energy consumption (in mJ)). Note that the memory usages for SRAM and ROM are added to represent the total memory footprint.

	CSProp ($e = 65463$, $e_{low} = 3$)					
	Encryption			**Verification**		
Key Size (bits)	**Latency (ms)**	**Memory Footprint (bytes)**	**EC (mJ)**	**Latency (ms)**	**Memory Footprint (bytes)**	**EC (mJ)**
512	11	42	15	15	49	21
1024	29	69	23	35	82	36
2048	61	125	39	71	134	48
	Traditional RSA ($e = 2^{16} + 1 = 65537$)					
	Encryption			**Verification**		
Key Size (bits)	**Latency (ms)**	**Memory Footprint (bytes)**	**EC (mJ)**	**Latency (ms)**	**Memory Footprint (bytes)**	**EC (mJ)**
512	634	320	540	915	441	882
1024	1665	552	828	2135	738	1512
2048	3502	1006	1404	4331	1206	2016
	Traditional RSA ($e = 65463$)					
	Encryption			**Verification**		
Key Size (bits)	**Latency (ms)**	**Memory Footprint (bytes)**	**EC (mJ)**	**Latency (ms)**	**Memory Footprint (bytes)**	**EC (mJ)**
512	891	504	615	1290	539	1071
1024	2349	828	943	3010	902	1836
2048	4941	1500	1599	6106	1474	2448

As it can be observed, the results when $e = 65537$ are to some extent better than those when $e = 65463$ in the case of traditional RSA operations. The reason is that 65537 is a Fermat number. Fermat numbers are $2^n + 1$ primes, and they are recommended [243] since only the first and last bits of their binary representation are ones (100...001); a feature expedites computations on computers.

Comparing the general performance of the ciphers suites tested, it can be concluded that for the same security level as proven in Section §5, CSProp always outperforms traditional RSA public-key operations in both latency and power consumption.

Chapter 7

Internet Filtering in the Kingdom of Saudi Arabia: A Longitudinal Study

Several countries apply local restrictions on access to information on the Internet, which they deem inappropriate (*e.g.,* pornography). I refer to this type of access restriction as *Internet filtering*[1]. In this study, I focus on answering three questions: (a) what content is filtered? (b) how is filtering implemented? and (c) how does filtering evolve over time in response to geopolitical conditions?

The Kingdom of Saudi Arabia is often considered to be one of the most conservative countries in the world. The government manages the access to the Internet with a filtering system to restrict content it deems unacceptable or inappropriate. These filtering decisions

[1]Note: Although some filtering is used for censorship in some cases, I choose to use the term filtering to keep my focus on the technology.

are motivated by the government's desire to protect the values of the Saudi society (which center around Islam, the official religion of the country), in addition to implementing national security and public safety policies [91]. The debate over how web content should be filtered, what constitutes inappropriate material and whether (and how) the public should be protected from remains controversial. On the one hand, many organizations criticize the level of Internet filtering in Saudi Arabia. For instance, Freedom House characterizes Saudi Arabia as "not free" [21], and it is ranked 172 out of 180 countries by Reporters Without Borders as one of the top violators of press freedom [20]. The same organization also published a report in January 2016 which ranks Saudi Arabia as one of the "15 enemies of the Internet" since 2005 [12]. On the other hand, we see many domestic organizations that do support Internet filtering. For instance, the Ministry of Media provides regulations for digital publishing activity [212, 211] which strictly prohibit any content that violates the provisions of Islamic law, breaches the national security, incites violence among citizens, or violates copyrights law. The Ministry of Interior (MOI) launched an electronic service [210] to help citizens report any type of cyber crimes including production of websites violates public moral (such as pornographic and gambling sites) or impinges on public order and religious values as stated in the Anti-Cyber Crime Law [92]. Many individuals and non government-linked organizations support the filtering service [102, 154, 37]. Internationally, the Safer Internet Day (SID) [4] is an organization that provides an education and awareness-raising effort spanning more than 100 countries including Saudi Arabia [93].

In the last years, Saudi Arabia has undergone significant sociopolitical changes, combined with significant political events in the region, which seem to have brought forward

a more progressive approach regarding access to information. I provide some highlights of such events between 2018 and 2020. This period of time coincides with the start of the execution of Saudi Arabia's national Vision 2030 and National Transformation 2020 programs [247] that were announced in April 2016. The two vision documents were nationally adopted as a roadmap and implementation guidance for economic and developmental investments and projects across all fields of endeavor in the country, with the stated aim of creating a tolerant country with Islam as its constitution and moderation as its goal. Interestingly, the two vision documents introduce a series of sweeping social reforms. As evidence that they are being implemented, on September of 2017, a royal decree lifted the ban on women driving that was enacted in 1957 [18]. Additionally, according to the Kingdom's General Authority for Entertainment (GAE) [141], in 2018, Saudi Arabia has allowed the operation of movie theaters for the first time in decades. In addition, during the same year, the Saudi government allowed international artists to perform musical concerts as part of public life, including the participation of women; both of these (public music performances, and women participation) are surprising and progressive moves in the context of a Saudi society that has never experienced either before. At the same time, there have been a number of geopolitical developments, such as rising tensions with Qatar, Iran, and Turkey.

Given both these significant social changes, and geopolitical events, a natural question is whether such events translate into changes in the Internet filtering policies. There are a few studies (that are older, and less comprehensive than my work) focusing on Saudi Arabia, while there are several studies of Internet filtering in other countries, which are reviewed in Chapter §2.

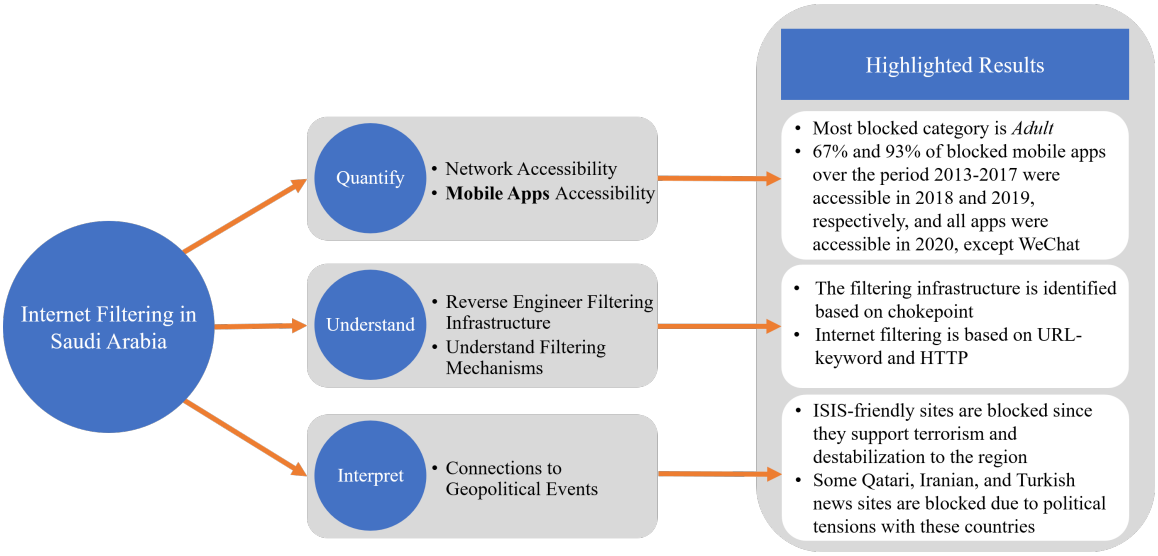


Figure 7.1: Overview of the key questions, contributions, and findings of my work

To the best of my knowledge, this research direction represents the first systematic, longitudinal study of Internet filtering in Saudi Arabia. The study spans the period from March 2018 to April 2020, with three separate measurements, one in each calendar year. My goal is to answer the three questions mentioned earlier. I pursue a three pronged strategy summarized in Figure 7.1: (1) Quantification: I measure website network accessibility as well as mobile application accessibility (specifically, those used for voice/video communication); (2) Understanding: I seek to reverse engineer the filtering infrastructure and understand the different filtering mechanisms employed; and (3) Interpreting: I explore whether societal shifts in response to the moderation visions, as well as geopolitical events influence the filtering policy over time. Each of these directions are discussed in more detail below.

a. Quantification: What is filtered? I provide a fairly extensive study on the accessibility of both websites and mobile apps from within Saudi Arabia and its evolution over

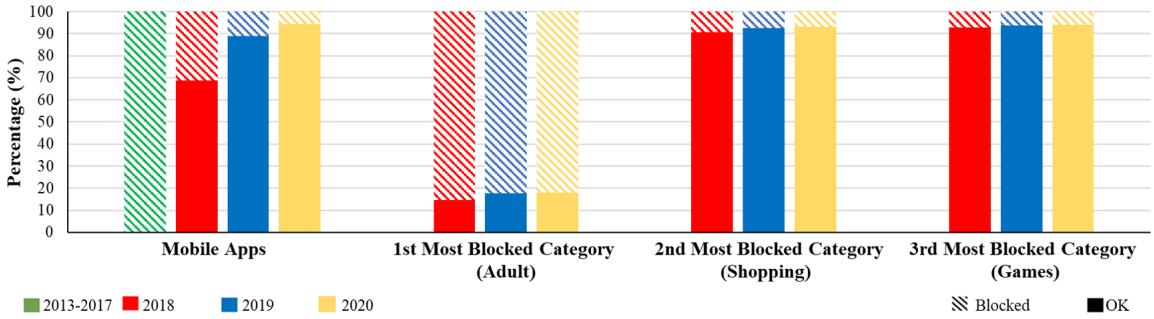


Figure 7.2: Overview of the extent of filtering over time per category. We observe a significant relaxing of the filtering rules for both Internet and mobile apps. Note that I did not measure the period 2013-2017, but rely instead on personal experience and public sources. The bars for mobile apps represent the percentage of the apps that were tested at that time period. For instance, in 2018 I tested 16 apps while in 2019 and 2020 I tested 18 apps.

time.

- Network accessibility: I probe the most popular websites worldwide according to the Top Sites lists overall and by category published by Alexa [34]. I collected the top 500 websites in 18 different categories [35]. Interestingly, as shown in Figure 7.2, we find that the most blocked category is *Adult* in which 85.4%, 82.2%, and 82% of sites are blocked in 2018, 2019, and 2020, respectively. We also find that the second most blocked category is *Shopping* in which most of the sites are either belong to convenience stores or apparel retailers that do not have a branch inside Saudi Arabia (*e.g.,* The Home Depot, CarMax, Bloomingdales, and Kohls) or sell products that are considered illegal (*e.g.,* alcohol and guns).

- Mobile applications accessibility: I conduct systematic experiments on 18 of the most

popular mobile social network applications worldwide and in the middle east including Facetime, Tango, Viber, Line, SOMA, and WeChat. The results show that Saudi Arabia is cautiously yet decisively opening its digital borders. As shown in Figure 7.2, the results show that 80% of the blocked mobile apps in 2018 have become accessible in 2019 while all of them turned to be accessible in 2020, except WeChat which is still debatable. This perhaps reflects that the kingdom is moving towards moderating regulations on Internet filtering.

b. Understanding: How is filtering implemented? I go below the application layer and identify the point and mechanisms in the communication interaction, where the filtering takes place.

- Reverse engineer filtering infrastructure: the general topology of the internet filtering is discussed in some of the official documents from the Ministry of Information. I conduct experiments to independently reconstruct the network topology of the filtering system.
- Understand filtering mechanisms: my results show that Internet filtering in Saudi Arabia is based on HTTP filtering augmented with TLS filtering for connections using HTTPS. I believe that this research direction is the first to identify TLS level filtering [267].

c. Interpreting: How is filtering affected by geopolitical events? I finally go at the political and policy level and examine the manifestation of real-world events on filtering.

- We find that ISIS-friendly sites are blocked due to the fact that ISIS supports terrorism

and destabilization to the region. We also find that more news sites got blocked. For instance, some Qatari, Iranian, and Turkish news sites got blocked in 2017, 2018 and 2020, respectively, amid continued political tensions with these countries.

To increase the confidence in my observations, I performed network measurements inside Saudi Arabia from *four* major cities spread across the country (Riyadh, Jeddah, Makkah, and Al-Khobar) and spanning *three* major Internet Service Providers (ISPs) in the country. This methodology differs from prior studies of filtering in other countries, which overwhelmingly rely on VPNs or PlanetLab nodes [267, 276, 208].

The remainder of this research direction is organized as follows. I present a brief history of Internet filtering in Saudi Arabia in Section §7.1. I present some background on Internet filtering in Section §7.2. I present the methodology I employ in the measurement study in Section §7.3. I present the results of the study in Section §7.4, including some analysis of the impact of local and regional geopolitical events on the evolution of the filtering policies. I present my analysis of the filtering infrastructure in Section §7.5, and I discussed related work in Chapter §2. Finally, I present some concluding remarks in Section §7.6.

7.1 History and Background

The history of the Internet in the Kingdom begins when King Fahd University of Petroleum and Minerals (KFUPM) connected to the Internet in 1993. This initial system used two dedicated Domain Name System (DNS) servers for name resolution [30]. After being intensively used by the academic sector, the Internet was brought to the public in

1999 after the Council of Minsters officially issued Resolution No. 163 [184]. It gave King Abdulaziz City for Science and Technology (KACST), located in Riyadh, the authority to create the Internet Services Unit (ISU)[2] and to carry out oversight over the licensing, deployment and management of the Internet in the country. Since then, the unit, in cooperation with the Communications and Information Technology Commission (CITC), is responsible for providing Internet service in Saudi Arabia. Despite this late adoption, especially in terms of connectivity, the number of Internet subscribers increased rapidly: CITC estimates that the number of Internet users has grown from nearly 1 million in 2001 to 3 millions in 2005 [86]. At the end of 2018, the number of users is more than 27 millions which represents 83.4% of the population in Saudi Arabia [140]. In addition, in April 2016, the Crown Prince Mohammad bin Salman Al-Saud announced the Vision 2030 and National Transformation 2020 programs [100] that gave high priority to the Communication Information and Technology (CIT) sector, which has a goal of ultimately providing 90% broadband coverage across the different regions of the country.

Internet filtering in Saudi Arabia is managed by a central and standardized system located in KACST. All ISPs direct their web traffic to an ISU proxy server which keeps a log of user activities. If there is an ISP-level firewall or other network security functionality, it is legally required not to bypass this proxy. Although there are always ways to circumvent Internet filtering [267, 252, 83, 136], this type of practise is considered a cyber crime [92].

Website filtering uses two sources of information [90]:

1. Commercial list: The CITC has contracted with an unnamed international company

[2]A government organization that takes responsibility of organizational and administrative aspects of the Internet. See https://www.kacst.edu.sa/eng/ScientificServices/ ISU/Pages/landing.aspx

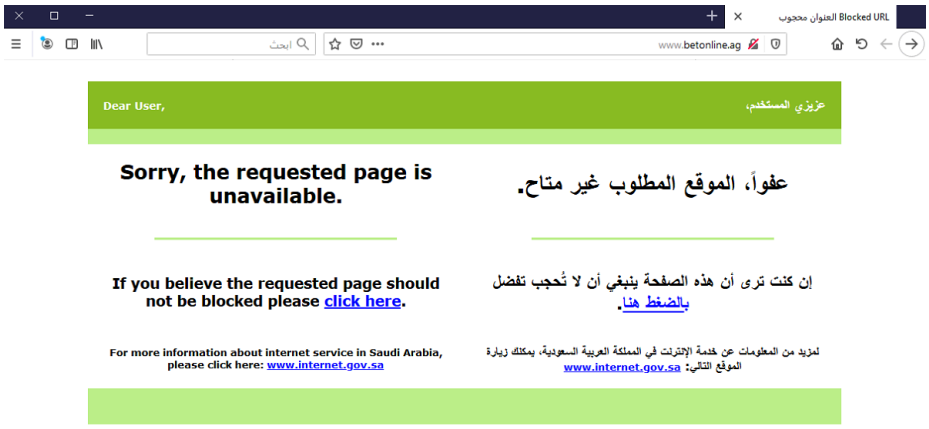


Figure 7.3: General filtering warning page

specialized in website ranking to obtain their own list of websites categorized into more than 90 categories. Those which related to pornography, gambling and drugs are summarily blocked; an effort to provide a family safe Internet. The list is updated by the company on a daily basis, and there is a continuous line of communication with the company to correct errors related to websites classification.

2. Saudi-Arabia-specific list: It is an internal list prepared by CITC. The websites are blocked based on requests by regular users and specialized authorities [88] after reviewing them and ensuring that they contain materials inappropriate to the Saudi society. The CITC reported that 92.80% of these websites are related to pornography while 2.77% are related to gambling, sorcery, drugs, etc.

Unlike other countries who transparently block web pages without informing users (*e.g.,* Bangladesh, Russia, India, China, Turkey and Malaysia) [267], Saudi Arabia makes blocking explicit. While the list of prohibited websites is not publicly available, any re-

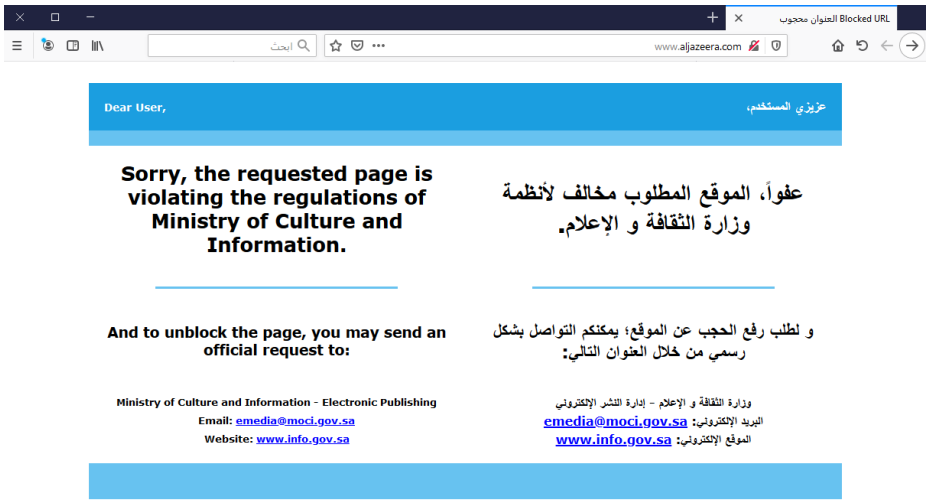


Figure 7.4: Filtering warning page by the Ministry of Culture and Information

quest to a blocked website causes the user to be redirected to a web page owned by ISU. For instance, Figure 7.3 shows a warning page in response to visiting `www.betonline.ag` (a gambling website). Figure 7.4 shows a different warning page in response to visiting `www.aljazeera.com` (a news website) which has been blocked due to political tensions with Qatar where Aljazeera is headquartered. CITC also receives requests to block or unblock websites from customers. For instance, more than 8 thousand unblock requests were received in 2016 [87], with 7% of these ultimately accepted. On the other hand, more than 900 thousand requests to block websites were received with 56% of the requests resulting in a block. This number increased to more than 1 million by the end of 2017 [89]. Overall, the CITC has handled over 6 millions blocking requests since 2008.

There are many other countries where Internet filtering is present due to historic and domestic issues (*e.g.,* Nazi websites in Germany [127], pedophilia in the European Union [98], violation of copyright law in France [149]). In Saudi Arabia, national security and cultural beliefs are the origin of the government's concerns in regards to Internet ac-

cess [91]. The government has been practicing Internet filtering since 2001 when the Council of Minsters issued a resolution outlining the basis for content filtering [185]. Accordingly, the ISU published a "black-list" of prohibited websites [282]. Here are some examples of blocked content:

- Two episodes of the "American Dad!" TV series were blocked for having a scene showing a negative portrayal of Saudis [244].
- The Ministry of Media blocked "Pirate Bay" and "Torrentz.eu" websites for distributing copyrighted materials [246].
- To promote Islam being a peaceful religion [275], any websites that promotes the so-called "Islamic State" of Iraq and Sham (ISIS) are called to be banned [31].
- Some social network applications, such as Whatsapp and Skype, were banned in 2013 for violating regulatory requirements [5].

7.2 Overview of Internet Filtering Mechanisms

In this section, I review some of the most commonly deployed Internet filtering mechanisms, which can enable filtering at different levels of granularity. Internet filtering refers to a multitude of technical policies that can be employed to prevent users from accessing specific content or Internet-connected machines; the policies can vary from filtering or blocking all connections in a particular country to micro-focused strategies blocking specific websites, servers, and even words. These policies rely on a number of mechanisms that interfere with the user's ability to access these resources. Specifically, given the steps needed to establish a connection to a website from a browser, filtering can interject to

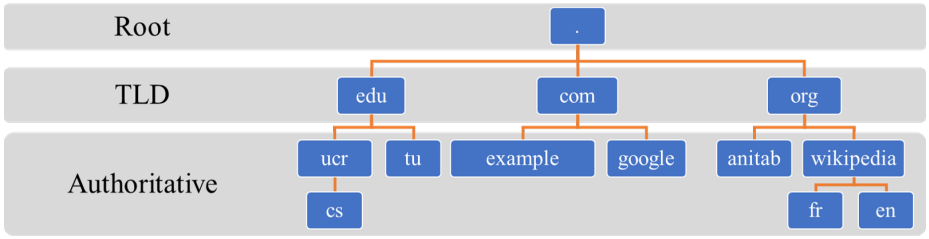
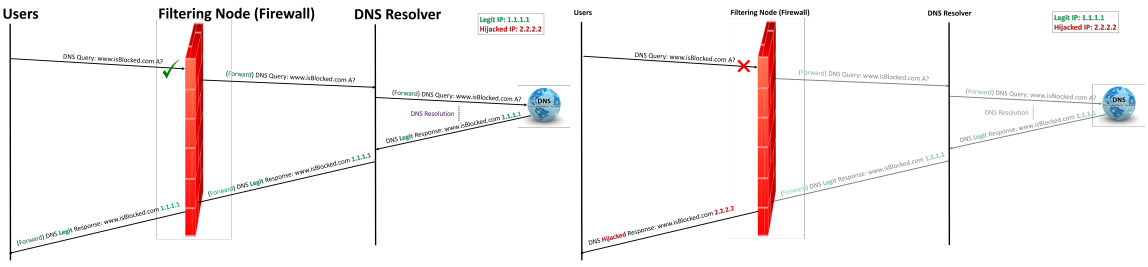


Figure 7.5: Hierarchy of DNS name servers

interfere with any of these steps (e.g., TCP connection establishment). Note that it is possible to use a combination of mechanisms to implement an overall filtering policy.

7.2.1 DNS-level Blocking

Accessing an Internet connected service (e.g., a web site) starts with a Fully Qualified Domain Name (FQDN) address, specified as part of a Universal Record Locator (URL) in a browser address bar (*e.g.,* `http://www.example.com`). The FQDN is the portion of the URL that fully identifies the domain name of the target server (*e.g.,* `www.example.com` without the `http://` prefix). The Domain Naming Service (DNS) protocol provides a resolution service to map FQDNs to their corresponding Internet Protocol (IP) addresses [46]. DNS is supported by a hierarchical infrastructure which contains a set of distributed name servers (an example is shown in Figure 7.5). At the top of the hierarchy, there are 13 root servers that are geographically distributed at a global level. At the next level, there are a set of Top-Level Domain (TLD) servers which service queries for top level domains (*e.g.,* `.edu`, `.com`, and `.org`), and at the bottom there is an embedded hierarchy of authoritative name servers that hold the translation from the FQDN for the part of the address space they manage to the corresponding IP addresses.



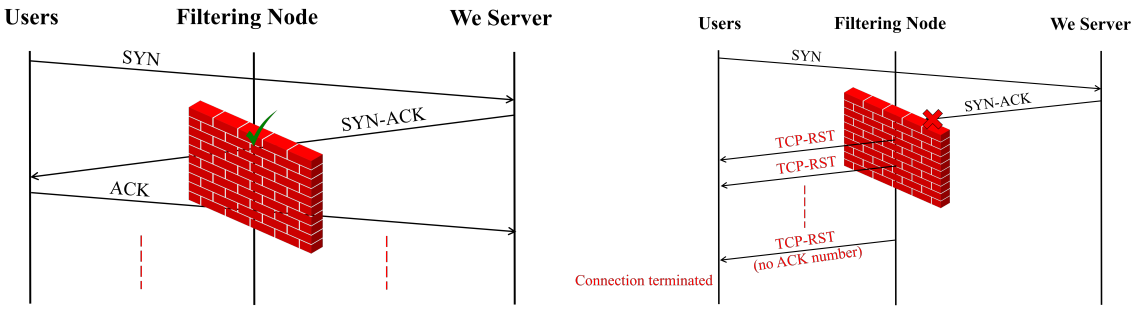
(a) OK (b) Blocked

Figure 7.6: DNS-Level Blocking

DNS queries from clients are processed by resolvers that can walk the hierarchy until they reach the authoritative name servers responsible for the FQDN being resolved. In countries where authorities have control over DNS resolvers, they interject in the resolution process to manipulate translations and redirect users from accessing a filtered site (see Figure 7.6-a and Figure 7.6-b). In other words, they can prevent the translation of FQDNs to their corresponding IP addresses, and instead they direct users to another server under their control, for example, to display a message indicating that the target server is blocked. For instance, in Iran, when a client sends a DNS query to access a banned site, instead of receiving the legitimate IP address, the client is directed to a private IP address (`10.10.34.34`) that is controlled by the filtering module [51]. Similarly, extensive work show evidence that China's Great Firewall (GFW) uses this type of filtering [205, 201, 11, 126].

7.2.2 IP Address Based Blocking

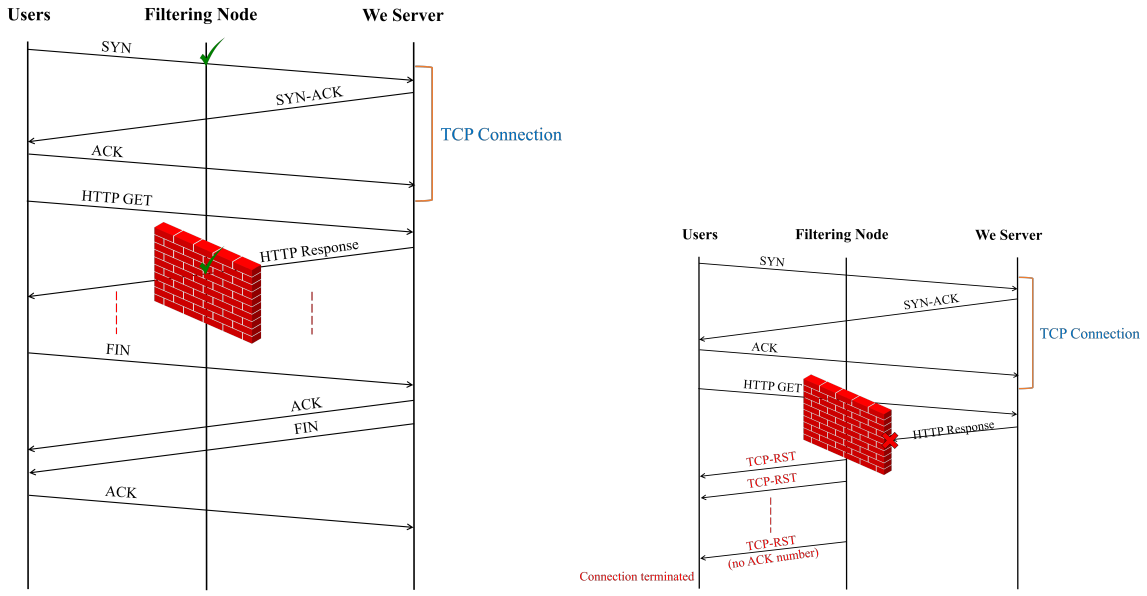
After DNS resolution, the next potential intervention point available for filtering is at the level of TCP (or UDP) connection establishment, which uses IP addresses to identify



(a) OK (b) Blocked

Figure 7.7: IP Address Based Blocking

the destination of the connection. Governments (or enterprise) maintain a pre-defined list of IP addresses belongs to banned sites. This blacklist is usually handed to Internet Service Providers (ISPs) to execute the filtering [276]. Basically, when a client requests access to a forbidden site, the ISP will prevent connection establishment (by dropping the `SYN` or `SYN/ACK` packet). For instance, in China, requests to banned sites are intercepted by GFW servers and then responded to by spoofed `TCP-RST` packets (see Figure 7.7), forcing clients to terminate the TCP connection [267]. Filtering based on IP addresses can also be used to block access to entire subnets. For instance, the Syrian authorities filter IP addresses belong to specific geographical regions (*e.g.,* Israel) [79]. Blocking using IP addresses has the advantage of working even when the connection is encrypted, or when the users bypass the use of DNS.

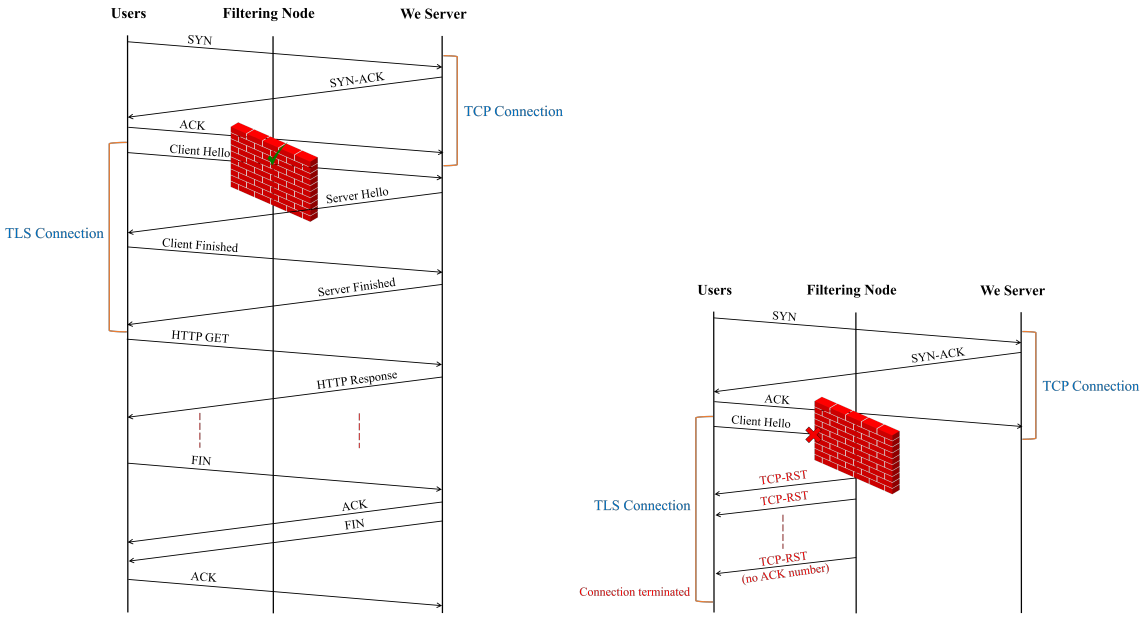


(a) OK (b) Blocked

Figure 7.8: HTTP Filtering

7.2.3 HTTP Filtering

DNS-level and IP addresses based blocking interfere with the connection based on the destination site address or name. One way to circumvent this type of blocking is to change IP addresses and DNS records of the blocked servers to evade filtering until the lists are updated again. As a result, another filtering intervention uses HTTP-level filtering. This mechanism typically uses one of two cases: (1) FQDN and (2) General keyword. The FQDN filtering checks the URL string (typically in HTTP GET requests) after the `http://` prefix, while the general keyword filtering checks the URL string against a list of forbidden keywords. Note that beyond checking the FQDN, keyword filtering can enable filtering based on the occurrence of the keyword in the URL.



(a) OK (b) Blocked

Figure 7.9: TLS Filtering

This intervention is typically applied to HTTP, either during initial GET request or when responses are received. When requests are filtered, the filtering system can force the connection to reset or timeout then displays an error to the client. In this scenario, requests never reach the destination web server. Similarly, when responses are filtered, typically HTTP filtering is applied forcing the connection to be terminated (see Figure 7.8). Previous work has shown that Pakistan [220] and China [267] use this type of filtering. Interestingly, Verkamp and Gupta [267] show evidence that the filtering node in Saudi Arabia responds back with a warning page upon filtering an HTTP request. I provide a deeper analysis from vantage points inside the country and found technical details that contradict with their findings.

7.2.4 TLS Filtering

Since HTTP filtering requires deep packet inspection to identify the keywords inside the payload of the packet, if encryption is used (i.e., HTTPS rather than HTTP), the keyword is not available and HTTP filtering fails. This gives users and websites a simple way to bypass filtering. To prevent this, additional filtering can be implemented at the TLS handshake level. After the TCP 3-way handshake succeeds, when a client tries to establish a TLS connection with a blocked website, the filtering node sends `TCP-RST` packets after the `Client Hello` message forcing the TLS connection to be terminated as shown in Figure 7.9. If TLS filtering is implemented to complement HTTP filtering, if HTTPS is used, TLS filtering can prevent the establishment of the encrypted session. We believe that this work is the first to identify this type of filtering.

7.2.5 Other Strategies

These strategies—DNS tampering, IP blocking, HTTP, and TLS filtering— are the widely used filtering methods. There are other fine-grained techniques that can be used. For instance, general deep packet inspection can enable sophisticated filtering based on packet characteristics and content. Another method commonly used is traffic shaping in which filtering authorities delay access to blocked websites. Port numbers can also be blocked, regulating and restricting the use of specific web services (*e.g.,* email servers, or instant messages). I did not observe such mechanisms in action during my study.

7.3 Methodology

In this section, I explain the tools, measurement methodology and the data collected in my experiments. I start by presenting some ethical considerations that I contemplated as I undertook this study.

7.3.1 Ethical Considerations

While it is out of the scope of this paper to further discuss the debate about Internet filtering, I acknowledge that this study may be beneficial to entities on either side of the filtering. Indeed, my analysis helps understand the technical aspects of the actual filtering ecosystem in Saudi Arabia.

To avoid legal complications, I discussed the scope of this study with a Saudi high-ranking government official, who is an expert in Saudi Arabian law, and he confirmed that the study does not violate the Saudi Arabian law. Clearly, Internet filtering is a sensitive topic, and I had to consider whether the experiments would violate not only ethical considerations, but also any laws or regulations in Saudi Arabia. Article 6 in the Anti-Cyber Crime Law of the country [92] states that the production of any artifacts that would undermine public order is strictly illegal. In the context of my work, I had to verify that this measurement study does not present mechanisms to bypass the filtering which would make it illegal under article 6.

I took precautions to ensure that my study would not jeopardize any individual within or outside Saudi Arabia. I never disclosed the personal information of my anonymous volunteers, nor did I re-distribute or otherwise share the detailed experimental logs data

(now or in the future). I only analyze aggregated information that neither exposes details of the network or any identifiable information with respect to my measurement points. Additionally, since Internet filtering in Saudi Arabia is evident and explicit (as shown in Figure 7.3 and Figure 7.4), the act of probing blocked sites is legal.

7.3.2 Measurement Methodology

To identify the scope of Internet filtering inside Saudi Arabia, I tested the reachability of the most popular websites worldwide according to the Top Sites lists overall and by category published by Alexa [34]. The top 500 websites are collected in 18 different categories [35]: *Adult*, *Arts*, *Business*, *Computers*, *Games*, *Shopping*, *Society*, *News*, *Regional*, *Reference*, *Sports*, *Global*, *Saudi Arabia*, *Home*, *Health*, *Recreation*, *Kids & Teens*, and *Science*. The experiments were conducted three times, roughly one year apart between March 2018 and April 2020. For each iteration of the measurements, I got the updated lists of the top 500 websites ending up with three lists for each category corresponding to the three measurements. We find that the lists remained almost identical (less than 3% of change) across the three years and the changes were typical at the very bottom of the list.

In addition, I tested the availability of 18 mobile applications, including Line, Skype, and Facetime, as I discuss later.

7.3.3 Tool Overview

To conduct my experiments, I developed a tool which was inspired by an earlier filtering tool, Samizdat [220]. Samizdat was used in a study of Internet filtering in Pakistan in 2013, and this effort is reviewed in Chapter §2. Despite the functionality of the tool, I

needed to develop significant new capabilities to go to the level of granularity that wanted in my study.

For each website in my lists, I carry out the following measurements.

1. DNS Filtering: First, the tool performs a DNS lookup using UDP and records the IP address in the response packet; otherwise, the retrieved error code (*e.g.,* `Timeout`, `SERVFAIL`, `REFUSED`, `NXDOMAIN`, etc) is recorded. It then performs the same test using TCP and records the results.

2. IP Address Filtering: If the website is successfully resolved in the first test, the tool initiates a TCP connection using a stream socket to the IP address and port `80`. If the connection is established, the test is recorded as successful; otherwise, it is recorded as a failure.

3. HTTP Filtering: This experiment is divided into two phases. In the first phase, I check if there is direct HTTP filtering of the FQDN in the `GET` request. Specifically, the tool tries to establish an HTTP connection and sends a `GET` request to the website. Both the response and returned code are recorded. In the second phase, using a non-blocked website (*e.g.,* `www.google.com.sa`), the tool appends the URL of the website we want to test (*e.g.,* `www.aljazeera.com`) to Google's URL (*e.g.,* `http://www.google.com.sa/www.aljazeera.com`). The normal behavior is to see the well-known `Page Not Found` HTTP `404` error code. If a different error code (*e.g.,* `403`) is returned, this means HTTP-URL-Keyword filtering is enabled.

4. TLS Filtering: Separately, the tool is extended to establish a TLS connection with the web server of the site we want to test to check if there is filtering on the HTTPS protocol.

My methodology and tool include the following steps and measurements in order to enhance and strengthen the results of my study.

First, I wanted to ensure that network issues (e.g., temporary unreachability, packet losses and other networking pathologies) do not affect the measured results. For this reason, I randomly selected and re-probed 10% of sites per category 100 times each and discovered no errors in the initial measurement for these websites. I also modified the set of open DNS servers used by Samizdat.

Second, in addition to the default DNS servers used by my vantage points (which belong to the respective ISP DNS service), I measured the Internet filtering on the following public servers: Google (8.8.8.8), Quad9 (9.9.9.9), OpenDNS (208.67.222.222), Norton (199.85.126.10), Comodo (8.26.56.26), and Level3 (209.244.0.3).

Third, to get more precise results, I performed DNS lookups using both UDP and TCP protocols. I tested site accessibility over both HTTP and HTTPS protocols since HTTPS is not amenable to keyword based filtering.

Finally, I also conducted a number of Wireshark measurements to capture and analyze the detailed network behaviors and to verify the subtleties of the filtering mechanisms.

7.3.4 Filtering at the Mobile App Level

In what is arguably, a relatively novel dimension in filtering, we want to assess if mobile applications are affected by filtering. For that, I conduct a systematic measurement study with two smartphones one in USA and one in Saudi Arabia and I compared the differences in terms of downloading and using apps. The analysis covers the period from

Figure 7.10: Geolocation of the vantage points

2013 to 2020. I discuss the detailed description and analysis of this study and the results in Section §7.4.5.

7.3.5 Measurement Vantage Points

I conducted measurements from six different vantage points distributed across four major cities in Saudi Arabia using the three major ISPs in the country. The four cities whose geographical location is shown in Figure 7.10) are: (1) Riyadh, which is the capital and the largest city of Saudi Arabia (population 6.5 million). Riyadh is centrally located; (2) Jeddah, which is located on the west coast and is the second largest city in the country (population 4 million); (3) Makkah, which is the birthplace of Islam and the spiritual center of the kingdom (population 2 million); and (4) Al-Khobar which is one of the major cities in the eastern region of the country (population 1 million). I selected

ID	City	ISP
N1	Riyadh	STC
N2	Jeddah	STC
N3	Al-Khobar	STC
N4	Makkah	STC
N5	Makkah	Zain
N6	Makkah	Mobily

Table 7.1: Measurement Vantage Points

these cities because of their different nature, roles they play in the kingdom, as well as for geographical distribution. Table 7.1 shows the details of each network. I chose vantage points connecting to different Internet Service Providers (ISPs) to understand whether there is ISP level filtering, or other variation in the experienced filtering based on the ISP. The machines in N1, N2, and N3 are connected to the Internet through the same ISP which is the Saudi Telecommunication Company (STC). I conducted these measurements in one city, Makkah, through 3 different vantage points connecting the same machine to all three major ISPs: STC, Zain, and Mobily. All machines are connected to their default gateways, or routers, with a 1GB Ethernet cable. All machines run Windows 10 Professional Edition.

I conducted the measurements multiple times to eliminate the effect of transient phenomena, such as short lived outages. In addition, to establish a baseline external reference point, the same measurements are also executed on my lab machine in the United States (US), which also runs Windows 10 Professional Edition and is connected to the university network. I used these measurements to better understand the results from my

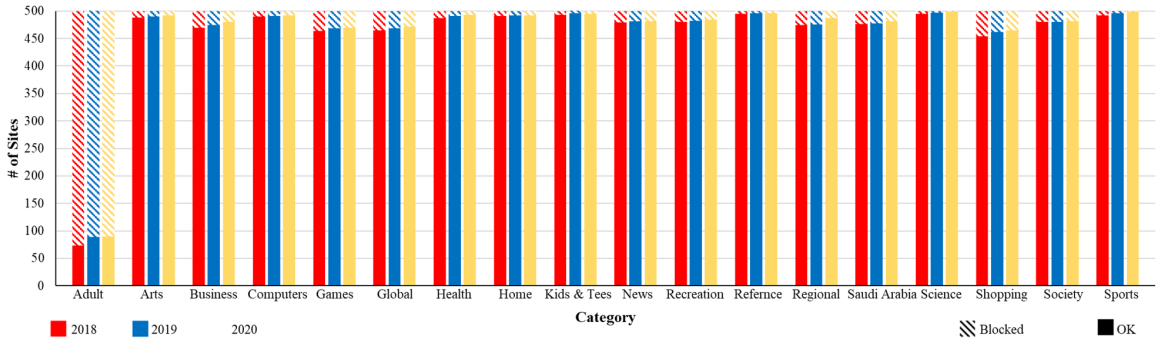


Figure 7.11: The scope of Internet filtering in Saudi Arabia

machines in Saudi Arabia. For example, to confirm that an unreachable website is indeed blocked inside Saudi Arabia, I checked if it is accessible from the US. If the website returned the same error code (*e.g.,* HTTP codes `503` or `301` indicating that the server is unavailable or moved permanently, respectively) in both countries, I consider that the lack of accessibility is not due to filtering.

7.4 Results

Overall, we observe that the filtering rules are significantly relaxed over the time for both websites and mobile apps; an evidence that Saudi Arabia is cautiously yet decisively opening its digital borders. I summarize the results of my study in Figure 7.11.

Adult. Unsurprisingly, we see that the most blocked category is *Adult* where 87.2%, 83.4%, and 84.4% of the websites are blocked in 2018, 2019, and 2020, respectively. The content of the sites in this list is usually related to pornography, gambling, drugs, violence, and similar content inappropriate for young audience. I spot-checked the content of some of the *Adult* sites that were not blocked and found that most are related to art work including

comics and caricatures (I did not find any that are pornographic, gambling, or drug-related for example).

Shopping. We find that the second most blocked category is *Shopping*. After deep inspection, we observe that most of the sites either belong to convenience stores or apparel retailers that do not have branches inside Saudi Arabia (*e.g.,* The Home Depot, CarMax, Bloomingdales, and Kohls) or sell products that are considered illegal (*e.g.,* alcohol and guns).

Games. The third most blocked category is *Games* in which all blocked sites related to gambling.

Global. In analyzing the *Global* category (which represents the most popular websites worldwide), we see that more than 7% of the these websites are blocked across the three-year measurements. We found that nearly 60% of these blocked sites also belong to the *Adult* category. The remaining 40% of the blocked sites belonged to social network applications such as WeChat and VK, as well as a few websites from China. Interestingly, in January 2019, students and faculty at the University of California (UC) have been warned not to use WeChat while visiting China; apparently, this warning is issued to protect their communications since that application raises security and privacy concerns [145].

A filtering conundrum: most visited and yet blocked. Surprisingly, we see that 24, 23, and 29 of the most visited websites from Saudi Arabia per Alexa are blocked in 2018, 2019, and 2020, respectively. One might wonder how a popular site in the country is visited, while it is blocked. Alexa determines the popularity of a site based on two metrics: (1) Unique Visitors which is determined by the number of unique visitors of a web page; and (2)

Pageviews which corresponds to the number of URL requests (*i.e.,* HTTP `GET request`) for a website [33]. My multi-layer analysis of filtering was able to resolve the mystery. The results show that all blocked sites from this category passed the DNS and IP filtering tests. This suggests that users received the requested DNS resolution, but were never able to view the web page, since it is blocked by other mechanisms as I discuss later in the section.

We also find that for a blocked website, the filtering system blocks the whole domain. For example, I tried to access the accounts of a number of blocked news sites (*e.g.,* `www.aljazeera.com`) on other social media applications (*e.g.,* Twitter and Instagram). Since these applications use HTTPS, and hence, HTTP-URL-keyword filtering is not applied, we find that they can be accessed and viewed from inside the country.

With respect to the operation of the filtering mechanism, we find that Internet filtering rules applied uniformly across the different vantage points and ISPs I tested: thus, I suspect that there is no additional ISP-level filtering. After verifying this observation, I show results only from one of the vantage points in the remainder of the paper (N6).

In regards to mobile apps measurements, my results show that 80% of blocked apps in 2018 were accessible in 2019. The experiment are repeated in 2020 and we find that all apps were accessible, except WeChat which is still debatable. This perhaps reflects that the kingdom is moving towards moderating regulations on Internet filtering.

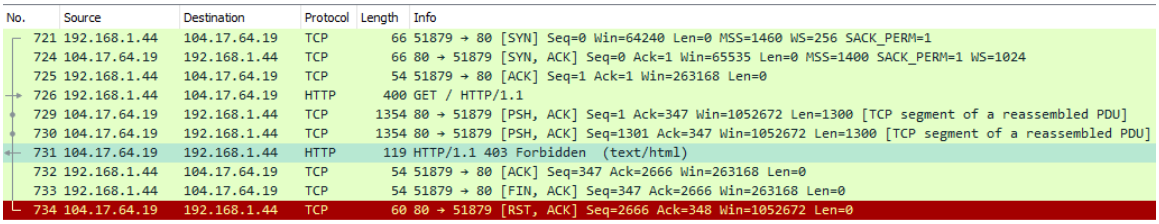
7.4.1 DNS Filtering

In my experiments, I did not encounter any evidence of DNS filtering. The numbers shown in the UDP and TCP columns in Table 7.2 related to transient connectivity issues

Table 7.2: Breakdown of Internet filtering results against Alexa top 500 websites in 18 categories. Numbers in blue, green, and red denote results in 2018, 2019, and 2020, respectively. The HTTP and TLS/HTTPS results are for status code `403`.

Category	DNS						IP			HTTP			TLS/HTTPS		
	UDP			TCP											
Adult	5	7	3	7	9	5	2	8	8	427	411	410	4	1	1
Arts	2	5	6	5	6	4	9	11	10	12	10	8	10	7	7
Business	3	10	11	3	8	5	6	13	14	30	26	20	38	29	15
Computers	0	0	0	0	0	0	1	0	0	10	9	8	8	6	3
Games	0	0	1	0	0	0	0	0	0	36	31	30	13	11	11
Global	6	7	7	6	7	7	12	12	12	35	31	28	9	8	6
Health	4	3	5	4	3	2	7	6	5	13	9	7	17	8	2
Home	1	2	0	1	2	1	4	5	4	9	8	8	16	10	7
Kids& Teens	2	1	1	2	1	1	3	1	0	7	6	5	12	6	5
News	0	0	0	0	0	1	1	0	0	21	19	19	20	17	13
Recreation	0	0	0	0	0	0	6	2	1	20	18	16	18	15	11
Reference	0	1	1	0	0	0	0	2	1	5	4	4	6	6	6
Regional	1	1	0	1	1	1	1	3	2	26	25	22	21	20	20
Saudi Arabia	12	12	12	11	12	13	18	18	18	24	23	19	15	15	14
Science	3	0	1	3	0	0	3	1	1	5	3	1	6	5	5
Shopping	1	1	0	1	1	1	1	1	1	46	38	35	48	36	33
Society	0	0	0	0	0	0	4	1	2	20	20	19	18	17	17
Sports	0	0	0	0	0	0	1	0	0	8	4	1	11	9	8

such as `TIMEOUT` and `SERVFAIL`. A very small portion of DNS lookups, consistently returned `REFUSED`, `NoAnswer`, or `NXDOMAIN` DNS error codes, but the number is negligible. To ensure that these websites are actually not blocked, I checked their status using my machines in



No.	Source	Destination	Protocol	Length	Info
721	192.168.1.44	104.17.64.19	TCP	66	51879 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
724	104.17.64.19	192.168.1.44	TCP	66	80 → 51879 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024
725	192.168.1.44	104.17.64.19	TCP	54	51879 → 80 [ACK] Seq=1 Ack=1 Win=263168 Len=0
726	192.168.1.44	104.17.64.19	HTTP	400	GET / HTTP/1.1
729	104.17.64.19	192.168.1.44	TCP	1354	80 → 51879 [PSH, ACK] Seq=1 Ack=347 Win=1052672 Len=1300 [TCP segment of a reassembled PDU]
730	104.17.64.19	192.168.1.44	TCP	1354	80 → 51879 [PSH, ACK] Seq=1301 Ack=347 Win=1052672 Len=1300 [TCP segment of a reassembled PDU]
731	104.17.64.19	192.168.1.44	HTTP	119	HTTP/1.1 403 Forbidden (text/html)
732	192.168.1.44	104.17.64.19	TCP	54	51879 → 80 [ACK] Seq=347 Ack=2666 Win=263168 Len=0
733	192.168.1.44	104.17.64.19	TCP	54	51879 → 80 [FIN, ACK] Seq=347 Ack=2666 Win=263168 Len=0
734	104.17.64.19	192.168.1.44	TCP	60	80 → 51879 [RST, ACK] Seq=2666 Ack=348 Win=1052672 Len=0

Figure 7.12: Wireshark trace of HTTP-URL-Keyward filtering

the US and verified that these websites return the same errors indicating that they have likely gone offline.

We further find that there was no difference between DNS lookups performed using UDP and TCP. This indicates there are no constraints in DNS over TCP deployment in the country. To be more certain, I repeated the same DNS lookups on 6 open DNS servers. We find that the final results confirm the findings. There were minor variations between the behavior of the open resolvers; for example Comodo had the lowest success rate in DNS lookups (especially in UDP) in all categories. As a result, we conclude that DNS filtering is unlikely to be in use.

7.4.2 IP Address Filtering

The measurements show a number of websites in which the IP address filtering failed as shown in Table 7.2. The data suggests that the main cause of this issue is not the Internet filtering system, but DNS lookup failures since no IP address is retrieved. For all other websites, the tool was able to connect to their IPs on port 80. This indicates there is no filtering at the level of TCP (or UDP) connection establishment.

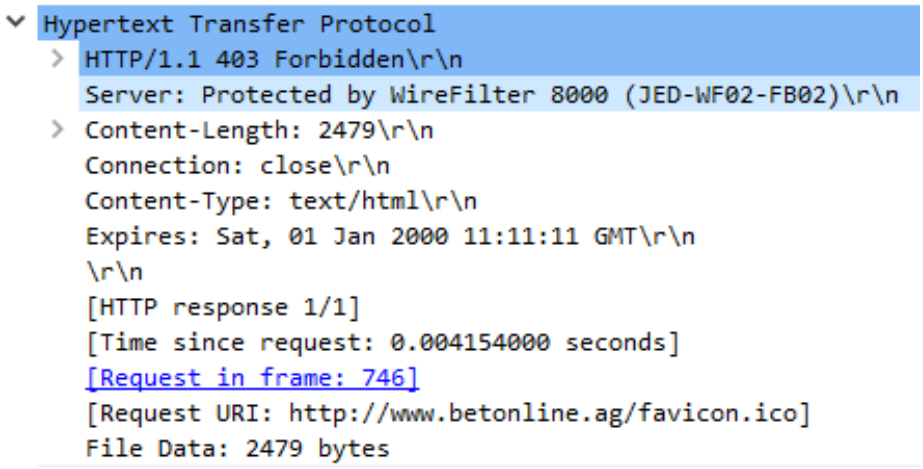


Figure 7.13: Wireshark trace showing the company in charge of filtering in Saudi Arabia: WireFilter

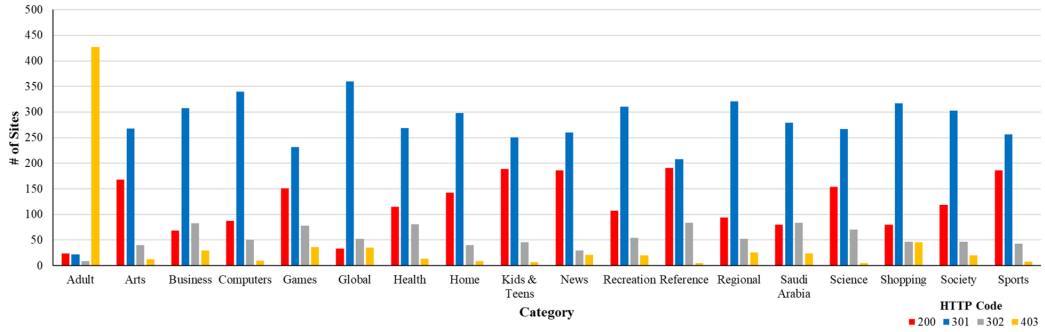
7.4.3 HTTP Filtering

As shown in Table 7.2, a large number of websites were filtered based on the HTTP URL string, (either FQDNs or special keywords): 82.2%, 7.6%, and 6.2% of the *Adult*, *Shopping*, and *Games* websites were blocked in 2019, respectively. Looking at the detailed logs, the results show that the TCP 3-way handshake process between one of my Saudi machines and the forbidden site's web server establishes successfully. I consider FQDN and Keyword based filtering separately.

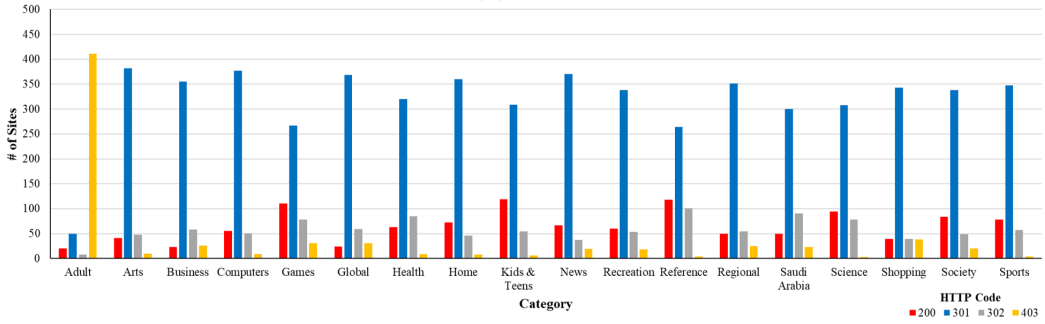
FQDN-Based Filtering. I send the `GET` request directly to the FQDN I want to test. The filtering system allows the client to send the `GET` request; however, instead of receiving a legitimate HTTP response, the system replies back with an HTTP response with the status code `403` for accessing forbidden content. This observation contradicts the findings by Verkamp et al. [267] where the authors report that a spoofed HTTP response with a status code `200` is returned, perhaps indicating that the filtering implementation has

changed. The filtering mechanism then directs the user to one of the warning pages shown in Figure 7.3 and Figure 7.4, based on the site's category. The warning page is an HTML `<iframe>` presenting a warning message both in Arabic and English. Figure 7.12 shows a Wireshark trace for blocked website `www.betonline.ag` (a gambling website). The trace shows that after receiving the `403` HTTP error code, the client receives `TCP-RST` packets forcing a termination of the HTTP connection. The error message displayed when a blocked website is accessed explicitly indicates that this filtering is maintained by a company called *Wire Filter* [274] (see the HTML `<iframe>` in Figure 7.13).

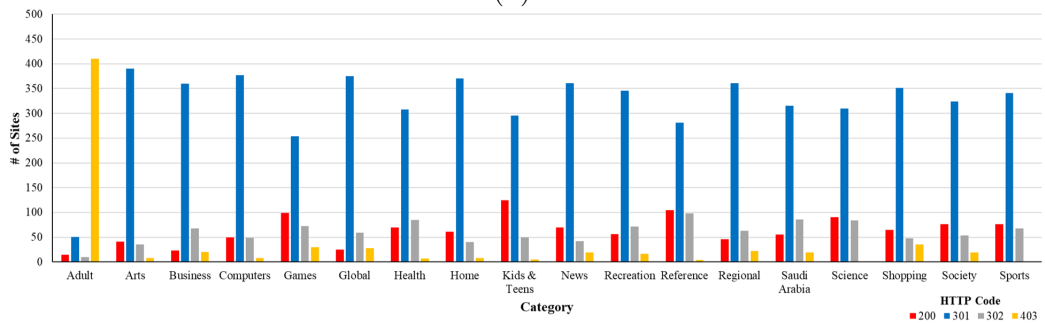
URL-Keyword Filtering. Recall that in this experiment we access an unblocked system, but have the url include a blocked site as part of the URL (e.g., `http://a.com/b.com/`, where `a.com` is unblocked and `b.com` is blocked). The results show identical behavior of filtering as in the case of FQDN-based HTTP filtering, confirming that the filtering uses URL-keyword filtering. The difference in these two cases occurs when using HTTPS. When I repeat the keyword-filtering experiments using HTTPS, the filtering does not work and the client receives the `404 Page Not Found` code indicating that URL-keyword filtering works only at the HTTP-level connection. In this case, the connection (including the TLS handshake) is being established to the unblocked website (`a.com`), which returns that the specific page (the one I created with the blocked domain) is not found – no filtering occurred. A breakdown of HTTP filtering results is shown in Figure 7.14.



(a) 2018



(b) 2019



(c) 2020

Figure 7.14: HTTP filtering results by returned status code. The HTTP `200 OK` success status response code indicates that the request has succeeded. The `301` and `302 Found` status codes are used to indicate that the URL has been permanently and temporally, respectively, moved/redirected to a new URL. Code `403` indicates that access to the requested URL is forbidden due to client-related issues; in my case the reason is filtering.

No.	Source	Destination	Protocol	Length	Info
275	192.168.1.44	104.17.64.19	TCP	66	51889 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
279	104.17.64.19	192.168.1.44	TCP	66	443 → 51889 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024
280	192.168.1.44	104.17.64.19	TCP	54	51889 → 443 [ACK] Seq=1 Ack=1 Win=263168 Len=0
281	192.168.1.44	104.17.64.19	TLSv1	571	Client Hello
282	104.17.64.19	192.168.1.44	TCP	60	443 → 51889 [RST, ACK] Seq=1 Ack=2 Win=1052672 Len=0

Figure 7.15: Wireshark trace of TLS filtering

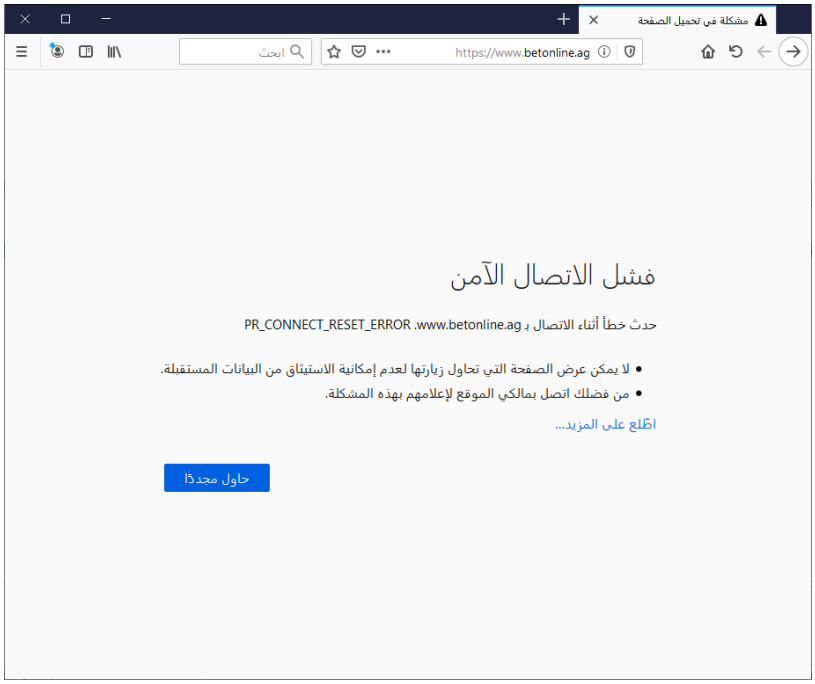


Figure 7.16: TLS/HTTPS connection cannot be established for a blocked site

7.4.4 TLS Filtering

HTTPS disables keyword based filtering since it is not possible for the filtering system to access the encrypted contents of the TCP packet holding the `GET` request. Therefore, I tested the accessibility of the websites with HTTPS. We discovered that if the website being contacted is blocked, it remains blocked under HTTPS. When examining the traces, we discovered that this is due to TLS level filtering. As shown in Figure 7.15 when trying to access the blocked website `www.betonline.ag`, the filtering system allows the TCP 3-way

handshake but sends a `TCP-RST` packet when the client tries to establish a TLS connection. On Windows, when I sent the `GET` request, the Windows socket error code `10054` was returned, indicating that the HTTPS connection was forcibly closed by the server. In this case, the browser displays a page (as shown in Figure 7.16) indicating that the HTTPS/TLS connection could not be established.

7.4.5 Mobile Application Filtering

In this section, I report the results in regards to mobile application filtering, and I start by providing some context regarding the policies of Saudi Arabia.

Historical context regarding mobile app usage. In 2013, CITC blocked the Voice over Internet Protocol (VoIP) call services on Viber, a popular mobile application that offers free video/voice calls [262]. VoIP calls on similar applications were slowly being blocked including FaceTime, Skype, WhatsApp and Snapchat. I believe the main reason behind blocking is economic, since these applications provide free alternatives to services that otherwise generate revenue to the ISPs. CITC received requests from service providers such as Mobily and STC to block the free or low-cost VoIP calls on these applications to protect their competitiveness and rights [137, 144]. However, in 2017, CITC responded to citizens demands and announced its intent to lift the ban on all applications that provide voice and video communications over the Internet that meet the regulatory requirements of the country [94]. We conjecture that this decision was also driven by the Vision 2030 and National Transformation 2020 programs published with the aim of modernizing society: one of the stated goals is to provide transparency and clarity with respect to policies especially in the telecommunications and technology technology sectors.

Table 7.3: Breakdown of Internet filtering results against popular messaging mobile applications. I tested the text, audio and video communication services. Symbols show if a communication service is supported (✓), blocked (✗), not applicable (NA) (e.g. service not available at the time), or not tested (NT). Note that the results displayed for the period 2013-2017 are based on personal experience and not extensive measurements. Also note that the release date of all apps except HouseParty (released in 2019) is either before or within this period. For instance, Line, Telegram, and Google Duo were initially released in 2011, 2013, and 2016, respectively.

Application	2013-2017			2018			2019			2020		
	Text	Audio	Video	Text	Audio	Video	Text	Audio	Video	Text	Audio	Video
Viber	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
Tango	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
FaceTime	✓(iMessage)	✗	✗	✓(iMessage)	✓	✓	✓(iMessage)	✓	✓	✓(iMessage)	✓	✓
YeeCall	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Skype	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
WhatsApp	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓	✓
Line	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
Telegram	✗	✗	NA	✗	✗	NA	✓	✓	NA	✓	✓	NA
AllApp	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Google Duo	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Houseparty	NA	NA	NA	NA	NA	NA	✓	✓	✓	✓	✓	✓
WeChat	NT	NT	NT	NT	NT	NT	✗	✗	✗	✗	✗	✗
SOMA	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Snapchat	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Google Hongout	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Facebook Messenger	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
imo	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
JusTalk	NT	NT	NT	✓	✓	✓	✓	✓	✓	✓	✓	✓

Measurements. I conduct three measurements for mobile apps whose results I summarize in Table 7.3. I consider 16 communication mobile apps including FaceTime, Tango, Line,

Viber, SOMA, YeeCall, Facebook Messenger, WhatsApp, Snapchat, and imo. I attempt to install and use them on two iPhones, one in Saudi Arabia and the other in USA. I tested the text, audio, and video communication services. All of these apps support text, audio and video communication.

In March 2018, five applications failed to establish at least one of the text, audio, and video communication services, as shown in Table 7.3. WhatsApp and Viber established an active connection for 1-2 seconds, but then the calls got disconnected suddenly. I believe that this experiment indicates that these two applications were indeed blocked in Saudi Arabia [43]. We also find that VoIP calls on imo are completely blocked in Saudi Arabia.

In October 2019, I repeated the experiment and added two more applications: Houseparty and WeChat. We find that both audio and video calls are blocked on WhatsApp, which was corroborated by a CITC statement [23].

The application WeChat exhibits a uniquely interesting behavior. WeChat is one of the most popular messaging applications owned by the Chinese company Tencent [209, 1, 145]. In Saudi Arabia, the installation of the application comes with a pre-condition: the user has to show s/he has a friend on WeChat, who needs to meet additional requirements as shown in Figure. 7.17! These requirements for the friend are that s/he: (a) has been a WeChat user for at least one month if s/he an international user or for 6 months if s/he is a China Mainland user; (b) has not completed "Help Frined Register" check for other new user in the past month; (c) has not been blocked from using WeChat in the past month; and (d) if s/he is a China Mainland user, s/he has activated WeChat Pay. By contrast, WeChat can be installed on an iPhone user in the US without any such requirements. Intrigued, I

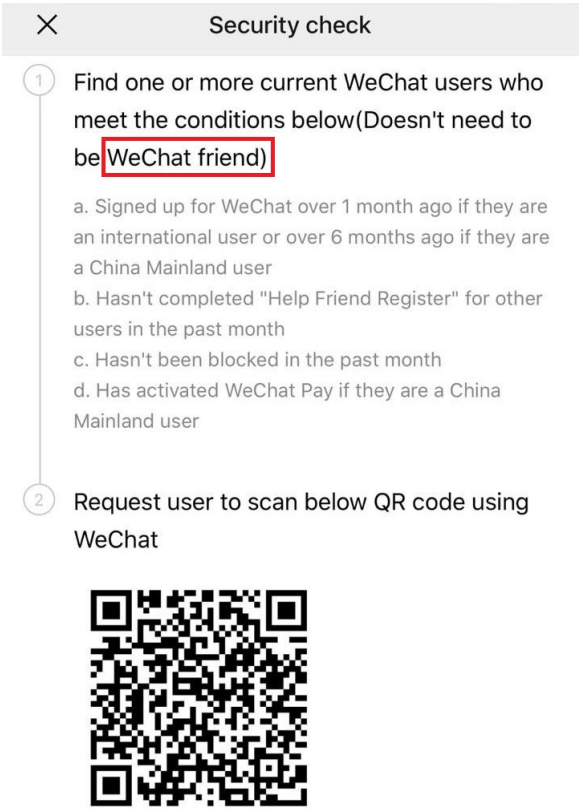


Figure 7.17: WeChat Security Check

repeated the experiment with three more iPhones in Saudi Arabia. The installation failed on all of them for the same reason. Although the vice president of Tencent announced back in 2013 that WeChat is available in Saudi Arabia [10], this is not fully accurate. Currently, I am not sure if the installation failure is caused by Tencent or the Internet filtering system.

All other tested messaging applications are open and supported including Viber and imo.

Finally, in April 2020, I repeated the experiment and found that nearly all previously-blocked messaging applications were accessible, including WhatsApp, with the only exception being WeChat.

7.4.6 Relation Between Geopolitical Events and Internet Filtering

Over the past years, the Middle East experienced several major political events that affected societies internally and externally. Consequently, these events affected Saudi Arabian policies regarding access to information, and my measurements capture the effect of such events, as I discuss below.

A prominent event is the rise of the so-called "Islamic State" in Iraq and Syria (known as ISIS) in the last decade, whose effect emerges in my measurements. I tested the accessibility of a number of ISIS-friendly websites (which I obtained by scouring the web and following previous practice I do not disclose for ethical reasons) and found that all of them were blocked. ISIS and its affiliates have exploited social media websites, such as Twitter, to spread their propaganda and to recruit new members [227]. This type of activity has in turn been countered by efforts from the Saudi Arabian government by regulating information access for Saudi citizens. For instance, at the Shura Council, the chairman of the Islamic and judicial affairs committee called for the blocking of all ISIS websites, since they considered them to advocate terrorism and destabilization to the region [31], a stance that has been followed by many other countries and institutions as well [128, 238, 120, 225, 204]. In addition, many Saudi citizens launched an online campaign on Twitter aiming to lock down user accounts belonging to or supporting ISIS [153].

Another prominent event is the increased political tension between Qatar and Saudi Arabia, which was also captured in my measurements. Because of these tensions, the Saudi authorities blocked some Qatari news web sites [16]. For instance, as shown in Figure 7.4, a warning page by the Ministry of Culture and Information was displayed when

I tried to visit `www.aljazeera.com`; one of the most popular news websites in Qatar. In addition, in April 2020, I obtained a list of Qatari news sites [16] and found that all of them were blocked.

Another notable event is the ongoing conflict between Saudi Arabia and Iran. Following an attack on the Saudi embassy in Tehran in January 2016 [14], Saudi Arabia cut all diplomatic relations with Iran. Our measurements show evidence that this event had impact on the Internet filtering. For instance, in 2018, our measurements show that some Iranian sites (mostly from the *News* category) got blocked.

Finally, in April 2020, we observed a change in the access for some Turkish sites compared to the earlier measurements. Upon investigation, we find that Saudi authorities blocked two prominent Turkish news websites, Anadolu and TRT Arabic platforms, amid what the Ministry of Media communicated as continued violations of their regulations. I conjecture that the move was partly driven by a campaign on Twitter by Saudi citizens calling for the Turkish news platforms to be blocked [260].

7.5 Internet Filtering Infrastructure

In the past, all network traffic in Saudi Arabia was directed to a central network at the Internet Service Unit (ISU). In 2016, the volume of international Internet traffic rose by 114% relative to the traffic in 2015 (to 3.185 TB/s [87]). With the increasing market penetration of Internet connectivity and the accompanying growth in Internet traffic, the Internet filtering infrastructure has also evolved. All outbound Internet connectivity is routed through two main Data Service Providers (DSPs): (1) the Integrated Telecom

```
C:\Users\AHC>tracert 94.237.49.240

Tracing route to 94-237-49-240.uk-lon1.host.upcloud.com [94.237.49.240]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.1.1
  2    28 ms    12 ms    14 ms  User's public IP address
  3    12 ms    12 ms    19 ms  [redacted]
  4    12 ms    12 ms    11 ms  [redacted]
  5    11 ms    11 ms    12 ms  10.188.199.32
  6    61 ms    60 ms    61 ms  [redacted]
  7    84 ms    82 ms    80 ms  [redacted]
  8    87 ms    85 ms    91 ms  [redacted]
  9    94 ms    89 ms    90 ms  94-237-49-240.uk-lon1.host.upcloud.com [94.237.49.240]

Trace complete.
```

Figure 7.18: Output of `tracert` between a machine in Saudi Arabia and a machine in UK

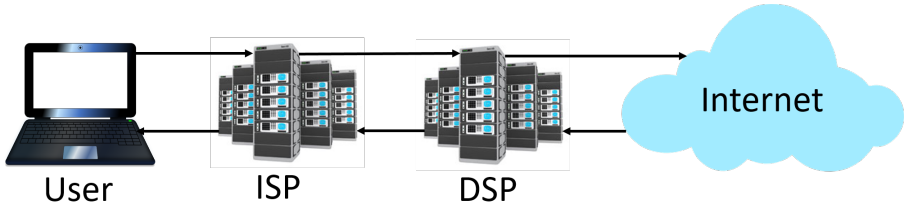


Figure 7.19: The infrastructure of the filtering system in Saudi Arabia

Company (by Mobily) and (2) Bayanat al-Oula for Network Services (by Mawarid Holding Group), to provide national and international Internet communications services [19].

To explore the filtering infrastructure, I used the `tracert` utility to identify the hops through the communication between machines inside and outside the country. I conducted my measurements by randomly selecting 10 public IP addresses of machines outside Saudi Arabia which were verified using the tool [172]. I also confirmed that these machines are accessible and responding to `tracert` from my US vantage point.

When an end-user in Saudi Arabia sends a request to an international website, the request first goes through her ISP's proxy before it reaches the DSPs' proxy servers.

Then, I observed that before connecting to hops outside the country, the packets has to go through a node with a private address (`10.188.x.x`) as shown in Figure 7.18, which I assume is the IP address of the system used for Internet filtering. CITC clearly states that the Internet filtering servers are located in a centralized access point at KACST, which is located in Riyadh [91, 90]. The filtering infrastructure is illustrated with in Figure 7.19.

To further validate that `10.188.x.x` is the filtering center, I conducted a two-directional measurement as follows. I conducted the same measurement to `tracert` from one of my machines in Saudi Arabia, say A, to my lab machine in the US, say B. I then compared the path of the routers visited $A \rightarrow B$ and $B \rightarrow A$ using `tracert`. The two measured paths were similar in terms of router prefixes with the exception of the filtering node (`10.188.x.x`), which was not present on the traceroute originating from USA host to Saudi Arabia.

As a final step to increase my confidence, I repeated the above measurement from 10 hosts within Saudi Arabia and from two cities: Jeddah and Makkah. The `tracert` outputs reveal a large amount of nodes that share the same address space as the filtering node, confirming that all network traffic within the country passes through centralized servers as well.

7.6 Concluding Remarks

The Internet has such a disruptive effect on many societies, which often forces institutions and governments to regulate the access to content and services for reasons both political and cultural. While the debate around regulating the digital and Internet

access is going on, it is important to develop methods and conduct studies for tracking and understanding the Internet filtering behavior.

In this paper, I study Internet filtering in the Kingdom of Saudi Arabia: a traditionally conservative country, which seems to have made significant steps towards modernization in the last five years. This move towards openness is amply supported by my work. My contributions are twofold. First, I develop a comprehensive methodology and tools to measure, collect and analyze Internet filtering behavior at a refined level of granularity. Second, I present a comprehensive longitudinal study of Internet filtering in Saudi Arabia over the period of three years. Specifically, I conduct measurements to evaluate filtering behavior by probing Alexa's top 500 websites in 18 different categories from viewpoints covering the three largest telecommunications companies in Saudi Arabia and five cities. As expected, we find that the filtering is most common for sites in the *Adult* category of Alexa. I also conduct measurements to test mobile application accessibility by examining the status of 18 of the most popular mobile social network applications worldwide and in the middle east such as WhatsApp, Facetime, and Skype, and find that WeChat is still not freely available in Saudi Arabia.

The three year span of my study enables us to study changes in the filtering policy and view these changes in the wider geopolitical context experienced by the kingdom and the region. For example, we find evidence that the emphasis on modernization is leading to relaxing regulations on filtering (67% and 93% of the blocked mobile apps over the period 2013-2017 were accessible in 2018 and 2019, respectively, and all tested apps were accessible in 2020, except WeChat which is still debatable). We find that ISIS-friendly sites

are blocked. Interestingly, we found that news sites from the countries of Qatar, Iran, and Turkey got blocked in 2017, 2018 and 2020, respectively, amid rising diplomatic tensions between the kingdom and these countries.

I see this work as a solid step towards developing a deep understanding of the various techniques that are used to support filtering.

Chapter 8

Future Work and Concluding Remarks

To conclude, in this dissertation I consider a common scenario where IoTs (or generally, edge computing devices) are being connected to the Internet and participate in Internet scale protocols such as DNS and TLS, HTTPS, and others. Traditional defenses for these protocols rely on cryptographic algorithms which require computationally expensive operations that can be far too complex for low-power and inexpensive devices (*e.g.,* IoTs). Typically, in such a scenario, one of three options will occur: (1) Retain security and extend it to the edge/IoT devices — in this case performance and energy will suffer; (2) Sacrifice security entirely from all interactions, therefore the last link is left insecure; or (3) Rewrite the entire system or even develop new security protocols specifically to incorporate these lightweight devices. In real-world implementations, developers and system administrators choose one or two out of three in order to provide efficient configurations for their clients.

In the light of this context, my research pursues three main directions in addition to a side direction to recommend further research. In my first direction, I emphasize on the importance of retaining security by demonstrating an attack on the DNS infrastructure. In the second direction, I devise and study the feasibility to integrate all three: security, performance, and development without sacrificing one to save the other. The contributions in this direction retains end-to-end security and more importantly can protect the last link to reach the edge device. In the third direction, I demonstrate that the first option is not desirable in resource-constrained environments. Basically, I analyze and measure the cryptographic overhead of conventional cryptographic primitives that has been used extensively in various applications. The performance results which are related to latency and energy consumption confirm with the measurement results in the third research direction. Lastly, I present a slightly different research direction to shed the light on an interesting topic into the conversation: I present a systematic longitudinal study on Internet filtering in the Kingdom of Saudi Arabia, a country that has embarked on economic and societal changes in many aspects of its daily operations and public policy decisions, with the stated objectives of modernization towards building a more tolerant Islamic country.

For future work, I hope that the lessons learned from these directions help me to build (or at least critically understand) the best framework for IoT and edge computing devices. More precisely, I need to understand the required security primitives that overcome all the three aforementioned challenges. With CSProp, I worked on optimizing public key operations for resource-constrained devices. In the near future, I will work to see how I can optimize private key operations as well.

Bibliography

[1] China's wechat, weibo and baidu under investigation. *BBC News*. , available at https://www.bbc.com/news/world-asia-china-40896235, year = 2017.

[2] Mac OS X/Darwin man pages: setrlimit (2). http://www.manpages.info/macosx/setrlimit.2.html. Online; accessed 8 May 2018.

[3] Mills() Function. https://www.arduino.cc/reference/en/language/functions/time/millis/.

[4] Safer Internet Day. https://www.saferinternetday.org/.

[5] Saudi orders telcos to ensure skype, whatsapp meet local laws. *Reuters Newsletter*. , available at https://www.reuters.com/article/saudi-telecoms-ban/saudi-orders-telcos-to-ensure-skype-whatsapp-meet-local-laws\-idUSL5N0CN0DH20130331, year=2013.

[6] setrlimit(2) - Linux man page. https://linux.die.net/man/2/setrlimit. Online; accessed 8 May 2018.

[7] TLD Zone File Statistics. https://www.statdns.com/. Online; accessed 24 July 2018.

[8] Use of DNSSEC-ECDSA Validation for World (XA). https://stats.labs.apnic.net/ecdsa/XA. Online; accessed 25 July 2018.

[9] Multiple DNS implementations vulnerable to cache poisoning. http://www.kb.cert.org/vuls/id/800113, 2012.

[10] Wechat arrives in saudi arabia. *Saudi Gazette* (2013). , available at http://saudigazette.com.sa/article/47305.

[11] Towards a comprehensive picture of the great firewall's DNS censorship. In *4th USENIX Workshop on Free and Open Communications on the Internet (FOCI 14)* (San Diego, CA, Aug. 2014), USENIX Association. , available at https://www.usenix.org/conference/foci14/workshop-program/presentation/anonymous.

[12] The 15 enemies of the internet and other countries to watch, 2016. , available at https://rsf.org/en/news/15-enemies-internet-and-other-countries-watch.

[13] Dns amplification attacks. *US-CERT* (2016).

[14] Hun condemns attack on saudi embassy in iran. *BBC News* (2016). , available at https://www.bbc.com/news/world-middle-east-35229385.

[15] dnsmasq. https://wiki.archlinux.org/index.php/dnsmasq, 2017.

[16] Here is a list of all qatari web sites that are blocked in saudi arabia. *Alarabiya* (2017). , available at http://ara.tv/z8sek.

[17] Top Sites in United States. https://www.alexa.com/topsites/countries/US, 2017.

[18] The ban on saudi women driving is ending: Here's what you need to know, 2018. , available at https://www.cnn.com/2018/06/22/middleeast/saudi-women-driving-ban-end-intl/index.html.

[19] Saudi arabia country report - freedom on the net 2018. *Freedom House* (2018). , available at https://freedomhouse.org/report/freedom-net/2018/saudi-arabia.

[20] The 2019 world press freedom index, 2019. , available at https://rsf.org/en/saudi-arabia.

[21] Freedom in the world 2019: Saudi arabia. *Freedom House* (2019). , available at https://freedomhouse.org/report/freedom-world/2019/saudi-arabia.

[22] Lightweight cryptography, 2019. Available at https://csrc.nist.gov/Projects/Lightweight-Cryptography.

[23] Whatsapp calls are blocked in saudi arabia .. the reasons are organizational. *Sabq* (2019). , available at https://sabq.org/wY2Vmk.

[24] Wyze cam v2 smart home camera, Visited on 2020-03-28. Available at https://www.amazon.com/Wyze-Indoor-Wireless-Detection-Assistant/dp/B076H3SRXG/ref=sr_1_3?dchild=1&keywords=wyze+cam+v2&qid=1585439194&s=electronics&sr=1-3.

[25] Arduino mkr wifi 1010, Visited on 2020-04-01. Available at https://store.arduino.cc/usa/mkr-wifi-1010.

[26] Arduino uno rev3, Visited on 2020-04-01. Available at https://store.arduino.cc/usa/arduino-uno-rev3.

[27] Avr crypto library, Visited on 2020-04-02. Available at http://www.emsign.nl/.

[28] Lightblue bean, Visited on 2020-04-02. Available at https://www.adafruit.com/product/2732.

[29] Aceto, G., Botta, A., Pescapè, A., Feamster, N., Awan, M. F., Ahmad, T., and Qaisar, S. Monitoring internet censorship with ubica. In *International Workshop on Traffic Monitoring and Analysis* (2015), Springer, pp. 143–157.

[30] Al-Tawil, K. M. The internet in saudi arabia. *Telecommunications Policy 25*, 8-9 (2001), 625–632.

[31] Alarabiya. Saudi arabia .. al-shura demands to ban all isis web sites. *Alarabiya* (2016). , available at `http://ara.tv/gqz88`.

[32] Albrecht, M., Gentry, C., Halevi, S., and Katz, J. Attacking cryptographic schemes based on perturbation polynomials. In *Proceedings of the 16th ACM conference on Computer and communications security* (2009), ACM, pp. 1–10.

[33] Alexa. How are Alexa's traffic rankings determined? available at `https://support.alexa.com/hc/en-us/articles/200449744-How-are-Alexa-s-traffic-rankings-determined-`, note = Online; accessed 27 April 2020.

[34] Alexa Top Sites. The alexa top sites web service. , available at `https://aws.amazon.com/alexa-top-sites/`.

[35] Alexa Top Sites. The top 500 sites on the web (2018). , available at `https://www.alexa.com/topsites`, note = Online; accessed 27 April 2020.

[36] Alharbi, F., Chang, J., Zhou, Y., Qian, F., Qian, Z., and Abu-Ghazaleh, N. Collaborative client-side dns cache poisoning attack. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications* (2019), IEEE, pp. 1153–1161.

[37] Alkadhi, Meshal. Al-Burhan. `http://www.el-burhan.com/blog/`.

[38] Alrawais, A., Alhothaily, A., Cheng, X., Hu, C., and Yu, J. Secureguard: A certificate validation system in public key infrastructure. *IEEE Transactions on Vehicular Technology 67*, 6 (2018), 5399–5408.

[39] Alrawi, O., Lever, C., Antonakakis, M., and Monrose, F. Sok: Security evaluation of home-based iot deployments. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P). https://doi. org/10.1109/SP* (2019).

[40] Altman, D., Machin, D., Bryant, T., and Gardner, M. *Statistics with confidence: confidence intervals and statistical guidelines*. John Wiley & Sons, 2013.

[41] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., et al. Understanding the mirai botnet. In *26th {USENIX} Security Symposium ({USENIX} Security 17)* (2017), pp. 1093–1110.

[42] Antonov, P., and Antonova, V. Development of the attack against rsa with low public exponent and related messages. In *Proceedings of the 2007 international conference on Computer systems and technologies* (2007), ACM, p. 50.

[43] Arab News. Saudi Communications Commission activates Internet calls, WhatsApp still blocked. https://www.arabnews.com/node/1164956/saudi-arabia, 2017.

[44] Aranha, D. F., and Gouvêa, C. P. L. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[45] Arduino. Arduino software, Visited on 2020-04-01. Available at https://www.arduino.cc/en/main/software.

[46] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. RFC 4035 - Threat Analysis of the Domain Name System (DNS), 2005.

[47] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. Rfc4033:dns security introduction and requirements. Tech. rep., 2005.

[48] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. Rfc4034:resource records for the dns security extensions. Tech. rep., 2005.

[49] Arkko, J., Rissanen, H., Keranen, A., and Sethi, M. Practical considerations and implementation experiences in securing smart object networks.

[50] Arris Router. Watts up pro portable power meter, Visited on 2020-03-28. Available at https://www.amazon.com/NVG468MQ-802-11ac-MoCA%C2%AE2-0-Frontier-Wireless-AC/dp/B073F17BSG/ref=sr_1_1?dchild=1&keywords=Arris+NVG468MQ&qid=1585441528&sr=8-1.

[51] Aryan, S., Aryan, H., and Halderman, J. A. Internet censorship in iran: A first look. In *FOCI* (2013).

[52] Ateniese, G., and Hohenberger, S. Proxy re-signatures: new definitions, algorithms, and applications. In *Proceedings of the 12th ACM conference on Computer and communications security* (2005), ACM, pp. 310–319.

[53] Atkins, D., and Austein, R. RFC 3833 - Threat Analysis of the Domain Name System (DNS). https://tools.ietf.org/html/rfc3833, 2004.

[54] Aumasson, J.-P., Henzen, L., Meier, W., and Naya-Plasencia, M. Quark: A lightweight hash. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2010), Springer, pp. 1–15.

[55] Banerjee, U., Juvekar, C., Wright, A., Chandrakasan, A. P., et al. An energy-efficient reconfigurable dtls cryptographic engine for end-to-end security in iot applications. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)* (2018), IEEE, pp. 42–44.

[56] Bauer, S. Attacking exponent blinding in rsa without crt. In *International Workshop on Constructive Side-Channel Analysis and Secure Design* (2012), Springer, pp. 82–88.

[57] Bellare, M., and Rogaway, P. Optimal asymmetric encryption padding–how to encrypt with rsa. In *Advances in Cryptology–EUROCRYPT'94*, pp. 92–111.

[58] Bellare, M., and Rogaway, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security* (1993), ACM, pp. 62–73.

[59] Bellare, M., and Rogaway, P. The exact security of digital signatures-how to sign with rsa and rabin. In *International Conference on the Theory and Applications of Cryptographic Techniques* (1996), Springer, pp. 399–416.

[60] Bellovin, S. M. Security problems in the tcp/ip protocol suite. *ACM SIGCOMM CCR 19*, 2 (1989), 32–48.

[61] Bernstein, D. J. "dns forgery". `https://cr.yp.to/djbdns/forgery.html`, 2002. Internet publication.

[62] Bernstein, D. J. "the dns random library interface". `https://cr.yp.to/djbdns/dns.html`, 2008. Internet publication.

[63] Beverly, R., Berger, A., Hyun, Y., et al. Understanding the efficacy of deployed internet source address validation filtering. In *ACM IMC* (2009), pp. 356–369.

[64] Beverly, R., Koga, R., and Claffy, K. Initial longitudinal analysis of ip source spoofing capability on the internet.

[65] Biryukov, A., and Perrin, L. P. State of the art in lightweight symmetric cryptography.

[66] Blaze, M., Bleumer, G., and Strauss, M. Divertible protocols and atomic proxy cryptography. In *International Conference on the Theory and Applications of Cryptographic Techniques* (1998), Springer, pp. 127–144.

[67] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. Present: An ultra-lightweight block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2007), Springer, pp. 450–466.

[68] Boneh, D. Twenty years of attacks on the rsa cryptosystem. *Notices of the AMS 46*, 2 (1999), 203–213.

[69] Boneh, D., Durfee, G., and Frankel, Y. An attack on RSA given a small fraction of the private key bits. In *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings* (1998), pp. 25–34.

[70] Boneh, D., Durfee, G., and Frankel, Y. An attack on rsa given a small fraction of the private key bits. In *International Conference on the Theory and Application of Cryptology and Information Security* (1998), Springer, pp. 25–34.

[71] Boneh, D., and Gueron, S. Surnaming schemes, fast verification, and applications to sgx technology. In *Cryptographers' Track at the RSA Conference* (2017), Springer, pp. 149–164.

[72] Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., and Wustrow, E. Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 157–175.

[73] Bouhenguel, R., Mahgoub, I., and Ilyas, M. Bluetooth security in wearable computing applications. In *2008 international symposium on high capacity optical networks and enabling technologies* (2008), IEEE, pp. 182–186.

[74] Braeken, A., Liyanage, M., and Jurcut, A. D. Anonymous lightweight proxy based key agreement for iot (alpka). *Wireless Personal Communications 106*, 2 (2019), 345–364.

[75] Braun, M., Hess, E., and Meyer, B. Using elliptic curves on rfid tags. *International Journal of Computer Science and Network Security 2* (2008), 1–9.

[76] Brown, M. The linux traffic control HOWTO, 2006. Accessed May 2018 from `http://tldp.org/HOWTO/Traffic-Control-HOWTO/index.html`.

[77] Buchanan, W. J., Li, S., and Asif, R. Lightweight cryptography methods. *Journal of Cyber Security Technology 1*, 3-4 (2017), 187–201.

[78] Censored Planet. Censored Planet. `https://censoredplanet.org/about`. Online; accessed 29 April 2020.

[79] Chaabane, A., Chen, T., Cunche, M., De Cristofaro, E., Friedman, A., and Kaafar, M. A. Censorship in the wild: Analyzing internet filtering in syria. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), ACM, pp. 285–298.

[80] Chauhan, A., Sidhu, I., and Pandey, A. Cryptographic-protocols-arduino-and-pc library, Visited on 2020-04-02. Available at `https://github.com/arpitchauhan/cryptographic-protocols-arduino-and-PC`.

[81] Chen, J., Diao, W., Zhao, Q., Zuo, C., Lin, Z., Wang, X., Lau, W. C., Sun, M., Yang, R., and Zhang, K. Iotfuzzer: Discovering memory corruptions in iot through app-based fuzzing. In *NDSS* (2018).

[82] Clayton, R. Failures in a hybrid content blocking system. In *International Workshop on Privacy Enhancing Technologies* (2005), Springer, pp. 78–92.

[83] Clayton, R., Murdoch, S. J., and Watson, R. N. Ignoring the great firewall of china. In *International Workshop on Privacy Enhancing Technologies* (2006), Springer, pp. 20–35.

[84] Cloudflare. Ecdsa: The missing piece of dnssec, Visited on 2020-01-09. , available at `https://www.cloudflare.com/dns/dnssec/ecdsa-and-dnssec/`.

[85] Comcast Corporation. Before the federal communications commission in the matter of broadband industry practices. http://fjallfoss.fcc.gov/ecfs/document/view?id=6519840991, 2008. Online; accessed 16 May 2018.

[86] Communications and Information Technology Commission (CITC). Annual Report 2005. http://www.citc.gov.sa/en/mediacenter/annualreport/Documents/PR_REP_001E.pdf. Online; accessed 14 April 2020.

[87] Communications and Information Technology Commission (CITC). Annual Report 2016. , available at http://www.citc.gov.sa/en/mediacenter/annualreport/Documents/PR_REP_012Eng.pdf, year = 2016.

[88] Communications and Information Technology Commission (CITC). Block Website Request. www.filter.sa. Online; accessed 14 April 2020.

[89] Communications and Information Technology Commission (CITC). CITC receives more than one million applications for blocking links by the end of 2017. http://www.citc.gov.sa/ar/MediaCenter/CITCinthemedia/Pages/2018051301.aspx. Online; accessed 27 April 2020.

[90] Communications and Information Technology Commission (CITC). Content Filtering in Saudi Arabia. http://www.internet.sa/ar/content-filtering-in-saudi-arabia-ar/. Online; accessed 14 April 2020.

[91] Communications and Information Technology Commission (CITC). General Information on Filtering Service. http://web1.internet.sa/en/general-information-on-filtering-service/. Online; accessed 14 April 2020.

[92] Communications and Information Technology Commission (CITC). Anti-Cyber Crime Law. https://www.citc.gov.sa/en/RulesandSystems/CITCSystem/Pages/CybercrimesAct.aspx, 2007.

[93] Communications and Information Technology Commission (CITC). Blocking protects Internet users from the damage caused by some inappropriate sites. https://www.citc.gov.sa/ar/mediacenter/citcinthemedia/Pages/PR_MED_026A.aspx, 2010.

[94] Communications and Information Technology Commission (CITC). In line with the needs of the user and in line with global trends, CITC announces the launch of Internet communications applications. http://www.citc.gov.sa/ar/mediacenter/pressreleases/Pages/2017092001.aspx, 2017.

[95] Congdon, P., Sanchez, M., and Aboba, B. Radius attributes for virtual lan and priority support. *Internet Engineering Task Force, Request for Comment 4675* (2006), 1–13.

[96] Coppersmith, D. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *Journal of Cryptology 10*, 4 (1997), 233–260.

[97] Coppersmith, D., Franklin, M., Patarin, J., and Reiter, M. Low-exponent rsa with related messages. In *International Conference on the Theory and Applications of Cryptographic Techniques* (1996), Springer, pp. 1–9.

[98] COUNCIL OF THE EUROPEAN UNION. Joint meeting of the Law Enforcement Working Party and the Customs Cooperation Working Party. `http://register.consilium.europa.eu/doc/srv?l=EN&f=ST20718120201120COR201`, 2011.

[99] Crocker, D., Hansen, T., and Kucherawy, M. Domainkeys identified mail (dkim) signatures. Tech. rep., RFC 6376, September, 2011.

[100] CSaudi Gazette. Full text of Saudi Arabia's Vision 2030. `https://english.alarabiya.net/en/perspective/features/2016/04/26/Full-text-of-Saudi-Arabia-s-Vision-2030.html`, 2016. Al-Arabiya News.

[101] Cusack, B., Antony, B., Ward, G., and Mody, S. Assessment of security vulnerabilities in wearable devices.

[102] Cyber Security for Kids. Cyber Security Initiative for Children: Voluntary Education, Training and Outreach. `https://www.cybersec4kids.com/`.

[103] Cyr, B., Horn, W., Miao, D., and Specter, M. Security analysis of wearable fitness devices (fitbit). *Massachusetts Institute of Technology 1* (2014).

[104] Dagon, D., Antonakakis, M., Vixie, P., Jinmei, T., and Lee, W. Increased dns forgery resistance through 0x20-bit encoding: security via leet queries. In *ACM CCS* (2008), pp. 211–222.

[105] Dainotti, A., Squarcella, C., Aben, E., Claffy, K. C., Chiesa, M., Russo, M., and Pescapé, A. Analysis of country-wide internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (2011), ACM, pp. 1–18.

[106] Damas, J., Graff, M., and Vixie, P. RFC 6891 - Extension Mechanisms for DNS (EDNS0). `https://tools.ietf.org/html/rfc6891`, 2013.

[107] Das, A. K., Pathak, P. H., Chuah, C.-N., and Mohapatra, P. Uncovering privacy leakage in ble network traffic of wearable fitness trackers. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications* (2016), pp. 99–104.

[108] Davi, L., Dmitrienko, A., Sadeghi, A.-R., and Winandy, M. Privilege escalation attacks on android. In *international conference on Information security* (2010), Springer, pp. 346–360.

[109] Denis, F. DNSCrypt. `https://www.dnscrypt.org/`, 2015.

[110] Deogirikar, J., and Vidhate, A. Security attacks in iot: A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)* (2017), IEEE, pp. 32–37.

[111] Department, S. R. Internet of things - number of connected devices worldwide 2015-2025, 2019. Available at https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[112] Devry, J. Mirai botnet infects devices in 164 countries, 2019.

[113] Dignan, L. Dyn confirms mirai botnet involved in distributed denial of service attack. *ZDNet* (2016).

[114] Ding, X., Mozzacchi, D., and Tsudik, G. Experimenting with server-aided signatures.

[115] dnsjava Library Statistics on Andriod. nsjava library statistics on andriod, Visited on 2020-01-09. Available at https://www.appbrain.com/stats/libraries/details/dnsjava/dnsjava.

[116] Dong, Q., Ding, W., and Wei, L. Improvement and optimized implementation of cryptogps protocol for low-cost radio-frequency identification authentication. *Security and Communication Networks 8*, 8 (2015), 1474–1484.

[117] Dornseif, M. Government mandated blocking of foreign web content. *arXiv preprint cs/0404005* (2004).

[118] Douceur, J. R. The sybil attack. In *International workshop on peer-to-peer systems* (2002), Springer, pp. 251–260.

[119] Doukas, C., Maglogiannis, I., Koufi, V., Malamateniou, F., and Vassilacopoulos, G. Enabling data protection through pki encryption in iot m-health devices. In *2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE)* (2012), IEEE, pp. 25–29.

[120] DW. Turkey blocks websites loyal to isis. *DW* (2015). , available at https://p.dw.com/p/1FyTF.

[121] Eastlake 3rd, D. Rfc3110:rsa/sha-1 sigs and rsa keys in the domain name system (dns). Tech. rep., 2001.

[122] Ehrenkranz, T., and Li, J. On the state of ip spoofing defense. *ACM Transactions on Internet Technology (TOIT) 9*, 2 (2009), 6.

[123] Emdadi, A., Karne, R., and Wijesinha, A. Implementing the tls protocol on a bare pc. In *2010 Second International Conference on Computer Research and Development* (2010), IEEE, pp. 293–297.

[124] Engels, D., Fan, X., Gong, G., Hu, H., and Smith, E. M. Hummingbird: ultra-lightweight cryptography for resource-constrained devices. In *International Conference on Financial Cryptography and Data Security* (2010), Springer, pp. 3–18.

[125] Engels, D., Saarinen, M.-J. O., Schweitzer, P., and Smith, E. M. The hummingbird-2 lightweight authenticated encryption algorithm. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues* (2011), Springer, pp. 19–31.

[126] Ensafi, R., Winter, P., Mueen, A., and Crandall, J. R. Analyzing the great firewall of china over space and time. *Proceedings on privacy enhancing technologies 2015*, 1 (2015), 61–76.

[127] Evans, P. Will germany's new law kill free speech online? *BBC News* (2017). , available at `http://www.bbc.com/news/blogs-trending-41042266`.

[128] Facebook. Hard questions: How we counter terrorism. *Facebbok* (2017). , available at `https://about.fb.com/news/2017/06/how-we-counter-terrorism/`.

[129] Fan, C.-I., Chiang, T.-P., and Hsu, R.-H. Light-weight authentication and key exchange protocols with forward secrecy for digital home. *Journal of Computers 18*, 2 (2007), 61–74.

[130] Fawaz, K., Kim, K.-H., and Shin, K. G. Protecting privacy of {BLE} device users. In *25th {USENIX} Security Symposium ({USENIX} Security 16)* (2016), pp. 1205–1221.

[131] Ferguson, N., and Schneier, B. *Practical cryptography*, vol. 141. Wiley New York, 2003.

[132] Fernandes, E., Jung, J., and Prakash, A. Security analysis of emerging smart home applications. In *2016 IEEE Symposium on Security and Privacy (SP)* (2016), IEEE, pp. 636–654.

[133] Fernandes, E., Paupore, J., Rahmati, A., Simionato, D., Conti, M., and Prakash, A. Flowfence: Practical data protection for emerging iot application frameworks. In *25th {USENIX} Security Symposium ({USENIX} Security 16)* (2016), pp. 531–548.

[134] Filasto, A., and Appelbaum, J. Ooni: Open observatory of network interference. In *FOCI* (2012).

[135] Fouque, P.-A., Kunz-Jacques, S., Martinet, G., Muller, F., and Valette, F. Power attack on small rsa public exponent. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2006), Springer, pp. 339–353.

[136] Freedman, M. J. Experiences with coralcdn: A five-year operational view. In *NSDI* (2010), pp. 95–110.

[137] Gaskell, H. Whatsapp's new call service to be blocked in ksa. *ITP.net* (2015). , available at `https://www.itp.net/602475-whatsapps-new-call-service-to-be-blocked-in-ksa`.

[138] Gatlan, S. Iot attacks escalating with a 217.5% increase in volume. *Bleeping Computer* (2019).

[139] Gebhart, G., and Kohno, T. Internet censorship in thailand: User practices and potential threats. In *2017 IEEE European symposium on security and privacy (EuroS&P)* (2017), IEEE, pp. 417–432.

[140] General Authority for Statistics (GASTAT). Annual Yearbook 2018. `https://www.stats.gov.sa/en/46`. Online; accessed 22 October 2019.

[141] General Entertainment Authority. `https://www.gea.gov.sa/en/`. Online; https://www.gea.gov.sa/en/.

[142] Gerez, A. H., Kamaraj, K., Nofal, R., Liu, Y., and Dezfouli, B. Energy and processing demand analysis of tls protocol in internet of things applications. In *2018 IEEE International Workshop on Signal Processing Systems (SiPS)* (2018), IEEE, pp. 312–317.

[143] Gilad, Y., Herzberg, A., and Shulman, H. Off-path hacking: The illusion of challenge-response authentication. *IEEE Security & Privacy 12*, 5 (2014), 68–77.

[144] Griffin, A. Facebook messenger blocked in saudi arabia: Chat apps have voice and video call functions banned over regulations. *Independent* (2016). , available at `https://www.independent.co.uk/life-style/gadgets-and-tech/news/facebook-messenger-blocked-in-saudi-arabia-chat-apps-have-\voice-and-video-call-functions-banned-over-a7027301.html`.

[145] Griffiths, J. University of california tells students not to use wechat, whatsapp in china. *CNN* (2019). , available at `https://www.cnn.com/2019/01/11/asia/university-california-china-wechat-intl/index.html`.

[146] Group, I. . W., et al. Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments. *IEEE Std 802*, 11 (2010).

[147] Gueron, S. Quick verification of rsa signatures. In *2011 Eighth International Conference on Information Technology: New Generations* (2011), IEEE, pp. 382–386.

[148] Gueron, S., and Krasnov, V. Fast prime field elliptic-curve cryptography with 256-bit primes. *Journal of Cryptographic Engineering 5*, 2 (2015), 141–151.

[149] Gunn, A. French assembly passes 'three strikes' hadopi law. *Beta News* (2009). , available at `https://betanews.com/2009/05/12/french-assembly-passes-three-strikes-hadopi-law/`.

[150] Gupta, V., Gupta, S., Chang, S., and Stebila, D. Performance analysis of elliptic curve cryptography for ssl. In *Proceedings of the 1st ACM workshop on Wireless security* (2002), ACM, pp. 87–94.

[151] Gura, N., Patel, A., Wander, A., Eberle, H., and Shantz, S. C. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *International workshop on cryptographic hardware and embedded systems* (2004), Springer, pp. 119–132.

[152] Hastad, J. Solving simultaneous modular equations of low degree. *siam Journal on Computing 17*, 2 (1988), 336–341.

[153] Hazaa, M. A. A popular campaign to close isis accounts on twitter. *Alarabiya* (2015).

[154] Hemaya Group. Saudi Group for Information Assurance. `http://hemayagroup.org/home/`.

[155] Herzberg, A., and Shulman, H. Security of patched dns. In *European Symposium on Research in Computer Security* (2012), Springer, pp. 271–288.

[156] Herzberg, A., and Shulman, H. Fragmentation considered poisonous, or: One-domain-to-rule-them-all. org. In *Communications and Network Security (CNS), 2013 IEEE Conference on* (2013), IEEE, pp. 224–232.

[157] Herzberg, A., and Shulman, H. Vulnerable delegation of dns resolution. In *European Symposium on Research in Computer Security* (2013), Springer, pp. 219–236.

[158] Hilton, S. Dyn analysis summary of friday october 21 attack. `https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/`. Accessed: 2019-09-24.

[159] Hinek, M. J. *Cryptanalysis of RSA and its variants*. Chapman and Hall/CRC, 2009.

[160] Hirose, S., Ideguchi, K., Kuwakado, H., Owada, T., Preneel, B., and Yoshida, H. A lightweight 256-bit hash function for hardware and low-end devices: lesamnta-lw. In *International Conference on Information Security and Cryptology* (2010), Springer, pp. 151–168.

[161] Ho, G., Leung, D., Mishra, P., Hosseini, A., Song, D., and Wagner, D. Smart locks: Lessons for securing commodity internet of things devices. In *Proceedings of the 11th ACM on Asia conference on computer and communications security* (2016), pp. 461–472.

[162] Hoffman, P. Rfc6014:cryptographic algorithm identifier allocation for dnssec. Tech. rep., 2010.

[163] Hoffman, P., and McManus, P. DNS Queries over HTTPS. `https://tools.ietf.org/html/draft-hoffman-dns-over-https-01`, 2017.

[164] Hoffman, P., and Wijngaards, W. Rfc 6605: Elliptic curve digital signature algorithm (dsa) for dnssec. internet engineering task force (ietf), 2012.

[165] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and Hoffman, P. Specification for dns over transport layer security (tls). Tech. rep., 2016.

[166] Huang, L.-S., Adhikarla, S., Boneh, D., and Jackson, C. An experimental study of tls forward secrecy deployments. *IEEE Internet Computing 18*, 6 (2014), 43–51.

[167] Hubert, A., and van Mook, R. RFC 5452 - Measures for Making DNS More Resilient against Forged Answers, 2009.

[168] Huston, G. Apnic, 2018. Available at `https://blog.apnic.net/2018/08/23/measuring-ecdsa-in-dnssec-an-update/`.

[169] ICLab. Uk outs extremism blocking tool and could force tech firms to use it. , available at `https://iclab.org/`.

[170] Initiative, O., et al. Internet filtering in saudi arabia in 2004. *Berkman Center for Internet and Society* (2004).

[171] Institute, I. C. S. The icsi certificate notary, 2018. Available at `https://notary.icsi.berkeley.edu/#statistics`.

[172] IPInfoDB. IP Address Information. `https://ipinfodb.com/`. Online; accessed 25 April 2020.

[173] ISO. International organization for standarization (iso). Available at `https://www.iso.org/standards.html`.

[174] Jackson, C., Barth, A., Bortz, A., Shao, W., and Boneh, D. Protecting browsers from dns rebinding attacks. *ACM Transactions on the Web (TWEB) 3*, 1 (2009), 2.

[175] Jan, M. A., Nanda, P., He, X., Tan, Z., and Liu, R. P. A robust authentication scheme for observing resources in the internet of things environment. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications* (2014), IEEE, pp. 205–211.

[176] Johnson, D., Menezes, A., and Vanstone, S. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security 1*, 1 (2001), 36–63.

[177] Jose, A. C., and Malekian, R. Smart home automation security: a literature review. *SmartCR 5*, 4 (2015), 269–285.

[178] Joye, M., and Michalevsky, Y. Rsa signatures under hardware restrictions. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security* (2018), ACM, pp. 51–54.

[179] Junior, D. M., Melo, L., Lu, H., d'Amorim, M., and Prakash, A. Beware of the app! on the vulnerability surface of smart devices through their companion apps. *arXiv preprint arXiv:1901.10062* (2019).

[180] Kaminsky, D. Black ops 2008: It's the end of the cache as we know it. *Black Hat USA* (2008).

[181] Kaminsky, D. Black ops 2008: It's the end of the cache as we know it. *Black Hat USA* (2008).

[182] Katagi, M., Moriai, S., et al. Lightweight cryptography for the internet of things. *Sony Corporation* (2008), 7–10.

[183] Killalea, T. RFC 3013 - Recommended Internet Service Provider Security Services and Procedures, 2000.

[184] King Abdulaziz City for Science and Technology. ISU History. `https://www.kacst.edu.sa/eng/ScientificServices/ISU/Pages/History.aspx`, Online; accessed 10 May 2018.

[185] King Saud University. Saudi internet rules, 2001. `http://fac.ksu.edu.sa/hidaithy/page/20215`, Online; accessed 27 April 2020.

[186] Kirkland, D. Ubuntu Manpage: systemd-resolved.service, systemd-resolved - Network Name Resolution. `http://manpages.ubuntu.com/manpages/bionic/man8/systemd-resolved.service.8.html`.

[187] Klaba, O. Octave klaba twitter. `https://twitter.com/olesovhcom/status/778830571677978624`, 2016. Accessed: 2019-09-24.

[188] Klein, A. Bind 9 dns cache poisoning. *Report, Trusteer, Ltd 3* (2007).

[189] Klein, A. OpenBSD DNS Cache Poisoning and Multiple O/S Predictable IP ID Vulnerability. `http://www.openbsdsupport.com.ar/books/OpenBSD_DNS_Cache_Poisoning_and_Multiple_OS_Predictable_IP_ID_Vulnerability.pdf.`, 2007.

[190] Klein, A. Windows DNS Server Cache Poisoning . `https://dl.packetstormsecurity.net/papers/attack/Windows_DNS_Cache_Poisoning.pdf`, 2007.

[191] Klein, A., Shulman, H., and Waidner, M. Internet-wide study of dns cache injections. In *IEEE INFOCOM*, pp. 1–9.

[192] Koc, C. K. High-speed rsa implementation version 2.0. *RSA Security* (1994).

[193] Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. Ddos in the iot: Mirai and other botnets. *Computer 50*, 7 (2017), 80–84.

[194] Kolkman, O., Mekking, W., and Gieben, R. Rfc6781:dnssec operational practices, version 2. Tech. rep., 2012.

[195] Kovacs, E. Hacker releases source code of iot malware mirai. *Security Week* (2016).

[196] Krebs, B. Krebsonsecurity hit with record ddos. `https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/`, 2016. Accessed: 2019-09-24.

[197] Larson, M., and Barber, P. RFC 4697 - Observed DNS Resolution Misbehavior. https://tools.ietf.org/html/rfc4697, 2006.

[198] Larson, M., and Gont, F. RFC 6056 - Recommendations for Transport-Protocol Port Randomization. https://tools.ietf.org/html/rfc6056, 2011.

[199] Lashkari, A. H., Danesh, M. M. S., and Samadi, B. A survey on wireless security protocols (wep, wpa and wpa2/802.11 i). In *2009 2nd IEEE International Conference on Computer Science and Information Technology* (2009), IEEE, pp. 48–52.

[200] Legion of the Bouncy Castle. Bouncy castle crypto apis, Visited on 2020-01-09. Available at https://www.bouncycastle.org/java.html.

[201] Levis, P. The collateral damage of internet censorship by dns injection. *ACM SIGCOMM CCR 42*, 3 (2012).

[202] Lim, C. H. Crypton: A new 128-bit block cipher. *NIsT AEs Proposal* (1998).

[203] Lim, C. H., and Korkishko, T. mcrypton–a lightweight block cipher for security of low-cost rfid tags and sensors. In *International Workshop on Information Security Applications* (2005), Springer, pp. 243–258.

[204] Lomas, N. Uk outs extremism blocking tool and could force tech firms to use it. *TechCrunch* (2018). , available at https://techcrunch.com/2018/02/13/uk-outs-extremism-blocking-tool-and-could-force-tech-firms-to\-use-it/.

[205] Lowe, G., Winters, P., and Marcus, M. L. The great dns wall of china. *MS, New York University 21* (2007), 1.

[206] Luo, Y., Chai, Q., Gong, G., and Lai, X. A lightweight stream cipher wg-7 for rfid encryption and authentication. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010* (2010), IEEE, pp. 1–6.

[207] Malina, L., Hajny, J., Fujdiak, R., and Hosek, J. On perspective of security and privacy-preserving solutions in the internet of things. *Computer Networks 102* (03 2016).

[208] Mathrani, A., and Alipour, M. Website blocking across ten countries: A snapshot. In *PACIS* (2010), p. 152.

[209] McDonell, S. https://www.bbc.com/news/blogs-china-blog-48552907. https://www.bbc.com/news/blogs-china-blog-48552907, 2019.

[210] Ministry of Interior. Reporting Cyber Crimes. , available at https://www.my.gov.sa/wps/portal/snp/servicesDirectory/servicedetails/6166.

[211] Ministry of Media. Ministry Regulations. https://www.media.gov.sa/en/page/ministry-regulations. Online; accessed 28 October 2019.

[212] Ministry of Media. Regulations for Digital Publishing Activity. https://www.moci.gov.sa/page/74. Online; accessed 14 April 2020.

[213] Miranda, P., Siekkinen, M., and Waris, H. Tls and energy consumption on a mobile device: A measurement study. In *2011 IEEE Symposium on Computers and Communications (ISCC)* (2011), IEEE, pp. 983–989.

[214] Mockapetris, P. State of IP Spoofing. https://spoofer.caida.org/summary.php. Online; accessed 7 May 2018.

[215] Mockapetris, P. RFC 882 - Domain Names : Concepts and Facilities. https://tools.ietf.org/html/rfc882, 1983. updated by RFC 973; obsoleted by RFCs 1034 and 1035.

[216] Mockapetris, P. RFC 883 - Domain Names: Implementation Specification. https://tools.ietf.org/html/rfc883, 1983. updated by RFC 973; obsoleted by RFCs 1034 and 1035.

[217] Mockapetris, P. RFC 973 - Domain system changes and observations. https://tools.ietf.org/html/rfc973, 1986. obsoleted by RFCs 1034 and 1035.

[218] Mockapetris, P. RFC 1034 - Domain Names - Concepts and Facilities. https://www.ietf.org/rfc/rfc1034.txt, 1987. updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, 8020.

[219] Mockapetris, P. RFC 1035 - Domain Names - Implementation and Specification. https://www.ietf.org/rfc/rfc1035.txt, 1987. updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604, 7766.

[220] Nabi, Z. The anatomy of web censorship in pakistan. In *FOCI* (2013).

[221] Nawir, M., Amir, A., Yaakob, N., and Lynn, O. B. Internet of things (iot): Taxonomy of security attacks. In *2016 3rd International Conference on Electronic Design (ICED)* (2016), IEEE, pp. 321–326.

[222] Nguyen, K. T., Oualha, N., and Laurent, M. Authenticated key agreement mediated by a proxy re-encryptor for the internet of things. In *European Symposium on Research in Computer Security* (2016), Springer, pp. 339–358.

[223] Nir, Y., Josefsson, S., and Pegourie-Gonnard, M. Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls) versions 1.2 and earlier. *Internet Requests for Comments, RFC Editor, RFC 8422* (2018).

[224] NIST, C. The digital signature standard. *Communications of the ACM 35*, 7 (1992), 36–40.

[225] Noon. Iraq: Minister of communications: Blocking isis websites tops our ministerial program. *Noon* (2014). , available at http://www.non14.net/public/56224.

[226] OpenNet Initiative. Opennet initiative. , available at https://opennet.net/research/data.

[227] Osborne, C. Anonymous targets isis social media, recruitment drives in opisis campaign. *ZDNet* (2015). , available at https://www.zdnet.com/article/anonymous-targets-isis-social-media-recruitment-drives-\in-opisis-campaign/.

[228] Pearce, P., Jones, B., Li, F., Ensafi, R., Feamster, N., Weaver, N., and Paxson, V. Global measurement of {DNS} manipulation. In *26th {USENIX} Security Symposium ({USENIX} Security 17)* (2017), pp. 307–323.

[229] Perdisci, R., Antonakakis, M., Luo, X., and Lee, W. Wsec dns: Protecting recursive dns resolvers from poisoning attacks. In *IEEE/IFIP DSN* (2009), pp. 3–12.

[230] Pi, R. , available at https://www.raspberrypi.org/.

[231] Porambage, P., Braeken, A., Kumar, P., Gurtov, A., and Ylianttila, M. Proxy-based end-to-end key establishment protocol for the internet of things. In *2015 IEEE International Conference on Communication Workshop (ICCW)* (2015), IEEE, pp. 2677–2682.

[232] Potlapally, N. R., Ravi, S., Raghunathan, A., and Jha, N. K. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on mobile computing 5*, 2 (2005), 128–143.

[233] Power Meter Store. Watts up pro portable power meter, Visited on 2020-01-09. Available at https://www.powermeterstore.com/p1206/watts_up_pro.php.

[234] Qian, Z., and Mao, Z. M. Off-path tcp sequence number inference attack-how firewall middleboxes reduce security. In *Security and Privacy (SP), 2012 IEEE Symposium on* (2012), IEEE, pp. 347–361.

[235] Qian, Z., Mao, Z. M., and Xie, Y. Collaborative tcp sequence number inference attack: how to crack sequence number under a second. In *ACM CCS* (2012), pp. 593–604.

[236] Raza, S., Seitz, L., Sitenkov, D., and Selander, G. S3k: Scalable security with symmetric keys—dtls key establishment for the internet of things. *IEEE Transactions on Automation Science and Engineering 13*, 3 (2016), 1270–1280.

[237] Raza, S., Shafagh, H., Hewage, K., Hummen, R., and Voigt, T. Lithe: Lightweight secure coap for the internet of things. *IEEE Sensors Journal 13*, 10 (2013), 3711–3720.

[238] Reisinger, D. Twitter has suspended 1.2 million terrorist accounts since 2015. *Fortune* (2018). , available at https://p.dw.com/p/1FyTF.

[239] Rescorla, E. Http over tls.

[240] Rescorla, E. Rfc8446:the transport layer security (tls) protocol version 1.3. Tech. rep., 2018.

[241] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.

[242] Robshaw, M. The estream project. In *New Stream Cipher Designs*. Springer, 2008, pp. 1–6.

[243] Rose, S. Rfc6944:applicability statement: Dns security (dnssec) dnskey algorithm implementation status.

[244] Saba, M. Will 'american dad' define the saudis for us? *Arab News* (2005). , available at `http://www.arabnews.com/node/277392`.

[245] Samaila, M., Sequeiros, B., Correia, A., Freire, M., and InÃ¡cio, P. *IoT Hardware Development Platforms: Past, Present, and Future*. 03 2018, pp. 107–139.

[246] Sar, E. V. Saudi arabia government blocks the pirate bay (and more). *TorrentFreak* (2014). , available at `https://torrentfreak.com/saudi-arabia-government-blocks-pirate-bay-140402/`.

[247] Saudi Vision 2030. Vision 2030. `http://vision2030.gov.sa/en`. Online; accessed 14 April 2020.

[248] Schindler, W., and Itoh, K. Exponent blinding does not always lift (partial) spa resistance to higher-level security. In *International Conference on Applied Cryptography and Network Security* (2011), Springer, pp. 73–90.

[249] Schomp, K., Allman, M., and Rabinovich, M. Dns resolvers considered harmful. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks* (2014), ACM, p. 16.

[250] Schuba, C. Addressing weaknesses in the domain name system protocol. *Master's thesis, Purdue University, West Lafayette, IN* (1993).

[251] Semeraro, G., Magklis, G., Balasubramonian, R., Albonesi, D. H., Dwarkadas, S., and Scott, M. L. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proceedings Eighth International Symposium on High Performance Computer Architecture* (2002), IEEE, pp. 29–40.

[252] Sfakianakis, A., Athanasopoulos, E., and Ioannidis, S. Censmon: A web censorship monitor. In *USENIX Workshop on Free and Open Communication on the Internet (FOCI)* (2011).

[253] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., and Iwata, T. The 128-bit blockcipher clefia. In *International workshop on fast software encryption* (2007), Springer, pp. 181–195.

[254] Shulman, H., and Waidner, M. Fragmentation considered leaking: port inference for dns poisoning. In *International Conference on Applied Cryptography and Network Security* (2014), Springer, pp. 531–548.

[255] Simmons, G. J. A "weak" privacy protocol using the rsa crypto algorithm. *Cryptologia 7*, 2 (1983), 180–182.

[256] Simmons, G. J. The prisoners' problem and the subliminal channel. In *Advances in Cryptology* (1984), Springer, pp. 51–67.

[257] Srisuresh, P., and Holdrege, M. RFC 2663 - IP network address translator (NAT) terminology and considerations. `https://tools.ietf.org/html/rfc2663`, 1999.

[258] Steinfeld, R., and Zheng, Y. On the security of rsa with primes sharing least-significant bits. *Applicable Algebra in Engineering, Communication and Computing 15*, 3-4 (2004), 179–200.

[259] Thales. 2018 thales data threat report - global edition. Tech. rep., 2018.

[260] The New Arab. Saudi Arabia blocks Turkish news sites Anadolu, TRT amid continued Ankara-Riyadh tensions. available at `https://english.alaraby.co.uk/english/news/2020/4/12/saudi-arabia-blocks-turkish-news-sites-anadolu-trt`, note = Online; accessed 14 April 2020 , year = 2020.

[261] Tian, Y., Zhang, N., Lin, Y.-H., Wang, X., Ur, B., Guo, X., and Tague, P. Smartauth: User-centered authorization for the internet of things. In *26th {USENIX} Security Symposium ({USENIX} Security 17)* (2017), pp. 361–378.

[262] Ushe, S. Saudi arabia blocks viber messaging service. *BBC News* (2013). , available at `https://www.bbc.com/news/world-middle-east-22806848`.

[263] Valenta, L., Sullivan, N., Sanso, A., and Heninger, N. In search of curveswap: Measuring elliptic curve implementations in the wild. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)* (2018), IEEE, pp. 384–398.

[264] van Rijswijk-Deij, R., Jonker, M., and Sperotto, A. On the adoption of the elliptic curve digital signature algorithm (ecdsa) in dnssec. In *2016 12th International Conference on Network and Service Management (CNSM)* (2016), IEEE, pp. 258–262.

[265] van Rijswijk-Deij, R., Sperotto, A., and Pras, A. Making the case for elliptic curves in dnssec. *ACM SIGCOMM Computer Communication Review 45*, 5 (2015), 13–19.

[266] Verizon. Data breach digest: Perspective is reality. Tech. rep., 2017.

[267] Verkamp, J.-P., and Gupta, M. Inferring mechanics of web censorship around the world. In *FOCI* (2012).

[268] Vixie, P. Dns and bind security issues. In *Usenix Security* (1995).

[269] Wang, X., Sun, Y., Nanda, S., and Wang, X. Looking from the mirror: evaluating iot device security through mobile companion apps. In *28th {USENIX} Security Symposium ({USENIX} Security 19)* (2019), pp. 1151–1167.

[270] Want, R. An introduction to rfid technology. *IEEE pervasive computing*, 1 (2006), 25–33.

[271] Weiler, S. Rfc3755:legacy resolver compatibility for delegation signer (ds). Tech. rep., 2004.

[272] Weiler, S., and Blacka, D. Rfc 6840 - clarifications and implementation notes for dns security (dnssec). `https://tools.ietf.org/html/rfc6840`, 2013.

[273] Wiener, M. J. Cryptanalysis of short rsa secret exponents. *IEEE Transactions on Information theory 36*, 3 (1990), 553–558.

[274] Wire Filter. `http://www.wirefilter.com/`. Online; accessed 28 April 2020.

[275] writer, S. 'isis is enemy no. 1 of islam,' says saudi grand mufti. *Al-Arabiya News* (2014). , available at `https://english.alarabiya.net/en/News/middle-east/2014/08/19/Saudi-mufti-ISIS-is-enemy-No-1-of-Islam-.html`.

[276] Xu, X., Mao, Z. M., and Halderman, J. A. Internet censorship in china: Where does the filtering occur? In *International Conference on Passive and Active Network Measurement* (2011), Springer, pp. 133–142.

[277] Yan, Z., Zhang, P., and Vasilakos, A. V. A survey on trust management for internet of things. *Journal of network and computer applications 42* (2014), 120–134.

[278] Yao, X., Han, X., Du, X., and Zhou, X. A lightweight multicast authentication mechanism for small scale iot applications. *IEEE Sensors Journal 13*, 10 (2013), 3693–3701.

[279] Zhang, K., Ding, L., and Guan, J. Cryptanalysis of hummingbird-2. Tech. rep., Cryptology ePrint Archive, Report 2012/207, 2012.

[280] Zhang, Q., and Liang, Z. Security analysis of bluetooth low energy based smart wristbands. In *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)* (2017), IEEE, pp. 421–425.

[281] Zhang, W., Subramanian, N., and Wang, G. Lightweight and compromise-resilient message authentication in sensor networks. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications* (2008), IEEE, pp. 1418–1426.

[282] Zittrain, J., and Edelman, B. *Documentation of internet filtering in Saudi Arabia*. Harvard Law School, 2002.

[283] Zuo, C., Wen, H., Lin, Z., and Zhang, Y. Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019), pp. 1469–1483.