NotaryCache: Preventing man-in-the-middle attacks through ubiquitous multi-path probing

Master-Thesis von Fabian Letzkus
Tag der Einreichung:

1. Gutachten: Prof. Dr. Johannes Buchmann
2. Gutachten: Johannes Braun

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Kryptographie und Computeralgebra

NotaryCache: Preventing man-in-the-middle attacks through ubiquitous multi-path probing

Vorgelegte Master-Thesis von Fabian Letzkus

1. Gutachten: Prof. Dr. Johannes Buchmann
2. Gutachten: Johannes Braun

Tag der Einreichung:

Abstract

There are various design issues with todays WebPKI, ranging from technical issues, such as buggy implementations, to organizational and governance issues. As a consequence, the WebPKI, that constitutes an essential part in todays internet security, is prone to attacks and threats from various actors. This problem is also strengthened by the large size of WebPKI, which prevents fixing the design issues within a short-range. However, to protect a client against different threat actors, few approaches have been evolved in research, which extend the existing WebPKI with different services.

The concept of multi-path notaries is one approach to deal with man-in-the-middle attacks. In this concept, special hosts on the internet called notaries are asked to establish a TLS connection to a target host. The notaries then establish a TLS connection to the target host and return the received X.509 certificate back to the client. The client is then able to compare the X.509 certificate, that was received during his own connection establishment, with the X.509 certificate, that was returned by the notary. The goal of this comparison is to give the client the ability to identify forged X.509 certificates, that were knowingly or unknowingly issued to an adversary by the WebPKI. If this process is repeated with different notaries around the world, then man-in-the-middle adversaries, such as rouge ISPs or governments, could be detected.

However, notaries face different problems, which lead to the fact, that existing implementations are unattractive to both endusers and server operators, which results in a low distribution of the concept. To better understand the goals and incentives of endusers and server operators, the current situation is examined and a set of requirements for notaries is established. To overcome possible gaps in existing notaries and to prevent the same issues in future implementations, NotaryCache is presented. NotaryCache abstracts from the actual notary to provide a common interface for endusers. It extends each notary with a cache, that can be downloaded by a client asynchronously to evaluate target hosts without connecting to the respective notary during connection establishment with the target host. In contrast to existing notaries, latency is not affected, if the target host is found in the cache. Moreover, NotaryCache extends the cached data with additional information, that can be used for service discovery of notary services or other NotaryCache deployments to prevent user configurable lists of notaries and their cryptographic material. NotaryCache also gives automatic mechanisms to restrict the size of the cache and the number of requests the server will handle to restrict disk and traffic usage. This mechanisms are supported by a context-dependent configuration to avoid complex installation and maintenance, which could otherwise exclude users due to missing expert knowledge. Moreover, NotaryCache introduces replication services to distribute caches to reduce load on the server, as well as monitoring services to give further incentives to take part in this ecosystem. Lastly, NotaryCache is evaluated regarding its resource consumption for an individual user as well as for the internet. It is shown, that NotaryCache could provide benefits for endusers, server operators and businesses.

Zusammenfassung

Internetsicherheit basiert heute im Wesentlichen auf Komponenten und Mechanismen der WebPKI, mit deren Hilfe die Authentizität von kryptographischen Schlüsseln und deren Besitzern bewiesen werden soll. In ihrer derzeitigen Form weist die WebPKI jedoch substantielle Mängel in technischen, wie auch in organisatorischen Aspekten auf. Die WebPKI sieht sich aufgrund dieser Schwachstellen einer steigenden Anzahl an Bedrohungen ausgesetzt, die die Unterbindung von essentiellen Schutzzielen wie Integrität und Vertraulichkeit von Internetkommunikation zum Ziel haben. Verstärkt wird das Problem durch die Reichweite der WebPKI, die schnelle Maßnahmen zur Lösung dieser Probleme verhindert. Gleichwohl konnten in der Wissenschaft mehrere mögliche Erweiterungen der WebPKI entwickelt werden, um Maßnahmen gegen bestimmte Bedrohungen und Angreifer zu treffen.

Ein Konzept zur Abwehr von Man-in-the-middle Angreifern stellen Multi-Path Notaries dar. In diesem Konzept werden die von einem Client während des Verbindungsaufbaus erhaltenen Zertifikate mittels sogenannter Notarserver überprüft. Notarserver bauen hierbei auf Anfrage des Clients eine Verbindung zum Zielserver auf und melden dem Client das erhaltene Zertifikat bzw. dessen Fingerabdruck zurück. Der Client wird somit in die Lage versetzt, das erhaltene Zertifikat mit dem des Notarservers zu vergleichen und ein gefälschtes Zertifikat, das durch einen Angreifer zugeschoben wurde, zu erkennen.

Jedoch sieht sich der Notaransatz mit mehreren Problemem konfrontiert. Durch diese Probleme bietet der Einsatz von Notarservern bislang noch sowohl für Benutzer, als auch für Serverbetreiber keinen Nutzen, was sich insbesondere in der geringen Verbreitung des Ansatzes äussert. Um die Ziele und Anreize für Benutzer und Serverbetreiber besser zu verstehen und die Ansätze hinsichtlich dieser Aspekte anzupassen, wurde der aktuelle Stand der Forschung hierzu aufgearbeitet und in eine Liste von Anforderungen gegenüber dem Notaransatz zusammengeführt. Weiterhin wurde eine Analyse bestehender Ansätze hinsichtlich der Erfüllung der Anforderungen durchgeführt. Die Lösung der daraus hervorgetretenen Differenzen bildet NotaryCache. NotaryCache abstrahiert vom eigentlichen Notarserver und bietet dadurch nach außen eine einheitliche Schnittstelle zur Implementierung beliebiger Clients. Weiterhin erweitert NotaryCache den Notarserver um einen öffentlichen Cache, welcher vom Benutzer z.B. beim Start des Browsers heruntergeladen wird. Dadurch kann sowohl die Anzahl der Anfragen an den Notarserver, als auch die Latenz im Vergleich zu bisherigen Notarservern reduziert werden. Der Cache wird darüber hinaus mit Informationen zu Diensterkennung und Bootstrapping angereichert, wodurch Benutzerinteraktion, wie z.B. das Hinzufügen neuer Notarserver, vermieden wird. Durch eine automatische Anpassung an die Systemlast wird zusätzlich die Attraktivität für Serverbetreiber gesteigert. Darüber hinaus bietet NotaryCache Replikationsdienste zur Verteilung der Netzwerklast beim Download der Caches sowie Monitoringdienste zur Integration in Zertifikatsmanagementsysteme und als mögliche Dienstleistung für CA Betreiber. Abschliessend wird eine Evaluation von NotaryCache durchgeführt. Hierzu wird die Ressourcenauslastung bei einem einzelnen Client evaluiert und auf den globalen Einsatz hochgerechnet. Hierbei wird gezeigt, dass NotaryCache dazu geeignet ist, die Anzahl an Requests an Notarserver zu reduzieren.

Contents

List of Figures

1 Introduction

As the internet grows, secure communication also gets more and more important. Today several services and websites rely on security to protect their users. For example, users may comprise of customers, whose credit card numbers must be handled securely according to regulatory frameworks such as PCI DSS. Users may also comprise of employees, transferring internal and confidential documents or other sensitive data to the company network over the internet. Users may also comprise of citizens, who are using applications from the e-government domain. In some countries, users may also comprise of members of the political oppositions, who want to communicate securely under the rights of free speech or freedom of the press, especially, when it comes to protests and revolutions in authoritarian countries like it was seen during the Arab Spring. In general, the internet has a growing number of more than three billion internet users worldwide, with around one billion websites [70].

On the internet, one of the main mechanisms to secure connections to web-, mail- or chatservers is Transport Layer Security (TLS), formerly known as Secure Socket Layer (SSL). TLS ensures security goals such as integrity, confidentiality and authenticity by making use of X.509 certificates, that are issued by an infrastructure called WebPKI. In the WebPKI, certificate authorities (CAs) are responsible for the issuance of certificates to their legitimate owners. Here, legitimate means, that only the respective owner of a domain is allowed to be provided with a certificate for that domain by a CA from the WebPKI. For example, only Google, Inc. can request certificates for the domain *google.com*, while anyone else is not allowed to. As of October 2014, TLS is enabled at approximately 60% of all websites of the Alexa Top one million websites [29] and is supported by nearly all current browsers, making it an essential internet technology.

Despite its acceptance and importance, the WebPKI is fundamentally broken. The main reason for this conclusion is, that each CA in todays WebPKI is able to issue certificates for each domain. By 2015, there is no technical or organizational measurement which could prevent issuing certificates to illegitimate entities. In the past there has yet been a multitude of incidents at major CAs which caused rouge certificates to be issued, such as Comodo [23], DigiNotar [62], Turktrust [22] or CNNIC [75], which either fell victim to attackers or got compelled by governments to issue certificates, which were later used in attacks against various services, such as Google's mail service. This problem is strengthened by the huge number of CAs in the WebPKI. Researches have identified up to 1590 trusted CAs, which belong to more than 683 organizations [32]. Many of these CAs are operated by governmental organizations which usually issue certificates for state-driven sites. It is also known that intelligence agencies undertake huge efforts to break TLS connections [77, 58, 15].

There have been several approaches to strengthen the security by extending the SSL/TLS ecosystem with additional services. One promising approach is the notary approach. In the notary approach, a new entity is created, which is called notary. A notaries task is to acknowledge a certificate, that was received by a client. To acknowledge a certificate, the notary has to be provided with the target host by the client.

From the clients perspective, a second path is created and the certificate is acknowledged using this path. In theory, this concept does prevent locally restricted attacks, for example by intelligence agencies.

Notaries suffer from various issues, ranging from high latencies at enduser side, missing bootstrapping and service discovery methods, to usability of existing implementations. So far, these issues prevented a widespread deployment yet. However, a widespread deployment is crucial for this approach. To address these problems, NotaryCache was developed. NotaryCache introduces a cache for each notary, that contains information about hosts, that were recently requested from notaries. It will be shown, that a cache can solve problems regarding latency, bootstrapping and service discovery. Moreover, NotaryCache provides benefits for various business cases, for example in certificate management. NotaryCache also extends the ecosystem with monitoring and replication services for additional business incentives and cost savings.

To explain the need for NotaryCache, chapter 2 gives an introduction to the background of this topic. In chapter 2, the WebPKI and the security model are introduced and the notary approach is described. Chapter 2 also contains a section about related work to give an overview of existing notary implementations and other mechanisms besides notaries. Chapter 3 covers the proposed solution. To reveal the gaps in current implementations, requirements for a solution are established. Based on these requirements, NotaryCache is introduced. Chapter 4 briefly describes the proof-of-concept implementation of NotaryCache, which is delivered with this thesis. In chapter 5, an evaluation of NotaryCache is done. Here, both the perspective of an individual user and the global perspective are covered and it is shown, how NotaryCache behaves in different situations. Chapter 6 draws the conclusion and closes the thesis with a short section for future work.

2 Background

The following chapter gives an introduction to the background for that thesis. First, the system model is described. Then, the security model, which consists of the attacker model and the concrete attack, and the protection mechanism, that is addressed in this thesis, will be explained. The chapter is closed with a statement of the related work, that was previously done in this field.

2.1 System model

As shown in the introduction, the internet is an essential part of our daily life. The basis for the internet is formed by the underlying infrastructure, which will be introduced in the next section. As the need for security on the internet increases, the common security goals and the cryptographic methods, that are used to achieve these goals, are described. After the definition of the security goals, Transport Layer Security (TLS) is introduced, which implements the theoretical mechanisms to achieve security on the internet. To achieve these goals, TLS makes use of a infrastructure called WebPKI, which is presented afterwards.

2.1.1 Basic infrastructure

In the context of this thesis, the basic infrastructure comprises of the Internet Protocol, its routing mechanisms and TCP, as well as the domain name system and the HTTP protocol.

Routing, IP and TCP

Figure 2.1 shows the basic structure of the internet. The internet is a network of networks. The networks on the internet are called autonomous systems (ASes), as they are managed by organizations or individuals autonomously, for example universities, companies or governments. Autonomous systems can be connected to other autonomous systems either directly via physical cable or via internet exchange points (IXP). IXPs provide the infrastructure to connect autonomous systems to several other autonomous systems by utilizing only a small number of physical connections. Todays internet consists of 50540 autonomous systems [17]. However, multiple ASes can also be operated by one organization, which is called Internet Service Provider (ISP). These organizations are differentiated by their size and their relationship to each other:

- Tier 3 ISPs provide network access to endusers and businesses. They usually operate local and regionally limited networks. ASes in Tier 3 networks are shown in blue in figure 2.1.
- Tier 2 ISPs interconnect Tier 3 networks, by providing mid scale networks to offer connections to multiple countries or regions. ASes in Tier 2 networks are shown in orange in figure 2.1.

- Tier 1 ISPs operate and provide access to the internet backbone. They are used to communicate over multiple countries or continents. An example AS is shown by AS1 in figure 2.1.

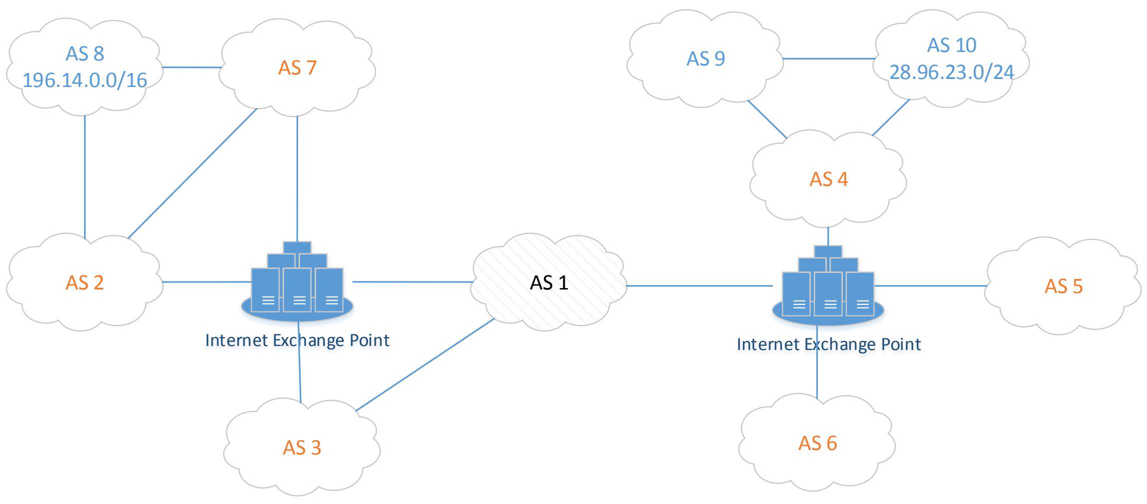

Figure 2.1: Internet architecture

Routing on the internet follows a best effort approach. Best effort means, that no resources are guaranteed and packets are routed as fastest as possible to the next destination. If a connection to one AS is busy, then another connection must be used.

The hosts on the internet are interconnected via the Internet Protocol (IP), which assigns each host a unique 32bit (IPv4) or 128bit (IPv6) IP address. To determine the correct destination of a packet, IP addresses are split into two parts. The first bits are called prefix. The prefix is used to address the network. The last bits are used by the respective network to determine the host, to which the packet is addressed to. To communicate with hosts of other autonomous systems, routing protocols like the Border Gateway Protocol (BGP) are used to determine the most efficient path towards all directly connected ASes. Figure 2.2 shows an example connection between two hosts. As one can see, packets are send over multiple AS, from a client within a network operated by a Tier-3 ISP down to a Tier-1 ISP, which transfers the packet, to a Tier-3 provider providing access to the target host.

IP is a stateless protocol, which means, that it does not carry information about the state or the desired purpose of a connection in its headers. To create a stateful connection, the Transmission Control Protocol (TCP) is used. TCP also specifies the purpose of the connection to a target host by adding so called port numbers, which can be used by an operating system to determine the service the packet must be delivered to. From a server operators perspective, a service can be accessed via IP and TCP, if it listens on the public accessible IP address and a TCP port.

Domain Name System

To enable human users to use memorizable names instead of IP addresses, the domain name system (DNS) was created. DNS utilizes a hierarchical naming system, whose elements are called domains or domain names, which are managed by authoritative nameserver for each domain. An authoritative

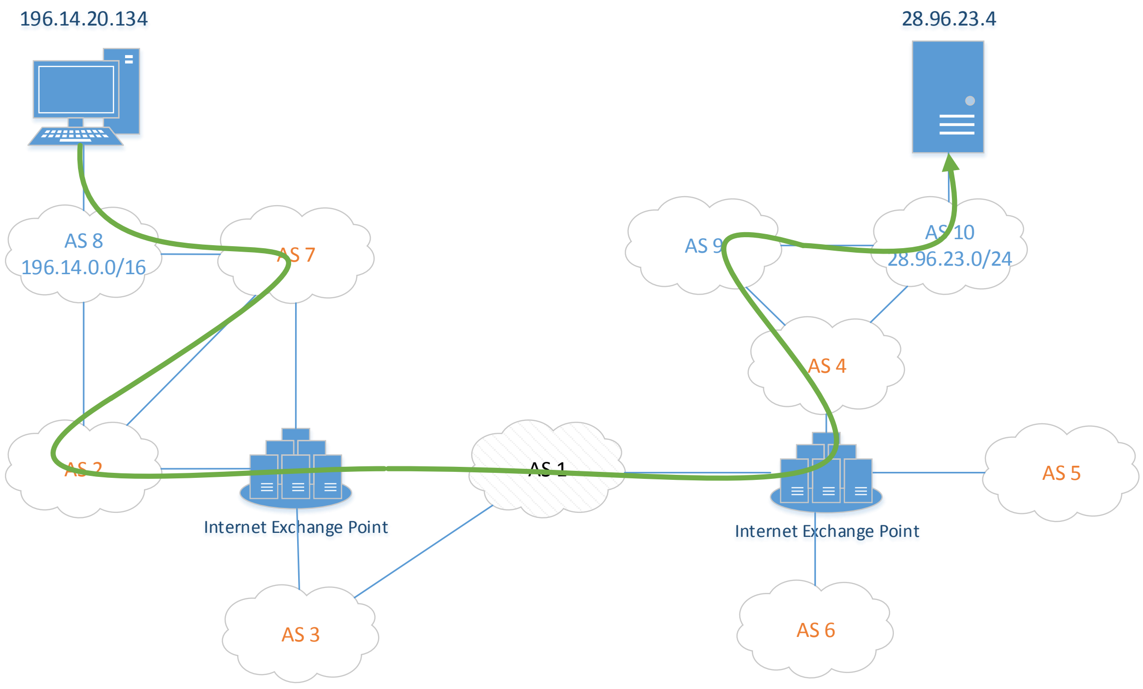

Figure 2.2: Communication on the internet

nameserver basically maps domain names to IP addresses, as well as states the responsible nameserver for lower hierarchies. The first layer of the hierarchy is given by the root zone. The root zone comprises of a fixed set of standardized nameservers, which manage a list of references to the nameservers of so called top-level domains, such as “net” or “com”. Likewise, the top-level nameservers manage a list of references to the actual domains of businesses, individuals or other service providers. An example is shown in figure 2.3.

Basically, the domain name resolution is iterative. To resolve a domain name, a client first requests the address of the nameserver of the top-level-domain from the root zone nameservers. Then, after receiving the addresses, the client requests the address of the authoritative nameserver for the next hierarchy of the domain name from the top-level nameservers. This process continues until the responsible authoritative nameserver is found and the domain name could be resolved. To reduce traffic, clients usually make use of resolvers, that enable caching to prevent resolving domain names, that are often requested. From a client perspective, the process is now recursive: The client requests the IP address for a given domain name from a resolver. The resolver then requests this domain name either directly using the iterative approach, or again, requests the IP address from another resolver.

DNS stores information in resource records (RR), which can individually be requested by a client. There are several types of RRs to store information for different purposes. Besides special purpose RRs, there are three important RRs:

- The SOA-RR gives information about the validity of the domain or the main contact point.

- The NS-RR contains a list of nameservers, that are authoritative for this domain.

- The A-RR (IPv4) or AAAA-RR (IPv6) maps a domain or subdomain from the domain name space to an IP-Address from the IP address space.

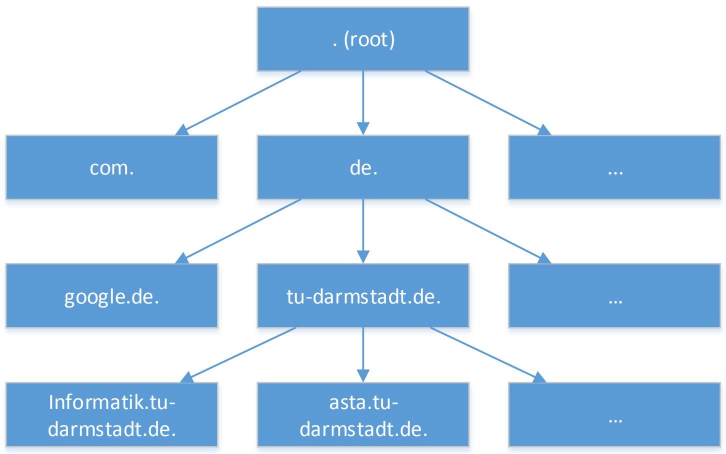

Figure 2.3: Domain Name System

If a domain or subdomain is requested from a nameserver, which could not be found in the database, the nameservers returns a statuscode, which is called NXDOMAIN.

Hypertext Transfer Protocol

If a domain name could successfully be resolved, the client establishes a stateful connection using TCP/IP with the respective host and service. In this thesis, the main service is the web server, which gives clients access to websites and webapplications hosted on this web server. To tell the web server which website should be returned to the client, the Hypertext Transfer Protocol (HTTP) is used. From a users perspective, both port and website address are contained in the uniform resource locator (URL), a special URI, that contains the protocol, the hostname and the website to be requested. As TCP ports are standardized for most protocols and services, the port number is optionally, as the service is determinable using the protocol.

The HTTP protocol gives a client the ability to request documents from a web server by using one of nine requests methods. The main request methods are the GET- and POST-method. They are used to retrieve a document (GET) or add information (POST) to this document. Dependent of the availability of the document or the correctness of the request method, various status codes could be returned by the web server or by the webapplication, that is responsible for executing the request.

2.1.2 Security goals

However, HTTP does not make use of any cryptographic security mechanisms to ensure, for example, confidentiality for user credentials during authentication at a web application. As businesses and endusers more and more rely upon the internet, security is a raising concern of endusers and service providers. There are three main security goals:

Integrity Integrity describes the property of data to not have been modified in an unauthorized way during transmission between sender and receiver.

Confidentiality Confidentiality describes the property of data to be only readable by authorized receivers. Thus, data must be protected against unauthorized exposure, e.g. to third parties during transmission by wiretapping.

Authenticity Authenticity describes the realness and genuineness of sender and receiver. The data transmitted during connection is also described as authentic, if it is integer and its source is flawlessly determinable. The process to determine authenticity is called authentication.

Security goals are achieved by applying cryptographic methods to the data in transmission. There are two cryptographic systems: Symmetric cryptographic systems make use of one key, that must be known to both the sender and the receiver of a message in order to encrypt the plaintext and decrypt the ciphertext. If the key is not known to a third party, than confidentiality is achieved. To achieve integrity, message authentication codes (MACs) are added to the encrypted message to determine changes in the ciphertext. Popular algorithms are the Advanced Encryption Standard (AES) and the Triple Data Encryption Standard (3DES). In symmetric cryptographic systems, the secure distribution of the key is the main problem.

To overcome this problem, asymmetric cryptography is used. Asymmetric cryptography is also called public key cryptography, as is makes use of two distinct keys: A public key, that is made available to everyone and can only be used to encrypt plaintext, and a private key, that can only be used to decrypt the previously encrypted ciphertext. Moreover, the private key can be used to digitally sign a document, while the public key can only be used to verify this signature. Digital signatures make use of cryptographic hash functions, that project the original document to a fixed number of bytes called message digest. Digital signatures provide both authenticity, as only the person in charge of the private key is able to create a signature, and integrity, as changes to the ciphertext would result in changes in the message digest. Popular examples of asymmetric cryptographic systems comprise of RSA, the Digital Signature Algorithm (DSA), approaches from the elliptic key cryptography (ECC) or the Diffie-Hellman method (DH), that is used for key exchange.

However, asymmetric cryptography is slow as compared to symmetric cryptography. Thus, the sender only encrypts the key material, that can than be used by symmetric cryptographic systems to encrypt the data stream and to create message authentication codes, by using the public key of the receiver. Since only the receiver is able to decrypt the message, both parties can securely communicate with each other.

2.1.3 Transport Layer Security

In practice, asymmetric and symmetric cryptography is implemented in the Transport Layer Security protocol family (TLS). Transport Layer Security (formally known as Secure Socket Layer/SSL) is a family of cryptographic protocols defined by the Internet Engineering Task Force [28]. Its primary goal is to "prevent eavesdropping, tampering, or message forgery" in communication protocols by ensuring confidentiality, integrity and authenticity. TLS encapsulates other protocols on top of a reliable transport protocol like the TCP protocol and is thus used by a lot of popular protocols on the internet to enable private and reliable communication, for example HTTP over TLS (HTTPS). Protection is done by making use of (slow) asymmetric cryptography for negotiating the master key and fast symmetric cryptography and hash-based message authentication codes for encrypting data and ensuring integrity and confidentiality by using this master key. Negotiation is done via the TLS handshake protocol, which is presented in figure 2.4.

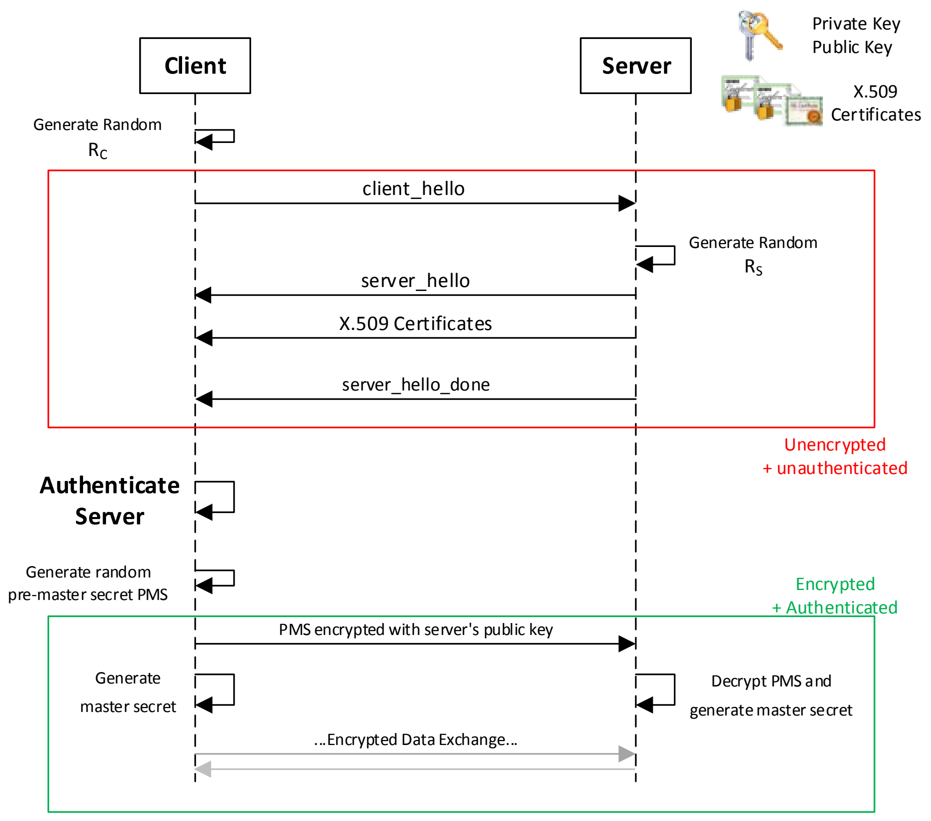
Private Key
Public Key
X.509 Certificates
Client
Server
Generate Random R_C
client_hello
Generate Random R_S
server_hello
X.509 Certificates
server_hello_done
Unencrypted + unauthenticated
Authenticate Server
Generate random pre-master secret PMS
Encrypted + Authenticated
PMS encrypted with server's public key
Generate master secret
Decrypt PMS and generate master secret
...Encrypted Data Exchange...

Figure 2.4: TLS handshake protocol

During the handshake protocol, a set of X.509 certificates is sent to the client. The X.509 certificates are used to authenticate the server towards the client. Besides information about the domain name, X.509 certificates also contain the public key of the server, that is used by the client to securely communicate the parameters, that are used for master key generation. The algorithm that is used to generate the public key given in the X.509 certificate is negotiated with the client_hello and server_hello messages (also called *cipher_suite*). The server does not need to necessarily possess only one private/public key pair. As of April 2015, Facebook.com offers multiple public keys from both RSA and ECC cryptographic systems.

Moreover, the client can also add extensions to the client_hello, on which the server has to respond. The Server Name Indication (SNI) extension is of particular interest, as it allows a client to request the certificates for a given domain name [31]. This extension arose from the need of server operators to assign multiple certificates to one server, that should host multiple domains. In most web server applications, different domains are implemented by creating *virtual hosts*. This leads to the option, that one web server can contain multiple virtual hosts, which are identifiable by the SNI extension. SNI is supported by all current web browsers and web servers. Studies estimate the number of connections using the SNI extension to be around 70% of all TLS traffic [4].

To set the extension, the client adds an additional parameter ServerName to the client_hello message containing the hostname. A server is now able to choose the certificate that must be presented to the client, if it has more than one certificate. If the server does not find the hostname indicated by the client, it has the options to abort the handshake or to continue it and send an alert via the TLS alert protocol. There is currently no mechanism for the client to find out, if the server really chose the virtual host identified by the ServerName.

It is seen in figure 2.4, that up to the authentication of the server, each packet was both unencrypted and unauthenticated. That means, that an attacker could have sent those information, trying to impersonate the server. Thus, the correct authentication of the server is fundamental. If authentication fails, the connection must be seen as insecure and must be aborted. On the internet, authentication is done with X.509 certificates that are issued by the WebPKI.

2.1.4 WebPKI

The foundation of the WebPKI is given by the X.509 framework of the ITU Telecommunication Standardization Sector and the PKIX profile of the Internet Engineering Task Force [24]. X.509 comprises of a hierarchical trust model, in which authorities vouch for the identity of other entities, such as domain owners. The assurance of an identity is embedded in a data structure called certificate. In X.509, authorities are therefore called certificate authorities (CAs). Basically, a certificate binds a public key to an entity, which is called subject. The certificate also includes maintenance information, such as issuer or validity periods, as well as usage-depended extensions, such as usage restrictions. To ensure full authenticity of the certificates, asymmetric cryptography is applied and authorities digitally sign all certificates they issue.

CAs are able issue certificates not only to end entities, but also to intermediate CAs to delegate the responsibility of issuing certificates to other entities. Thus, CAs are organized in a tree structure. The first CA in this tree structure is called Root CA. The certificate of the Root CA is self-signed, which means, that the public key within this certificate must be used to validate the signature of this certificate. The leaves of this tree are formed by end entities not allowed to further issue certificates. A X.509 certificate chain comprises all X.509 certificates needed to validate the X.509 certificate of the leave.

The believe of a client, that a certificate is legitimate for an entity, is called trust. Trust in X.509 is transitive, which means that trust decisions can be made by traversing the certificate chain backwards from a end entities certificate to the Root CA. The process of traversing is also called Certification Path Validation and specified in RFC5280. This process is basically accomplished by verifying all signatures and checking attributes for valid contents, such as validity dates or the subject of the certificate. If at least one CA in this path is trusted, then the certificate of the end entity is trusted. The Root CA is also called trust anchor.

Figure 2.5 shows an example certificate chain for the domain google.com[1]. Here, the root CA is operated by Equifax, which issues a certificate for the Google's internal CA. This CA issues certificates for Google's web server, which is accessible by the domain google.com and port 443.

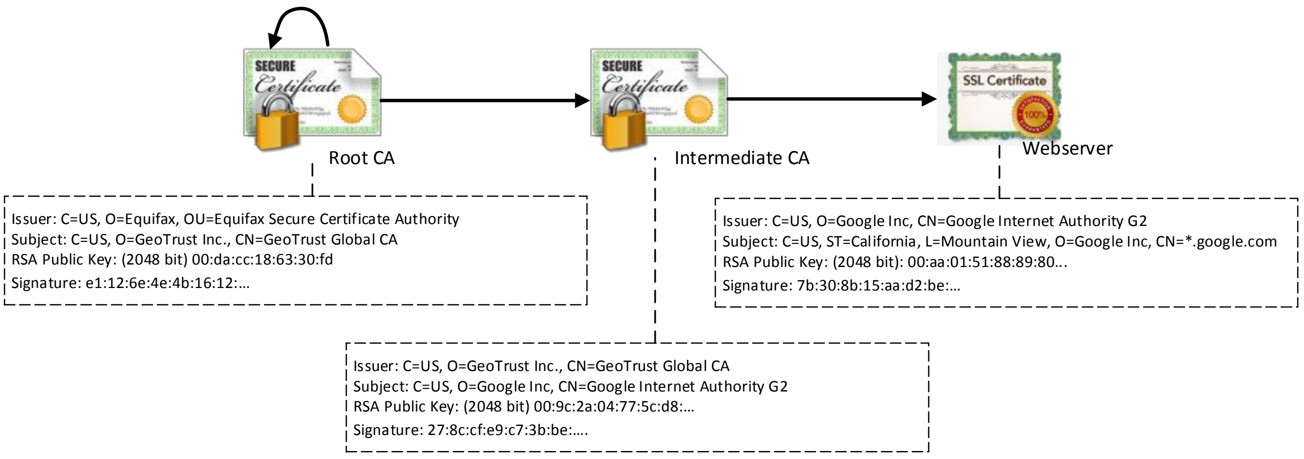
Root CA
Intermediate CA
Webserver
SSL Certificate
Issuer: C=US, O=Equifax, OU=Equifax Secure Certificate Authority
Subject: C=US, O=GeoTrust Inc., CN=GeoTrust Global CA
RSA Public Key: (2048 bit) 00:da:cc:18:63:30:fd
Signature: e1:12:6e:4e:4b:16:12:...
Issuer: C=US, O=Google Inc, CN=Google Internet Authority G2
Subject: C=US, ST=California, L=Mountain View, O=Google Inc, CN=*.google.com
RSA Public Key: (2048 bit): 00:aa:01:51:88:89:80...
Signature: 7b:30:8b:15:aa:d2:be:...
Issuer: C=US, O=GeoTrust Inc., CN=GeoTrust Global CA
Subject: C=US, O=Google Inc, CN=Google Internet Authority G2
RSA Public Key: (2048 bit) 00:9c:2a:04:77:5c:d8:...
Signature: 27:8c:cf:e9:c7:3b:be:....

Figure 2.5: Certificate chain of google.com

To make use of the Domain Name System, PKIX defines a Subject Alternative Name extension, which can hold a arbitrary number of DNS names, IP addresses or Uniform Resource Identifiers. Depending on the type of the certified entity, PKIX requires multiple other attributes to be set, such as usage constraints for the included public key. Nevertheless, the certificate authority is responsible for validating the information given by the requester and sets the required attributes depending on the use case.

```
Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert High Assurance CA-3
Subject: C=US, ST=CA, L=Menlo Park, O=Facebook, Inc., CN=*.facebook.com
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
            Public-Key: (256 bit data)
X509v3 extensions:
    X509v3 Subject Alternative Name: DNS:*.fbsbx.com, DNS:*.fbcdn.net, [...]
Signature Algorithm: sha1WithRSAEncryption
    92:c2:5f:c7:46:10:....
```

Listing 2.1: TLS certificate from facebook.com

Listing 2.1 shows the TLS certificate of the URL facebook.com. The certificate was issued by DigiCert and is not only valid for *.facebook.com, but also for *.fbsbx.com and facebooks own content delivery network *.fbcdn.net as mentioned in the Subject Alternate Name-extension. It does also contain a public key from a Elliptic Curve Cryptosystem (ECC), which is shown in Subject Public Key Info. This is of particular interest, because Facebook also offers a certificate containing a public key from an RSA cryptosystem, if connecting with different cipher_suites.

In practice, the client evaluates the authenticity of an entity by verifying the certificate chain and determining the trust status of the Root CA's certificate. In the WebPKI, trust is established by storing CA certificates in trust stores, that can be used by applications and operating systems for automatic trust

[1] Last Access: 25.12.2014

evaluation. Browser and operating system vendors such as Microsoft or Mozilla usually manage these trust stores themselves to facilitate trust management and trust evaluation for endusers.

However, todays WebPKI has several issues, that directly affect the security of TLS connections.

2.1.5 Issues

A central issue of the WebPKI is, that an arbitrary CA is able to issue valid and trusted X.509 certificates for arbitrary domains on the internet. There is no alignment to the Domain Name System, which could, for example, limit the issuance for state-driven CAs to country code top-level domains, such as *.de*, *.us* or *.fr*. For example, a CA operated by the government of the Netherlands is able to issue certificates for domains operated by US corporates [67]. The problem is compounded by the huge number of CAs in the WebPKI. Recent studies estimate the number of CAs to 1590 [30], although most internet users won't need to trust most of the CAs [20, 60]. That massive design issue leads to the scenario, in which only one faulty CA is able to break the whole system. That makes an arbitrary CA in the WebPKI a single point of failure.

This problem is also compounded by missing mechanisms to supervise the issuance of certificates by those CAs from the WebPKI. Vendors and endusers have to rely only on policies that are published within certificate practice statements, certificate policies, subscriber agreements and relying party agreements. However, there is no way to verify the compliance with those standards independently. This leads to the issue, that a CA has no incentives at all to fit to their own standards. As real world examples like the DigiNotar hack show, CAs moreover conceal information about security-related incidents. DigiNotar is a dutch CA, which was hacked in July 2011 by Iranian hackers, who issued 531 certificates including certificates for google.com [62]. Due to existing security mechanisms in the Google Chrome browser, fraudulent X.509 certificates have been detected in August 2011. The detection led to the removal of the Root CA certificates from popular browsers in September 2011.

Moreover, there are legal issues with the documents provided by the CAs [64]. Nearly every certificate practice statement contains a chapter about warranties and limitations. In this chapter, CAs deny any responsibility related to certificates issued by them. Moreover, the legitimacy of enduser agreements must be questioned, as most endusers don't give their assent. That also supports the problem, that a CA must not fear any legal threats, when it comes to breaches or violations of their own certificate practice statements and hence have no incentives at all, to comply to their statements.

More issues emerge from negative market implications. By 2013, 75% of the X.509 certificate market is controlled by four major CAs [50]. This leads to misaligned market incentives both for CAs and for server operators. There is currently a discussion in research, whether there is a "race to the bottom" in prices for TLS certificates. If their was a race to the bottom, this would result in negative security incentives and misaligned investments, which would have already been expressed in various incidents. A race to the bottom would also mean that server operators mostly choose cheap certificates over secure certificates[71]. The result would be cheap web authentication as there is also a low level of domain owner verification. Moreover, there is also a market for other products and services related to certificates [14]. In this market, customers may choose the CA, that offers the most valuable services to their certificate management, to issue X.509 certificates for their domain. Again, the security mechanisms of the CA are pushed into the background.

This market situation also results in issues regarding liability and conviction. In the case of DigiNotar, the Root CA certificate has been removed from the trust store of the web browser, because DigiNotar was only a small CA and the removal affected only few users. However, in case of an incident at one of the big CAs, a browser vendor is confronted with the problem, that it must remove the CA certificate, although there are a lot of endusers which rely on that certificate. If the vendor would remove the CA certificate from the trust store, the enduser would not able to connect to any server using certificates from that CA. That situation would threaten endusers more than CA owners [30], which leads to negative incentives for big market players.

Endusers also face different problems when connecting to servers, as CAs often issue X.509 certificates, that are not aligned to standards and best practices. In a study, up to 34% of all certificates of the top 1 million hosts on the internet were issued by non-trusted CAs, 82% didn't contain the requested domain name and around 20% were expired [71]. Previous studies have also observed missing standards, resulting in invalid certificate chains due to expired certificates or missing Root CA certificates [43, 10] which often goes hand in hand with compliance failures at small corporate or government-driven CAs [27]. This imposes a problem for endusers, that also occurs with self-signed certificates: An enduser is not able to evaluate trust himself, if presented with a prompt in case of a X.509 certificate error.

Especially when it comes to technical error messages, for example during a failed TLS session establishment, endusers often fail to make the right decisions. Multiple studies have examined user reactions of error messages and composed fundamental issues in the TLS system [68, 5]. This problem is supported by studies, which have determined, that nearly all certificate errors are false positives [40], which means, that none of the certificate errors occurred due to an attack, but due to misconfiguration on the server side. Moreover, this misbehavior makes user familiar with error messages, which leads to the issue, that most users are simply ignoring those messages. Herley et al also propose [40], that incentives for endusers have to be created to take error messages seriously.

Other issues related to the WebPKI can be found, when looking at content delivery networks (CDNs). Content delivery networks are operated by private institutions and offer functionality to cache contents near the endusers location. Therefore, CDNs are usually distributed worldwide. For example, a user from Europe will connect to a server in Europe, while a user from the US will connect to a server located in the US, although the real server is located in South Africa. This enables both fast access for endusers and traffic reduction for server operators. However, TLS impedes this mechanisms, because of the fundamental semantic conflict between the end-to-end security TLS offers and the man-in-the-middle position of a CDN operator[49]. From a server operators perspective, a CDN is similar to a man-in-the-middle attack. This results both in operational issues for CDNs as well as a complication of a rational trust decision from an enduser's view. In case of operational issues, CDNs use multiple mechanisms to provide their service, which lead to invalid certificates, private key sharing, neglected revocation of stale certificates or insecure back-end connections with the customers original website. From an enduser's view, the problem is much worse, as it does hinder security mechanisms to successfully evaluate a certificate, because typical artifacts of unknown certificates occur in benign contexts as well [8].

These issues are exploited by attackers around the world. However, some issues can only be exploited by experienced attackers or attackers with appropriate resources.

2.2 Security model

The threat model describes the security goals and threat actors of the man-in-the-middle attack against TLS, which is in focus of this thesis.

2.2.1 Man-in-the-middle attacks

The man-in-the-middle (MITM) attack describes an attack class, in which the attacker is positioned between the participants of a connection. The attacker is also called Dolev-Yao-attacker. This situation makes the attacker able to access the contents of the transmission, for example meta-data like IP addresses or contents like user entered data or results returned from the server. The connection is established by the client, who is either connected directly to the internet or possesses his own internal network to enable other devices to use the same internet connection. The packets are then transmitted by multiple routers (hops) to the servers network and to the target server itself.

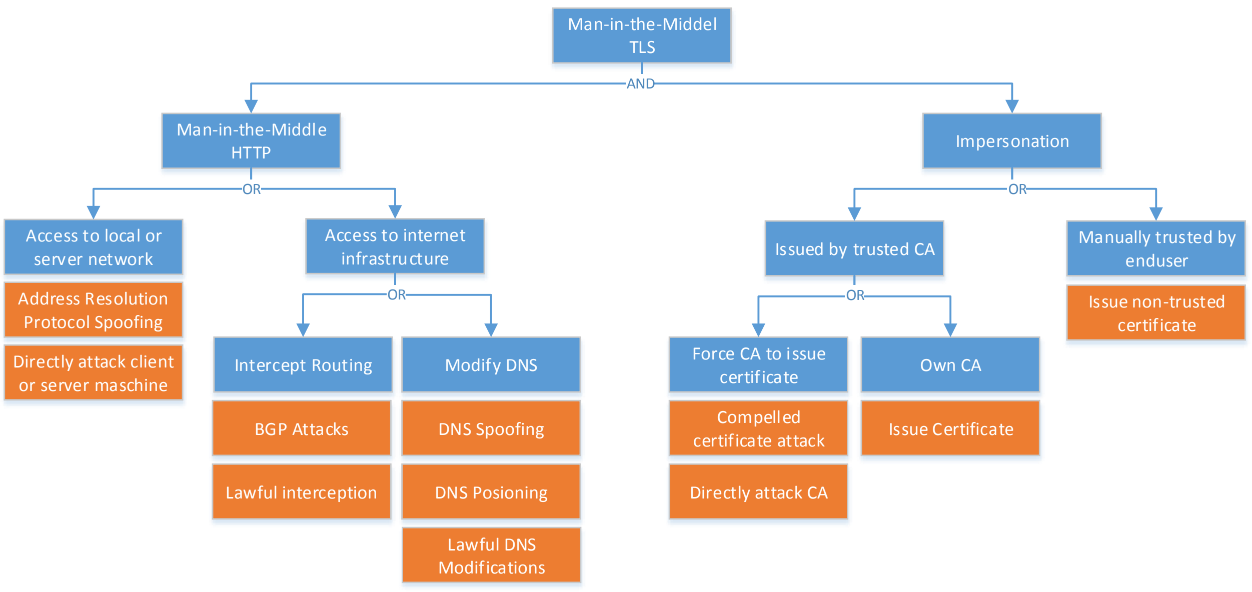
Man-in-the-Middel TLS
AND
Man-in-the-Middle HTTP
Impersonation
OR
Access to local or server network
Address Resolution Protocol Spoofing
Directly attack client or server maschine
Access to internet infrastructure
OR
Intercept Routing
BGP Attacks
Lawful interception
Modify DNS
DNS Spoofing
DNS Posioning
Lawful DNS Modifications
OR
Issued by trusted CA
OR
Force CA to issue certificate
Compelled certificate attack
Directly attack CA
Own CA
Issue Certificate
Manually trusted by enduser
Issue non-trusted certificate

Figure 2.6: Attack tree for a man-in-the-middle attack

Figure 2.6 shows the attack tree for a MITM attack against TLS (blue) together with practical attacks (orange). If TLS is not involved, an attacker only has to execute a man-in-the-middle attack against the unencrypted session without impersonating the server. This attack is shown on the left side of the tree. The goal of this attack is achieved by attacking either the server or client directly, or the infrastructure between both hosts:

- Within a local network, devices negotiate next hops using the address resolution protocol (ARP), which simply translates IP addresses to local switchable MAC addresses. The attacker has now the ability to change the clients or servers routing table by sending malicious ARP requests, e.g. it announces his MAC address for the routers IP address. This makes the receiver think, that the device acts as the default router. Every packet from the client to the target server and backwards is now transferred to the compromised device identified by the MAC address, which simply forwards it to the genuine router or the client respectively. However, the device is now able to read, alter, forge or redirect every packet.

- An attacker has access to the internet infrastructure, if he is able to change routing tables or if he is able to alter the responses the DNS servers return to the client. In the first case, an attacker can simply reroute the client to a router which is under control of the attacker. In the latter case, an attacker can spoof or alter the response of a DNS server, which causes the client to connect to a server under control of the attacker. In both cases, the attacker is able to transparently make the client believe, that he is talking to the legitimate target server.

If TLS is used between the client and the server, it is also necessary to impersonate the server. As shown in section 2.1.3, the X.509 certificate is central to TLS, as it both contains the public key and is signed by the certificate authority to be authentic. The impersonation attack is shown in the right subtree. The goal of the impersonation attack is to make the client believe, that the X.509 certificate of the attacker is a legitimate X.509 certificate for the server. An attacker has two possibilities to achieve this goal depending on his capabilities:

- He can depend on the naiveness of the enduser simply ignoring the certificate error message. In this case the attacker simply has to issue an arbitrary X.509 certificate, which will be used for the attack. For this attack, the attacker does not need any access to a trusted CA from the WebPKI.

- If the attacker is able to issue trusted certificates, no certificate error message is shown to the user and the attack will probably be unrecognized. For this attack, an attacker needs to have access to a trusted CA from the WebPKI, e.g. operated by himself, or he needs to force a CA to issue certificates without notifying the legitimate domain owner or server operator. This done either by forcing the CA to compel or by directly attacking the CA.

If TLS was implemented and set up correctly, the only way to initiate a man-in-the-middle attack is to exchange the certificate transmitted during the TLS handshake protocol, as sketched in figure 2.7. This thesis therefore concentrates on this type of attack and neglects implementation issues.

One can see, that the attacker basically impersonates the target server towards the client. In the background, the attacker also establishes a connection to the target server to forward the communication data from the client to server and vice verse. This is done for the client_hello and server_hello-messages. The certificate chain is exchanged by the attacker to include a server certificate, that contains a public key for which the attacker is in possession of the private key. This enables the attacker to encrypt any information sent by the client to the server. Because the attacker also establishes a connection to the server, he is also able to encrypt information, that he received from the server. Thus, the attacker is also able to impersonate the client towards the server.

As one can see in the attack tree, not every attacker is capable of each step of the attack. However, in todays internet several attackers could be found. Their goals, capabilities, limitations and motivations are described in the next section.

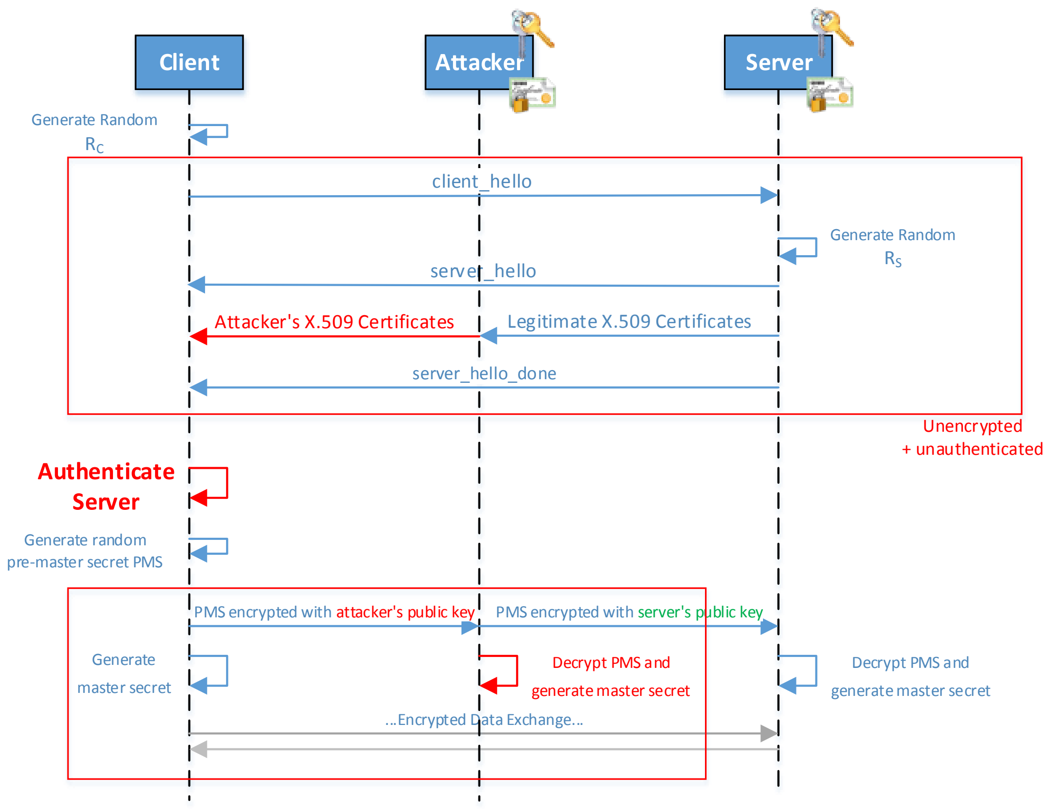
Client
Attacker
Server
Generate Random R_C
client_hello
Generate Random R_S
server_hello
Attacker's X.509 Certificates
Legitimate X.509 Certificates
server_hello_done
Unencrypted + unauthenticated
Authenticate Server
Generate random pre-master secret PMS
PMS encrypted with attacker's public key
PMS encrypted with server's public key
Generate master secret
Decrypt PMS and generate master secret
Decrypt PMS and generate master secret
...Encrypted Data Exchange...

Figure 2.7: Man-in-the-middle attack

2.2.2 Attacker-centric threat model

As the foundation for the treat model, the common Internet Threat Model is taken into account, which defines itself by the means of the following properties [63]:

- The network itself is not trustworthy. That means, that a router is possibly object to compromise and thus traffic can be read and modified without any detection on either the client or server side.
- The end systems are secure. That means, that an attacker is not able to access internal information, such as private keys or random number generators, from clients and servers. Also the attacker is not part of the internal network of the two actors.
- The attacker has huge computational abilities. That means, that an attacker is able to execute bruteforcing attacks and has the ability for big data analysis with previously collected information. This property is of special interest, as it limits the set of possible encryption- and signature algorithms in TLS. For example, the symmetric encryption algorithm RC4 is assumed to be broken, as one could calculate the plaintext in near realtime [6]. Thus, RC4 should also be prohibited in TLS according to RFC7465. In this thesis, it is assumed that an attacker is not able to break any of the cryptographic algorithms in use.

The internet threat model does not define any threat actors. However, the attack tree limits the number of possible threat actors. The main threat actor of this thesis is the government of an arbitrary state, as the government usually has both the capabilities and the incentives to mount such an attack. In 2013 and 2014, there have been leaked several internal documents from intelligence agencies by Edward Snowden and Wikileaks stating mass surveillance and attacks against businesses and individuals. Although nothing was confirmed from official side, there is no doubt, that governments must be seen as potential attackers [51]. These attacks are critical to the whole internet security, as governments are capable of complex, long-term and high-cost attacks against arbitrary targets ranging from military and enterprises to citizens. The goals of these attacks usually comprise of eavesdropping, information stealing, or identification of anonymous users. Attacks could be executed against submarine communication cables or satellite links. In some countries including the USA or Iran, also misuse of lawful interception of the internet backbone infrastructure like routers and switches at internet exchange points is possible. Moreover, countries could use juridical methods to deny publishing surveillance practices or notifying affected people.

In relation to the WebPKI, governments and intelligence agencies are also able to issue certificates using various methods [66, 67]. As mentioned in section 2.1.5, every trusted CA is able to issue certificates for every domain on the internet. As such, it is easy for government-driven CAs to issue certificates for potential victims. As pointed out by Soghoian et al, there are several Root CA certificates from state-driven CAs included in typical browsers and operating systems. However, the inclusion is not state-dependent. That means, that german citizens also have Root CA certificates from Estonian state-driven CAs included in their clients. Another approach is to force private-driven, browser-trusted CAs to issue certificates without notifying the public. This attack is known as the compelled certificate creation attack [67]. The issued certificates can now be used in man-in-the-middle attacks.

Nevertheless, states are limited by their juridical and physical borders. They are neither able to access infrastructure outside their sovereign territory, nor are they able to force CAs residing in other countries to compel to their regulations. This limitation is of particular interest, when it comes to routing. By now, a best effort approach is present on the internet. This allows data to be routed through multiple countries if this is the fastest route within the limits of all internet service providers. This is also the case for packets, which otherwise could have been routed within the same country only. If some countries are excluded from that routing infrastructure or from routes directly, then man-in-the-middle attacks would not be possible by that country. The same limitation is given by the operation of DNS servers. A country is not able to legally alter DNS data, if the server is not operated in the country. However, if a country does not restrain to international regulations, direct attacks to foreign infrastructure is possible. This discussion is also lead by the term “Cyberwar”.

Besides states-driven attacks, there are also other attackers capable of man-in-the-middle attacks. Common reasons for certificate changes are, for example, company security barriers or security software like anti-virus [45]. Both entities are able to install own trusted CA certificates or deploy soft- and hardware for pervasive monitoring or man-in-the-middle attacks. Especially companies often reason surveillance mechanism with legal restrictions and incentives in determining illegal internal information disclosures. Huang et al. estimated, that 0.2% of real-world connections are substituted with unauthorized forged certificates. Other surveys support this thesis, estimating that one of 250 connections is tampered by TLS proxies [57].

Amman et al. conclude [8], that a trust decision is impossible if an analysis of the certificate chain is central to the evaluation. They propose the usage of transparency approaches like Certificate Transparency or monitoring services that look for specific properties in certificates. In the next section, the multi-path notaries approach will be presented, which implements monitoring capabilities to cope with the presented threat actors and attacks.

2.3 Multi-path notaries

A notary is a remote service, that acknowledges certificates that were received by clients. It thus works as a control mechanism for the use of certificates, that were issued by CAs. Notaries implement multiple mechanisms of which multi-path probing is one mechanism. During multi-path probing, the notary is requested to establish a connection and to return the received certificate or its digest back to the requester. Thus, an alternative route is set up which - at best - uses completely different routers as the original connection that is established directly by the client. The client is now able to recognize man-in-the-middle attacks by comparing both certificates or their digests, as shown in figure 2.8.

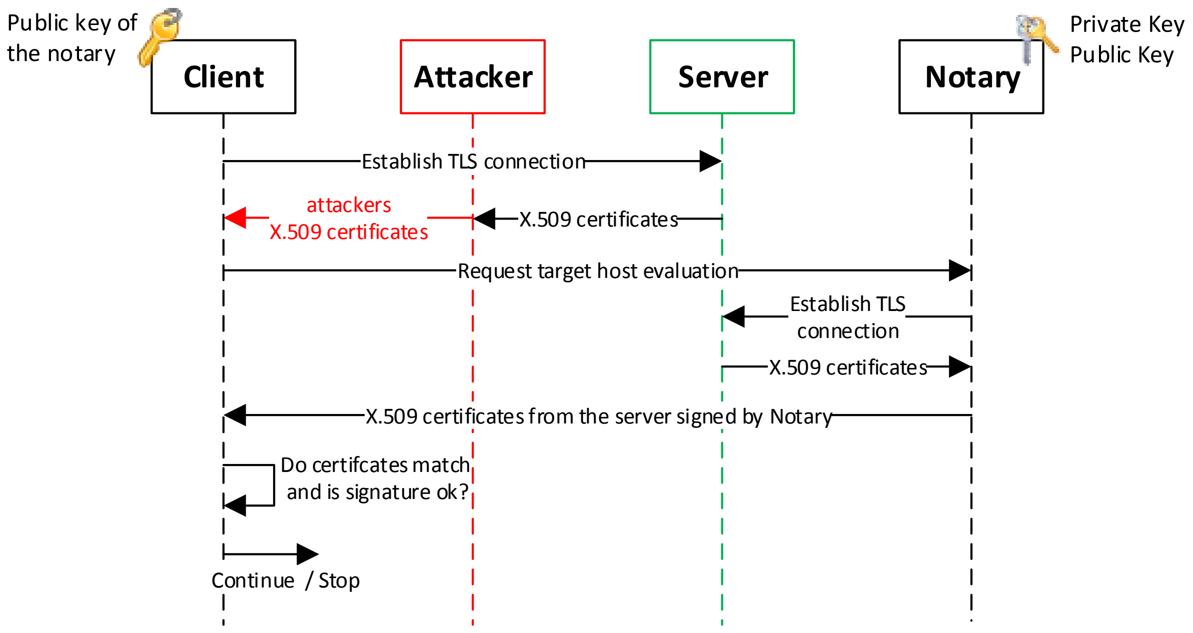

Figure 2.8: Target host evaluation

However, this is only the case, if the notary suffers the same attack as the client and thus is able to return the correct certificate to the client. In regard to the security model, this would imply at least, that the notary operator is not governed by the same law as the client. As a harder requirement, this could also imply, that the multi path notary is not governed by any suppressing law at all. To fulfill at least the first requirement, a widespread deployment is demanded. To fulfill the second requirement, a worldwide spread with at least one node in every country must be sought.

The first known implementation of this approach is done by the Perspectives project. Perspectives utilizes a network of multi-path notaries to extend the Trust-on-first-use (TOFU) approach and to detect attackers [73]. The TOFU approach comprises of the caching of a certificate digest and comparing the digests, that are received from both the notaries and the target server with the digest from the cache. Perspectives is by now the only actively developed and maintained notary software, which has set the standards for other approaches developed after Perspectives. Both aspects are the reason why Perspectives is used in this thesis as a an example implementation for multi-path notaries.

2.3.1 Architecture

Perspectives comprises of a server/client architecture. The server is implemented in Python 2.7 and distributed as a zip-archive, which can be downloaded from the official homepage. At the first start, Perspectives generates a public and private keypair, which is used to sign the responses from the notary. Perspectives also creates an empty database, as no hosts have been requested yet. However, a small tool is added to the installation, which adds a list of arbitrary hosts to the notary. Perspectives then stores the digest of each certificate, that was received, in its database, together with the the current timestamp and the domain name and port, that were provided in the request. When another request for that domain/port combination is received, Perspectives looks up that entry and returns all known certificates with their respective timestamps. If the domain/port combination was not found in the database, Perspectives stores that request in another database. Processing of the requests must be manually started by the server operator. The Perspectives project proposes, to start the processing of the stored requests automatically by using cronjobs on a Unix system, such that no manual start is needed.

The client is implemented as an addon for the Mozilla Firefox webbrowser. In the client, the user has to maintain a list of notaries. The public key of each notary must be maintained manually by the user. Moreover, the client offers settings to configure the number of notaries, that are requested, when a TLS connection to a target server is established by the browser. The perspectives client then asks a random subset of the notaries for the digest of the certificate of a given host, which is identified by a domain name. It is expected, that every notary returns the digest of the certificate, that the client also received during connection establishment. Based upon a key-trust policy, which determines the ratio of notaries to return the same certificate, the client is able to identify a potential problem.

2.3.2 Current situation

Perspectives has the problem, that it is not widely deployed. There are only a handful of working nodes[2] making Perspectives practically irrelevant to internet security today. The problem is exacerbated by the architecture, as it depends on the users and server operators to be implemented manually. However, the implementation requires both expert knowledge, as users must be able understand, why it is important, that the public key of a notary is received over a secure connection, and an understanding of the problem Perspectives is trying to solve, which could overburden endusers. Perspectives also depends on the server operator for a widespread deployment. Again, expert knowledge is needed for implementation

[2] The public available list of nodes is maintained under `http://perspectives-project.org/notary-servers/`. Last access: 25.12.2014

and operation. Moreover, it does not give any benefits for server operators, but could induce both weaknesses and vulnerabilities as well as higher resource consumption, which often leads to higher operational costs.

However, Perspective is object to (ongoing) research. Bates et al. have evaluated Perspectives and its discontinued successor Convergence, whose additional features have also been integrated into Perspectives, in a real-world scenario, where all certificates are checked by notaries within a university network in theory [16]. They come to the result, that convergence would increase network traffic by 0.1% and would require as little as 108ms to validate a connection. Also, Convergence was able to handle up to 300 connections per second, which results in delays of up to 260 ms for an enduser. However, scalability of Perspectives was not evaluated in a global context yet.

2.4 Related work

Besides Perspectives, there are other approaches which try prevent man-in-the-middle attacks on TLS connections. The approaches can be divided into notaries, supervision approaches and certificate pinning approaches. All approaches are used by clients to acknowledge previously received certificates.

2.4.1 Notaries

Besides multi-path probing, notaries can embody multiple other approaches. The ICSI Notary [9] collects SSL/TLS certificates by passively monitoring external traffic on seven different operational network sites. This database is accessible via DNS. A client asks the notary for an “A”-resource record of a hostname, which is formed from the SHA1 hash of the certificate. The notary then returns either an IP address indicating the trustworthiness based upon the Mozilla Root Trust Store - 127.0.0.1 means, that the certificate is not trusted and 127.0.0.2 means, that the certificate is trusted - of the certificate or NXDOMAIN, if the certificate hash was not found in the database. The “TXT”-resource record for the same hostname gives further statistics, e.g. how often the certificate was seen in a given timespan.

DoubleCheck [7] establishes a second connection using the Tor anonymization network. The returned certificate is then compared to the certificate gathered by a direct connection. Thus, Doublecheck does only rely on the Tor infrastructure, but not upon a trusted third party. This not only prevents privacy concerns on the user side, but also enables a widespread distribution without involvement of server operators. DetecTor [33] builds upon Doublecheck. DetecTOR proposes the implementation by TLS libraries to enable application independent operation. Moreover, it states and gives hints to solve drawbacks resulting from the the usage of the TOR network, for example security issues by compromised exit nodes.

Convergence builds upon Perspectives and adds multiple features to it, as well as fixing various weaknesses [52]. In contrast to Perspectives, Convergence aims at replacing the current CA infrastructure by introducing the concept of “trust agility”: The enduser is being able to add, change or reject nodes from the list, thus putting more control in the hands of an enduser. Most changes from the by now unmaintained Convergence project have already entered the Perspective project, making Convergence obsolete today.

Crossbear [44] extends the multi-path probing approach by a localizing feature, which tends to find the location of the attacker. A client sends its received certificates to the central Crossbear server. Crossbear then uses both own sources and Convergence for certificate verification and calculates a score which is returned to the client. If Crossbear fails to compare certificates due to different certificate digests, it asks different entities called hunters to return a traceroute to the target host to obtain route differences. According to the score and different thresholds, the enduser is now able to decide for himself, whether he wants to connect to the target server.

SignatureCheck [19] presents a protocol, which enables clients to gather certificate thumbprints from target hosts using an implementing service in realtime. The client asks the notary to connect to a target server and return the certificate thumb. The protocol is secured by RSA signatures independent of the WebPKI to ensure integrity of the response. Nevertheless the user has to maintain the list and the corresponding RSA keys by himself.

2.4.2 Pinning

Pinning approaches are used to receive the certificate previously to or during the connection establishment and store the certificate for a specific timeframe specified by the server. In this period, the certificate is valid, if it is received. Nearly all current browsers implement certificate pinning by just hardcoding a part of the certificate chain within the browser configuration and updating this certificates together with software updates. This approach is not usable for a huge number of certificates, so that other approaches were developed.

The Public Key Pinning extension for HTTP (PKP) [34] adds a new header to the HTTP protocol instructing the client to cache the servers' cryptographic identity for a specific period of time. This identity also includes a subset of allowed CA for that server. A client is now able to determine the correctness of the certificate based on the cached information. PKP is used together with HTTP Strict Transport Security (HSTS) [41]. HSTS is an extension to the HTTP protocol to protect websites against downgrading attacks. Downgrading attacks describe a form of man-in-the-middle attacks, in which the attacker alters transmission data in such way, that the client is not able to use encrypted connections. By using HSTS the server ensures, that the client is only able to securely communicate with the server within a defined period of time set by the server.

DNS-based Authentication of Named Entities (DANE) [42] is a protocol, that uses the DNS system, allowing server operators to place a certificate or public key in the DNS entry of the domain which can be used by the client to validate the certificate received during connection establishment to the server. The certificate can either be an end entity certificate, a CA certificate to be used as the trust anchor, or a domain-issued certificate, which does not need to be validated according to the PKIX standard. DANE heavily relies on the DNS Security extension [13]. Without that extension, resource records of DANE would easily be exchangeable by an attacker. However, DNSSEC is not widely deployed, as it has many drawbacks not solved yet.

2.4.3 Supervision

Supervision approaches target the general issue, that certificate issuance is not observable by the public. That issue allows CAs to issue arbitrary certificates without notification of the legitimate domain owner. Those certificates could also be used during attacks. Certificate Transparency [48] presents the concept of public, verifiable, append-only log. This log contains all certificates issued by certificate authorities, which should facilitate third parties to detect incorrectly issued certificates and enables clients to check whether the received certificate is contained in the log file. Certificate Transparency does not introduce a trusted third party, as the service which maintains the log does not need to be trusted, because the log is cryptographically verifiable and every compromise would be detectable. Although there's already an IETF standard track, the approach was not widely adopted by CA operators by now. In 2013 the authors added certificates gathered from a scan of the internet to the log [47] to raise popularity.

3 NotaryCache

This chapter introduces Notary by using a top down approach. First the goals, which NotaryCache tries to achieve, are explained, and the solution is shortly sketched. Then the requirements, which are imposed on the approach, are stated. The existing notary approaches are then evaluated based on the requirements. This leads to a gap of requirements not implemented, yet, by any approach. These requirements are central and will be fulfilled by NotaryCache, which will be shown in the next sections.

3.1 Goals

The goal of this thesis is to give an approach to achieve a widespread deployment of notaries. To accomplish this goal, an abstraction from the actual notary implementation is created to remove impediments that exist in existing notaries. NotaryCache is designed to be used as a basis for future notaries, as it gives guidelines for developers and researches, which information are needed to fulfill the requirements of a multi-path notary. Moreover, it enables the exchange and interoperation of arbitrary multi-path notaries, independent of notary-specific functionality.

In the first step, it is necessary to analyze the impediments of existing notaries for a widespread deployment. To evaluate the issues, a list of requirements for the design and implementation of a notary is determined in existing literature. This list comprises of the following fields:

Basic Requirements Basic requirements describe functionality, interfaces and data structures, which are used to accurately process input data and and generate the needed output.

Ubiquitous Computing Since the presented approach should be implementable in a ubiquitous way, requirements from the research field ubiquitous computing are to be included in the analysis.

Socio-Technical Challenges Socio-Technical challenges refer to issues of applications to adequately communicate information to endusers. This issues usually lead to rejection and dissatisfaction with applications, and thus must be included in every application, which depends on enduser interaction.

Economic incentives Success for security software is not only based upon the degree of increase of security, but also on economic incentives. To successfully implement a notary, requirements from economic research of the WebPKI are compiled and presented.

The existing approaches are then analyzed in regard to this requirements. The yet inadequately implemented requirements form the foundation of NotaryCache. The goal of NotaryCache is to accurately fulfill all requirements to enable a ubiquitous deployment.

3.2 Overview

NotaryCache introduces a layer of abstraction to implement arbitrary multi-path probing notaries in a ubiquitous way. NotaryCache extends a notary by a public available cache containing information about recently requested target hosts such as hostname, ip addresses or certificates received. The cache entries are added and deleted according to a specific caching strategy, which should enable notary operators to implement various business cases and thus gives incentives for implementation. Moreover, it enriches these entries with service information about that hosts to enable clients to implement automatic service discovery and bootstrapping without any user interaction. This not only prevents manual configuration of client software, but also enables notary operators to announce their service without any registration procedures in a central service. Moreover, a selection algorithm will be sketched based on country codes of the target host, which is used by clients to determine a list of notaries to request the certificate from according to the the security model.

NotaryCache is designed to be implemented as an extension to an arbitrary notary. To give a more concrete example, this thesis sketches an implementation which is compatible to the Perspectives notary. For evaluation reasons, our implementation does also implement real-time multi-path probing, instead of rescheduling of the request done by Perspectives.´

3.3 Requirements

An accurate requirement analysis is important, as this reveals the current misconceptions of multi-path notaries, which need to be solved by the herein presented approach.

3.3.1 Basic requirements & technical challenges

The basic requirements specify the basic functionality of a notary.

The specification of the input for a multi-path notary is deduced from the business logic of a notary: A notary has to connect to a given target host in order to gather the certificate of the TLS session. To establish a connection, the notary needs to know the network information, which clearly identifies the host. Within a TCP/IP network, such as the internet, the IP address and a TCP port number must be stated in the request. Moreover, the name of the host, given by the domain name, must be known to be used with the SNI extension of TLS to clearly identify the virtual host and to gather the correct certificate instead of only the default one.

> **Requirement 1** (Input). The solution must consume network information, which clearly identifies the entity of interest, e.g. the target host.

Requirement 2 specifies the output of a multi-path notary. Again, this requirement arises from the business logic of a notary: The notary has to return the complete certificate chain it has received during connection establishment to the target server. In addition to the certificate chain and to enable the client to check the response for consistency with the request, the notary has to return the ip address of the host and - in the optimal case - the route, which was used for connection establishment.

Requirement 2 (Output)**.** The solution must return the digests of the certificate chain offered by the target host and the connection details.

Laurie [48] states, that a solution for the issues of the WebPKI must not push decisions onto an enduser. This is supported by multiple studies, which show up, that users make wrong decisions when it comes to security. This implies the requirement, that a solution must be able to automatically evaluate trust without relying on enduser decisions. This requirement also raises the problem, that the user is not able to configure and maintain the implementation of the approach.

Requirement 3 (User Interaction)**.** The solution must be able to make a convincing trust decision without enduser interaction.

Compliance with the security model is essential, as it forces a precise definition of what is an attack and who is the attacker. It also implies an investigation of the actual situation to recognize possible pitfalls and false positives or true negatives and thus provides a flawless user experience, which is essential for a widespread deployment. Grant et al [38] have also titled this requirement as “response quality”, which means the accuracy of the response that is basis for the reactions the enduser will undertake. This also involves suitable strategies for target host evaluation based upon different attributes of the target host, like country or AS.

Requirement 4 (Adequacy)**.** The solution must recognize attacks in regard to the attacker and threat model.

Requirement 3 and 4 also lead to the reaction, that if an an attack could be identified free of doubt, then the result must be a hard fail, which must not be bypassed by the user [4]. In contrast, a soft fail would allow the enduser to accept the risk and to continue with the connection establishment. Therefore it is absolutely necessary, that the attacker model is defined accurately by the approach and that the mechanisms to identify the approach are suitable for recognizing these attacks.

As the notary itself could be target to man-in-the-middle attacks, it must ensure, that the responses are not alterable or that alteration is recognizable. Otherwise, attackers could simply spoof the response. This is of particular interest, as the notaries must be chosen wisely by the client (also see requirement 4), in order to calculate a meaningful result.

Requirement 5 (Security)**.** The solution must provide integrity and authenticity of the responses.

The goal of the herein presented approach is to increase the pervasiveness of existing multi-path probing approaches. This makes it also necessary to deal with a high number of requests to multi-path services. Bates et al. [16] have already evaluated Perspectives/Convergence based on this requirement. Grant et al. [38] have also investigated this requirement in their evaluation.

Requirement 6 (Scalability)**.** The solution must scale.

A ubiquitous deployment would also impose new risks. As multi-path notaries establish connections to other hosts, they could possible be used within distributed denial of service attacks (DDoS). In DDoS attacks, a large number of hosts is ordered to connect to a target host, which leads to overloading and unresponsiveness of the target host. There are two cases to examine, when considering distributed denial of service: In the first case, notaries are used to connect to target hosts in real time. In this case, the traffic, which is generated by the notary, must be higher than the traffic generated by the client when connecting directly to the target server. As HTTP is only used in conjunction with TCP today, a client has to do a TCP three way handshake first and send the actual HTTP request to the notary afterwards. The notary itself just opens the TLS connection to the target server, which generates less traffic than the connection from the client to the target server. The second case is, that notaries connect to target hosts asynchronously to user requests. If the attacker is able to predict the moment of connection establishment from the notary to the target server, then the attacker is able to multiply the traffic generated at the target host at a specific point in time. This is done by requesting the evaluation of a target host from a notary. In the moment of evaluation, the client will also connect to the target host. The target server has now to serve two connections in parallel. If the attack is executed with more than one notary, an attacker could achieve a high traffic load on the server. This leads to the requirement, that the moment of connection establishment must not be predictable.

> **Requirement 7** (Unpredictably of connection establishments)**.** The solution must consider random moments when gathering TLS certificates asynchronously.

Notaries can also be misused for port scanning attacks. From an attackers perspective, a port can serve both a TLS-based protocol, an unencrypted protocol or it is closed. However, it is not important for a notary to distinguish between a closed port and an unencrypted protocol. This information must therefore not be disclosed to a client. Moreover, notaries could expose information, which could give attackers an idea of how to attack a target server, without ever connecting to the target server. This leads to the following requirement:

> **Requirement 8** (Information Sensitivity)**.** The solution must not expose security-sensitive information to the client.

DDoS could also be executed against notaries, which could lead to defect notaries. This must not hinder the client to securely connect to the target host or to be victim of an attack, because the notary failed. Thus, the notary must not act as a single point of failure [38].

> **Requirement 9** (Reachability)**.** The solution must give mechanisms to ensure operation, even if the notary is not reachable by the client.

Notaries must not impose additional risks on server operators, as this could lead to missing incentives for implementation. Especially if the notary is operated besides existing services, it must not affect these services, for example to suffer from performance derogation. Other externalities for server operators comprise of exposing of internal information due to missing security, lower reputation due to service disruption or legal externalities because of broken SLAs [1].

> **Requirement 10** (Risks)**.** (Risks) The solution must not impose additional risks on server operators.

[1] Service Level Agreements - Legally forced requirements on service operation

Technical challenges must also be seen in regard to user interaction. From a technical point of view, the field of ubiquitous computing presents various requirements to developers and researchers regarding environmental interaction and enduser involvement, which must be considered, if it comes to ubiquity of a solution.

3.3.2 Ubiquitous computing

Ubiquitous hereby describes the property of a computer system to be both existing and invisible to the enduser[72]. This property is important for many security measures and products, although it was not often looked at in the past. The research field of ubiquitous computing defines multiple requirements for applications to be used in a "ubiquitous" way. This section will state the main requirements from literature, which support the goal of successfully getting implemented and accepted by endusers.

The first requirement to be discussed is adaptability. Adaptability means, that the application or the system handles user preferences and user context dynamically [55]. Adaptability differs two extreme approaches for implementation: The laissez-faire approach, in which the application is responsible for adaptions, and the application-transparent approach, in which the system is responsible for the adaption. Adaptability requires the ability of the application to be aware of the context, in which it is used.

> **Requirement 11** (Context-awareness)**.** The solution must gather and process information about the context, in which it is used.

In this thesis, context is defined as the users attitude towards the following items:

Risk As shown in multiple surveys, most certificate warnings result from invalid certificates and not from man-in-the-middle attacks. That makes it possible to distinguish between often occurring low-risk warnings, and rarely occurring high-risk warnings. Browsers and their security measures have to react upon each category in a different way [4].

Location The physical location of the user, the notary, target server and the server operator must be included in the decision whether to connect using the given certificate to prevent legally justified man-in-the-middle attacks [67].

Hardware and system usage Hardware component properties like disk usage or latency must be taken into account. For example, if disk space is low, the solution must not be allowed to increase disk usage. When it comes to network latency, the application has to evaluate the free bandwidth, it is able to use.

Adaptability is also described as agility of an application to be sensitive to different properties of the system, such as bandwidth or cpu usage, and to changes in resource availability in case of data sharing. In case of bandwidth, this requirement also relates to requirement 17, as a high latency could disturb and discourage the user. So, adaptability must be given to conquer the problem of high latency in case of high bandwidth. In general, adaptability is also described as context management [25].

Requirement 12 (Adaptability)**.** The solution must include user behavior and must react upon varying system constraints.

Adaptability can be extended by self-organization. Self-organization describes the ability of the application not only to adapt their behavior, but to also learn from that adaption, which leads to concepts like expert systems, chaotic theory and fuzzy logic. From a software perspective, this leads to dynamic software architectures [55].

Another requirement from the field of ubiquitous computing is standardization. Standardization comprises of standardized interfaces accessible by standardized URIs [2], such that every application can interact with arbitrary other applications. This also enables system discovery and dynamic bootstrapping without using 3rd party services. In the context of this thesis, standardization does also enable different approaches to be used together, which both broadens deployment and allows the development of different approaches "under one hood".

Requirement 13 (Standardization)**.** The solution must be accessible by a standardized URI.

Moving forward, ubiquitous computing not only examines technical challenges, but must also be seen in context of (end-)user interaction. As endusers and server operators play an essential role in the implementation and usage of such security systems, their interests must be looked at not only from a technical, but also from an social perspective.

3.3.3 Socio-technical challenges and success factors

Socio-technical challenges result from a system, which consists of both technical and social components [69]. As endusers and their trust relationships to other actors are part of the presented system model, socio-technical aspects must be considered as important drivers for acceptance. It is also by now widely acknowledged, that solution approaches, that adopt to socio-technical challenges during system development, are more acceptable to endusers and deliver a better value to stakeholders [59, 18]. In the context of this thesis, these challenges mainly result from the participation in the WebPKI and the interaction with various actors, such as browser vendors and CAs. This interaction demands trust in these actors, which, on the one hand, leads directly to privacy issues for endusers, because sensitive personal data is collected, stored, processed and communicated, as for example discussed by Geihs et al [36].

Requirement 14 (Privacy)**.** The solution must consider privacy as a legitimate property for endusers.

On the other hand, interaction always requires usability. Usability is needed to manage trust and trust decisions. Usability extends the engineering approach from socio-technical challenges with methods to achieve user satisfaction by including users abilities and needs [54]. In contrast, socio-technical systems theory only aims to supersede concerns with effectiveness and efficiency alone. There are two main trends, which directly relate usability aspects information system success: User satisfaction and user acceptance. While user satisfaction must not necessarily be given by a security measure, user acceptance of a security measure is highly important to application vendors. As such, acceptance must not only be

[2] Uniform Resource Identifier

considered for the herein presented approach. The externalities, which result from implementation of our approach, must also not decline acceptance in the core product. This requirement is essential, as it prohibits changes to the core application, which are not in the interest of the application developers.

A widely applied model to describe user acceptance is given by the Technology Acceptance Model (TAM) [26]. The model relates the beliefs about the perceived ease of use and the perceived usefulness to the behavioral attitude towards usage. According to this model, the presented approach must fulfill the following requirements

> **Requirement 15** (Ease of use)**.** The solution must not reduce the perceived ease of use with the core product.

> **Requirement 16** (Usefulness)**.** The solution must not reduce the perceived usefulness with the core product.

While requirement 15 is related to requirement 3, requirement 16 demands further investigation about the intention to use the core product [76], which leads to direct requirements for the approach. In this case, the core product consists of a web browser, which is used to access websites from third parties. From a users perspective, web browsing must not be interfered by security measures. As shown in [68, 40], any false-positive interference would lead to refusal of the security measure. As such, both requirement 15 and 16 directly relate to requirement 3, but add refinements concerning user interaction.

Both requirements and their realization can be observed in todays browsers by looking at already existing security measures as defined by the WebPKI standards. Revocation mechanisms, such as certificate revocation lists (CRLs) [24] and the online certificate status protocol (OCSP) [65], should provide clients with revocation information about certificates. This can be useful, for example, if the private key was disclosed and the existing certificate must be made invalid. However, both CRLs and simple OCSP are disabled in most browsers, because online revocation checks and downloads of CRLs slow down connection establishment and compromise privacy [46], which both is not in the interest of the application vendors, as it annoys users and lets them switch to competitors. So, besides privacy, latency is a strong requirement, that was also included in other analysis [38].

> **Requirement 17** (Latency)**.** The solution must not increase latency to a level, which disturbs the enduser.

Usability can further be extended to the effort to install and remove the solution and the effort to maintain the approach for both the users and the server operators. Too much effort - this also includes expert knowledge and the process of acquiring that knowledge - could discourage users and server operators to implement the approach. However, if the approach is implemented transparently to the enduser, such that no user interaction is necessary, this requirement would perfectly be fulfilled.

> **Requirement 18** (Effort & Expert Knowledge)**.** The solution must be easy to install, maintain and remove without expert knowledge.

However, these requirements are not only limited to endusers, but must be extended to server operators. Server operators have the fundamental interest, that their servers are not harmed by any other software they operate. According to TAM, ease of use and usefulness of the product must also be seen as a concern of the server operators. Nonetheless, most server operators have to pay for the hardware, the network or other services besides their interest in making money with their core business, thus expecting a benefit from the operation. From a server operator point of view the socio-technical challenges manifest themselves in economic incentives.

3.3.4 Economic incentives

Ross Anderson showed [11], that information security is not only of technical concern, but must also be seen due to perverse incentives. He explained these perverse incentives with rational business decisions, when it comes to market actions or investments. He identified perverse incentives like

- Asymmetric information: If customers could not differ between the quality of different products, then they are likely to choose the cheeper, low quality product. This principle is also described as the market of lemons [3].
- Missing security expenses: The goal of a company is to be successful in the market. The faster the company enters the market, the faster it could be aligned to the competitors and the faster it could be successful. This leads to an underestimation of security measures and to low or none security investments.
- Dumping the costs/risks to the users: Another perverse incentive of companies is to dump liability and risks to other entities, instead of avoid the risk by implementing own security measures. However, this is a rational decision from a companies perspective, as it is much cheaper and more effective than mitigating the risk by oneself.
- Buying security products only from market leaders: Large product vendors are often seen as highly responsible entities able to cope with security. This enables managers, on the one hand, to better justify expenses and, on the other hand, to be less liable when it comes to security breaches. This aspect is also called “liability shield”.

When it comes to incentives to implement security measures, companies rather invest in security technologies, which discriminate market competitors and support the own market position, then strengthening the product security. This can be seen for example, when looking at Digital Rights Management or Trusted Computing [12]. Both systems are used to restrict the users freedom, for example, by prohibiting the installation of other operating systems, instead of securing the original operating system. The same can be seen, when looking at printer vendors, which authenticate not the customer, but the cartridges, to restrict the user to cartridges of the same vendor.

Vratonjic et al. [71] have also found this incentives in the field of the WebPKI. Besides other weaknesses of the WebPKI, the authors assume, that buyers can’t meaningfully distinguish between secure and insecure certificates and that there is a huge information asymmetry, hence leading to a race to the bottom in prices for web certificates. However, Asghari et al. [14] have investigated data from the SSL Observatory[3], which suggests, that the market for certificates is highly commoditized and builds

[3] https://www.eff.org/de/observatory

around four major certificate authorities. They explain, that observation with high entry barriers to the market, since root certificates have to be trusted by browser vendors, which would examine every certificate authority. Asghari et al. moreover have spotted a market for additional services, which are sold at dramatically different prices. They also examined customer decisions to buy certificates only from large certificate authorities, because of certificate management and security services or security signals towards endusers, which should suggest, that the website is secure (site seals, high prices). Also a large certificate authority would work as a liability shield and the root certificate is less likely to be removed from the browser's root store.

When it comes to requirements for a solution, one has to consider these economic incentives. One simple conclusion for certificate authorities to implement the approach might be, that the approach must be usable within business cases, to attract potential customers and to create benefits as compared to competitors. Simultaneously, the approach must avoid disincentives, like raising costs for traffic or reorganizations.

> **Requirement 19** (CA Competition)**.** The solution must raise chances for customer engagement over the competition, e.g. give business cases.

When looking at server operators, a conclusion could be to simplify and strengthen certificate management and strengthen liability. It could also be applied as a signal to endusers, that security really matters. However, as discussion shows, the user is not directly interested in security, but more in his own business goals.

> **Requirement 20** (Server Operator Benefits)**.** The solution must be usable within certificate management procedures and liability evaluation, for example to lower costs of insurance contracts.

3.4 Gap analysis

In the following section, a gap analysis of existing approaches is done to determine the impediments of existing notaries to achieve a widespread deployment. Each approach - Perspectives, Convergence, Crossbear and SignatureCheck - is therefore analyzed regarding each requirement. The insights, that result from this analysis, are then used to shape the herein presented approach.

Requirement 1 Perspectives and Convergence inadequately implement requirement 1. Their notary implementations only consume the domain name and port of the target host, which are then used to establish the connection from the notary to the target. If the domain name resolves to more than one IP address, then this leads to the problem, that the connection may be established with other IPs instead of the IP, which is used by the client. In contrast, SignatureCheck implements requirement 1 by enabling the client to state the IP in the input data instead of an hostname, which lead to the problem, that, if the IP is controlled by an attacker, the notary does return the correct digest. In case of a DNS related attack, in which the domain resolves to an rouge IP address, the client would not be able to recognize the attacker due to the (correct) response from SignatureCheck. Crossbear does also consume an IP address, but does also require the hostname as an input variable. However, Crossbear does not use the provided IP address internally to connect to the target host[4]. Crossbear moreover consumes a digest of the received certificates to be able to start its “hunting” in case the certificates mismatch.

Requirement 2 Perspectives and Convergence implement requirement 2 by returning all certificate digests ever recorded for that domain name. However, there is no information given regarding IPs or ports. In combination with their implementation of requirement 1, they do not enable the client to moderately evaluate the result. Crossbear does neither return the certificate, nor the digest. Instead it returns an internal evaluation and an score based upon the provided information and the connection details received during connection establishment with the target host. This evaluation also contains an indicator, whether the provided certificate matches with the received certificate. However, together with requirement 1, Crossbear would also need to return the IP address it used to connect to. In contrast, SignatureCheck returns the IP address in its response, if it was stated in the request. But if the requests only contains a domain name, SignatureCheck only returns the domain name instead of the IP address used during connection establishment. This leads to the same problem as stated for Perspectives and Convergence.

Requirement 3 Besides SignatureCheck, none of the implementations fully realize requirement 3. SignatureCheck does not need any configuration, but is also not able to request details from more than one notary. In the other notary implementations, the client must always be configured manually by the enduser, for example a list of notary servers has to be maintained or the user has to configure his own security level. The latter is important, as it is assumed [74], that the user is not able to evaluate trust in a technical context. However, if the add-on is properly configured, then every implementation is able to fulfill requirement 3.

Requirement 4 Only Mozilla Firefox allows add-ons to access received certificates, while Google Chrome denies access to received certificates for unknown reason [37]. That leads to the problem, that compelled certificate creation attacks are not to be detected, if notary approaches are only implemented as browser add-ons for Google Chrome. That creates the situation, that all approaches are only implemented for Mozilla Firefox. Unfortunately, none of the presented approaches is implemented in a way other than as a browser add-on. Moreover, all approaches produce false-positives - a correct certificate is evaluated as fraud - and true-negatives - a fraud certificate is not detected at all. This is a direct result from the insufficient implementation of requirement 1. As shown above, this leads to the problem, that users become accustomed to simply acknowledging error messages, which reduces security and interest in the approach.

Requirement 5 All notary implementations fulfill requirement 5 by implementing cryptographic signatures over the response. However, there is no automatic and secure key distribution, so the public keys to verify the signatures must be maintained manually by server operators and endusers, which is error-prone and lowers security, if an enduser is not able to verify public keys properly.

Requirement 6 Perspectives, Convergence and Crossbear use static responses, which makes it applicable in content delivery networks [73]. When using SignatureCheck, one is not able to make use of caching, as the protocol does contain nonces. Also, monitoring and updating the key data from

[4] See https://github.com/crossbear/Crossbear/; Last access: 09.03.2015

Crossbear, Perspectives and Convergence is parallelizable, which enables high-load perspective nodes to be run on multi-core hardware.

Requirement 7, 8, 9, 10 In SignatureCheck, Convergence and Crossbear connections are established in real-time to a target server. Perspectives establishes its connections asynchronously and not in real-time. They are therefore only susceptible to be used in port scanning attacks. As opposed to this, Perspectives uses an external executable to process previously collected requests. This executable must be called manually by the server operator. The developers recommend implementing a cron-job for this task, whose configuration can be figured out by a capable attacker. There is no random execution on a per host basis, which makes it vulnerable to DDoS attacks.

When it comes to port scanning, Crossbear enables an attacker to misuse the notary to evaluate the security-level of the target server. The lower the score calculated by Crossbear, the more is the host likely to be vulnerable to an impersonation attack. So attackers could simply use Crossbear to create a list of possibly vulnerable hosts. Perspectives, Convergence and SignatureCheck do not reveal any information about a port except the digest of the certificate.

Per default, Perspectives and SignatureCheck only offer their services over unencrypted HTTP connections. This not only makes them prone to man-in-the-middle attacks and directly leads to reputation losses, but could also disrupt policies like PCIDSS[5]. For some companies this could also lead to legal issues. In contrast, Crossbear and Convergence both use hardcoded, self-signed certificates, which also makes them prone to attacks, if system operators are not well trained to maintain these implementations. Also the current development status raises risks, e.g. if a vulnerability is recognized.

Moreover, none of the implementations offers protection against DDoS attacks. If a capable attacker is blocking the access to any of the notaries, an enduser is not able to evaluate received certificates anymore. Since all notaries implement an default interface, which is easily detectable, blocking is not that difficult for threat actors like governments, who are in control over the internet infrastructure.

Requirement 11, 12 None of the notaries include context information like system/network performance or the location of the enduser or the notaries in their operation. This not only affects the browsing experience of the enduser, e.g. when network load is high, but it also makes the notary prone to security issues, as presented in the threat model. Crossbear and Convergence at least adapt to the endusers risk level, by defining an increasing number of notaries per risk level, from which the certificate is going to be requested. However, this must be done manually by the enduser and is not changed automatically or on a per target server level.

The is same situation is valid for the server component of a notary. None of the notaries gives a server operator the possibility to configure limits for system or network usage. This leads to the issue, that a notary could use all system resources without including other applications, which are potentially more important.

[5] Payment Card Industry Data Security Standard - A standard mandatory for companies, which offer payment services to customers.

Requirement 13 None of the notaries makes use of standardized interfaces and protocols, nor are there any plans to create a standard, for example in the IETF[6] or other organizations. Every developer defines his own interfaces and protocols, whereas other approaches are not in scope. Besides the issue, that this situation leads to a high fragmentation, it also complicates further development of add-ons, integrations in other applications or even the integration within the web browser itself without the help of add-ons.

Requirement 14 Neither Perspectives, nor Crossbear or SignatureCheck take privacy into account. These notary implementations are able to track endusers using the IP address or browser fingerprints and analyze their surfing behavior, which leads to massive privacy concerns. Only Convergence takes endusers privacy into account by implementing a technique called notary bouncing. During notary bouncing, Convergence requests the certificate from other Convergence notaries without mentioning the user. However, the original Convergence notary is able to read all parameters and is such able to build profiles of endusers, although it does not directly establish a connection to the target host.

Requirement 15 From an endusers perspective, every notary makes the situation to deal with certificate errors much more difficult to understand. Not only must an enduser be able to evaluate the responses from the notaries, which are for example present in error messages from SignatureCheck and Perspectives. He must also able to correctly configure the add-on to gain the full benefits. However, this makes usage of the core product - the webbrowser - much more difficult in terms of maintenance, configuration and simple web browsing.

Requirement 16 Perspectives decreases the perceived usefulness of the webbrowser, by showing error messages not only in error cases, but also, when notaries could not be reached. This disturbs the enduser, because there is no reason indicated, why an error message should be shown. Convergence removes this message and only operates, if the certificate could not be validated by the browser. SignatureCheck does not implement any error messages at all. SignatureCheck just adds a small icon to indicate the correctness of the certificate. The Crossbear addon is outdated and can't be installed by April 2015.

Requirement 17 All implementations need at least one request to a notary to evaluate the certificate, whereas in most cases one notary is not enough. Perspectives, Convergence and Crossbear therefore let the enduser define an risk-level he is willing to cope with. This defines the minimum number of notaries to request the certificate from. As shown, this is not always the best way, especially if the enduser is not an expert and can't define his own risk level. But raising security requirements also negatively affect latency, as every additional request slows down the connection. To cope with latency issues, all notaries except SignatureCheck have implemented caching mechanisms to prevent connection establishment to hosts, which were requested shortly.

Requirement 18 Although installation and removal is easy, because every implementation is available as a simple browser add-on, maintenance of Crossbear, Perspectives and Convergence is very so-

[6] Internet Engineering Task Force - The organization run by the Internet Society, which creates and manages standards for protocols and architectures used on the internet

phisticated for a non-expert enduser. Endusers have to configure extra notaries or take care of non-existing notaries. This also includes maintenance of cryptographic key material and risk levels, of which endusers are often not aware of and where normal endusers are not capable of. This makes usage of this add-ons not reasonable to endusers. In contrast, SignatureCheck doesn't provide any configuration, but also doesn't offer any possibilities to configure additional notaries. Also, none of the notaries implement mechanisms for automatic configuration, bootstrapping or service discovery.

From a server operators view, installation and maintenance is much more complicated. As shown in the evaluation of requirement 10, Perspectives doesn't offer any encryption, not does it offer any limitations on system or network resources. The latter is also true for other notary implementations, which forces the system operator to implement this feature on his own. Another problem is, that only Perspectives is actively maintained, leaving the solving of issues to the system operator. Also, there are no predefined packets for usual Linux distributions or Windows operating systems, which makes the situation for amateur system operators much more difficult.

Requirement 19 None of the approaches provide any perceivable incentives to CAs for implementation of the respective approach. Each of the approaches would just produce costs at no benefit towards CAs, which do not implement the approach. Also, the service could hardly be sold to customers, as only browser addons are offered.

Requirement 20 Perspectives and Convergence could be used to gain benefits besides enduser security. In the context of certificate management both notaries can be used to track the usage of certificates or the validity of the certificates in use, which could simplify certificate reissuance or optimization of current certificate usage. However, none of the approaches offers interfaces for 3rd party applications to feedback the data.

Conclusion

It was shown, that nearly none of the requirements were accurately fulfilled by any of the approaches. The evaluation revealed three central issue categories, that have evolved from this evaluation:

The first category is given by the technical issues. It comprises of inaccurately defined requirements for input and output data, misaligned strategies for the evaluation of a target host, and the the susceptibility against blocking and man-in-the-middle-attacks against those notaries. That not only leads to a high number of false positives, but moreover make current approaches ineffective from a technical point of view.

The second category describes the low integration of enduser requirements, hence making multi-path notaries unattractive to endusers, system operators and CA. As shown above, all notaries have different problems regarding service automation and context-sensitivity. Also configuration is a big issue, since most users are non-experts. Endusers and system operators must take notice of an implementation, which requires to be aware of the issues of the WebPKI, which can also not be expected from most endusers and even server operators. This limits the target audience massively to expert users. Other usability issues comprise of the handling of false positives, which leads to unexpected warning messages, failed connection establishment and, again, bugged endusers.

The third category is set by missing incentives for system operators, CAs or other participants of the system model to implement security services without any benefits for their own business. Yet, multi-path notaries do not provide any incentives to server operators to implement the software. They don't offer any benefits besides a "good feeling", that something is done for internet security. From a business perspective, they only impose disadvantages like increased costs for disk space and traffic, as well as an increasing risk by operating potentially insecure software.

3.5 Approach

As described in the last chapter, there are multiple issues with todays multi-path notary approaches. These issues lead to the current situation, that notaries are not widely deployed, making them irrelevant to internet security. To overcome these issues, a new layer of indirection is introduced, which adds mechanisms for automation of various operations by abstraction from the actual notary. To accomplish this abstraction, a new data structure is created, which is called "cache". This cache will be publicly available to any other participant of the WebPKI. Usually, a cache stores information to reduce three different criteria: the number of requests to a target server, the volume of network traffic and the latency on the endusers side [1]. In this approach, the cache is used for more than just reducing network load. It also specifies a format for entries making the cache interesting for bootstrapping and service discovery, hence making a client implementation of a notary independent from enduser maintenance. Simultaneously the cache creates new business cases and incentives for CAs and server operators to implement this approach. The abstraction moreover enables the cache to be run independently of the specific notary, as it only holds basic information added from the notary, but does not force any other changes to the notary besides the integration of the cache.

The following sections will introduce the architecture of the system and the structure of the cache. The cache implements a cache strategy, which is responsible for addition and removal of cache entries. At the end client processing, bootstrapping and decision-making are discussed and it will be presented, how a publicly available cache can help to simplify those operations.

3.5.1 Architecture

The concrete architecture is shown in figure 3.1. Each server in the architecture could act as a notary by implementing NotaryCache. Each notary in the architecture manages a cache file, which is available to all participants of the WebPKI. A cache consists of different entries, each representing exactly one target host, that was previously requested (*live request*) from that notary or that is added to the cache by the server operator. However, there can be multiple entries for one target host, for example, if a host delivers different public keys due to different *cipher_specs* or if a domain resolves to different IPs. The cache also includes a header containing information about the entity responsible for managing the cache and the validity of the cache. The notary regularly generates a private and public key pair, which is used to create a cryptographic signature over the cache and its header.

As already stated, one cache is managed by each notary. Management means, that the notary is responsible for adding and removing target hosts to or from the cache as well as providing the cache to the public. The server operator is assumed to be the only entity in the WebPKI capable of operating a

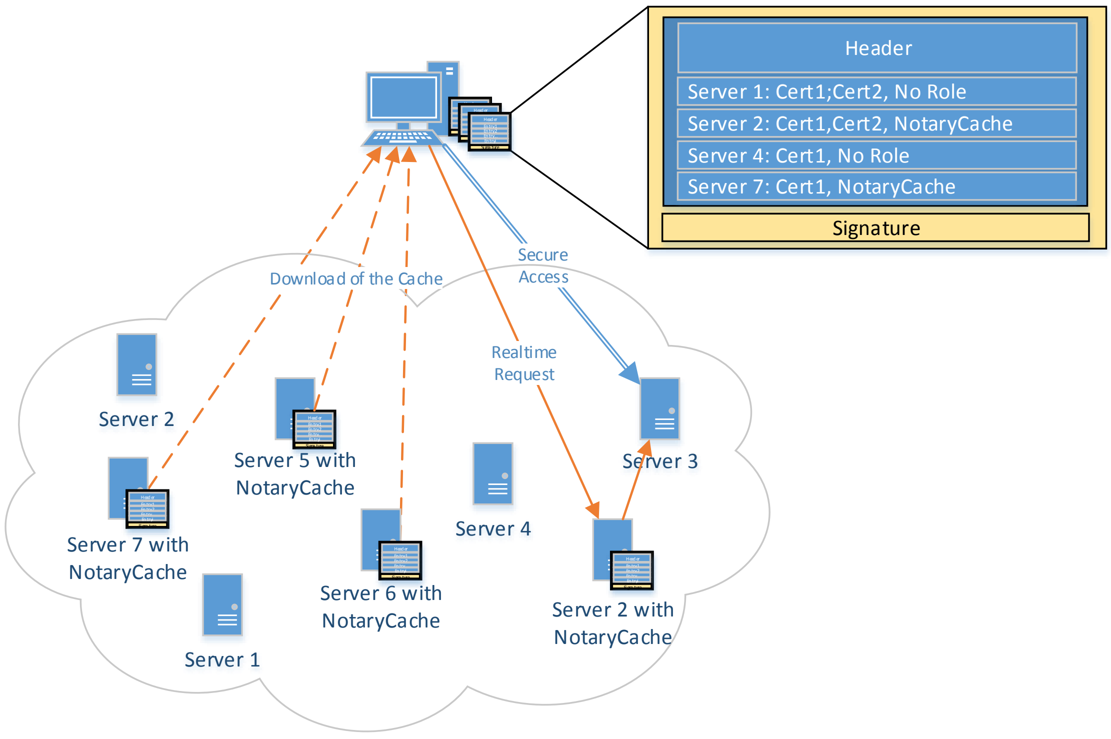
Header
Server 1: Cert1;Cert2, No Role
Server 2: Cert1,Cert2, NotaryCache
Server 4: Cert1, No Role
Server 7: Cert1, NotaryCache
Signature
Download of the Cache
Secure Access
Realtime Request
Server 2
Server 5 with NotaryCache
Server 7 with NotaryCache
Server 1
Server 6 with NotaryCache
Server 4
Server 3
Server 2 with NotaryCache

Figure 3.1: Concrete architecture

notary for several reasons: In contrast to a client, a server usually has a high uptime, making it available nearly 24/7. A server also usually has a high bandwidth and is able to serve multiple requests at a time. Also, server operators may have business incentives to operate additional software besides the core software for his businesses, which is not expectable from clients.

NotaryCache offers three basic interfaces:

- The configuration interface is used to access the public configuration of the notary to give the client the ability to use the cache in its defined context. This interface must only be accessible via TLS, as it provides the key material used by the client to verify cryptographic signatures to verify the integrity of the cache.
- The cache interface is used to download the current version of the cache. This interface does not need to be available via TLS and thus can be outsourced to content delivery networks or other service providers.
- The notary interface is used to request the certificate of a target host in real-time. Again, this interface must only be available via TLS to prevent altering of responses.

Similar to existing notaries, one could also think of a fourth interface, which is used to issue asynchronous requests, which will be processed at a random point in time. However, this interface is not needed for the herein presented concept and will therefore be neglected.

The client uses these interfaces during target host validation. Target host validation describes the process of validating the certificate of a target host by using the caches provided by the notaries. In contrast to existing notaries, this check does not have to be conducted online by requesting the certificate digest from the notary, but can be done offline by simply downloading a list of caches asynchronously or during a software update. The public configuration is used to gather the public key and other configuration items necessary for the client from the notary. It is a prerequisite for the communication, that the configuration is only accessible via TLS. The notary interface is used in addition to verify the public key in real-time, if this process seems necessary to the client, for example, if only a small number of caches effectively contain the target host.

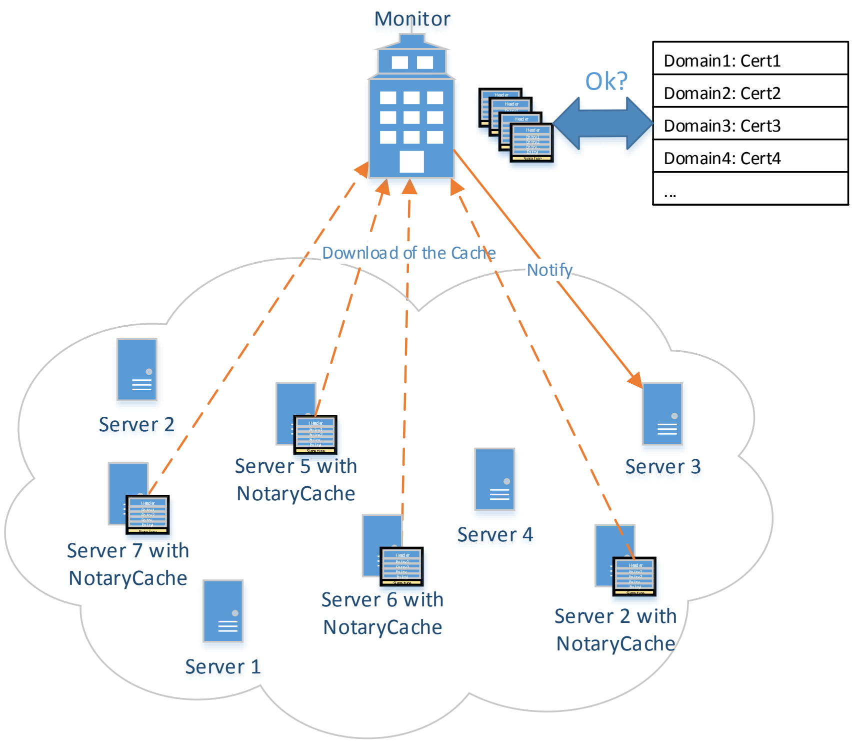

Figure 3.2: Cache monitoring

These interfaces are not only used by the client, but could also be accessed from other participants, which do have legitimate interests in the provided information. Besides the client and the server, the WebPKI states two other participants: As stated in chapter 2.1.5, browser vendors usually maintain the lists of trusted certificate authorities, hence being responsible for adding, removing or revoking CA certificates. By implementing a public accessible cache, software vendors would now be able to monitor the WebPKI in use, which simplifies certificate management and compliance checking, for example, by collecting only certificates for hosts from specific countries. Similar to certificate transparency, this reveals rouge certificates issued by trusted CAs. Also, this not only enables browser vendors to identify current attacks on the WebPKI, but also enables them to accurately state the number of hosts, which would need to change their certificate in case of a revocation of the CA certificate. This could be used to refine policies for CA certificate removal from browser software or to notify users in case of an incident at a CA of the WebPKI. Figure 3.2 shows, how NotaryCache enables participants to monitor caches.

But monitoring does not only harm the business interests of CAs, it also supports them by creating new security services, which could be sold to customers, who don't want to monitor caches on their own. A legitimate CA could then not only issue certificates for a target host, but also conduct a monitoring of other competitors by the monitoring of caches, which could be used to notify the customer of rouge certificates issued by foreign CAs.

In this architecture, the participants engage in different roles: The entity responsible for managing the cache will further be titled as the cache operator. The cache operator retrieves the certificate from the the target server and includes it in the cache according to the caching strategy. As already stated, this role is taken by the server operator. The entity, who is monitoring different caches, will be titled as the cache monitor. To enable the fast download of a cache, a third role is introduced to the general architecture. The cache replicate role is simply responsible for offering access to the cache file. It therefore replicates the cache from a cache operator by downloading the cache (passive replicate) or by receiving the cache from the cache operator (active replicate). The cache replicator could, for example, be implemented by a content distribution network to handle network load on cache operators, a public ftp server with write permissions for the cache operator or a service by the network operator of large computer networks, such as universities or big organizations. Figure 3.3 visualizes this categorization.

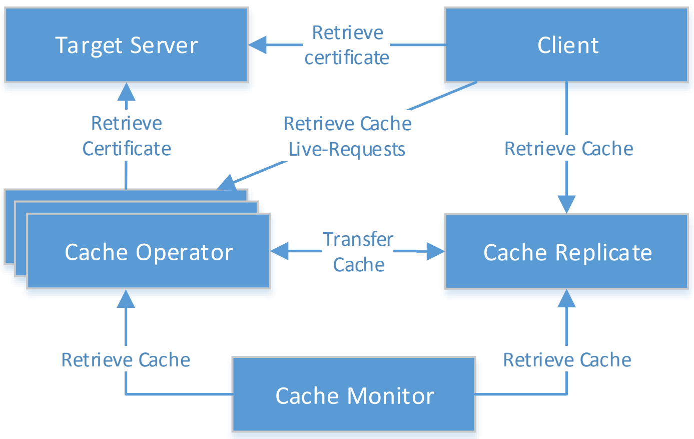

Figure 3.3: Generalized roles

3.5.2 Cache structure

The cache composes of three parts. The first part, the header, contains general information about the notary and the cache. The second part consists of a list of target hosts with various information necessary to the client. The third part provides a cryptographic signature over the header and the list of target hosts. The signature is used to verify the integrity of the cache by the client.

Header

The first part is the header, which contains the following fields:

IP+Port IP address and port on which the notary is accessible. This information is needed for the client to determine the geographical location of the notary as well the contact point to receive the notary configuration.

Hostname The hostname, which is used during the SNI extension and the TLS handshake protocol to establish a connection to the notary. If no hostname is given, then SNI is not used during connection establishment.

Validity start The timestamp, which states the issuance time of the cache.

Validity end The timestamp stating the point in time, when the current version of the cache will expire.

Digest Algorithm The algorithm used to calculate the digest of a certificate of a target host.

Signature Algorithm The algorithm, which is used to calculate the signature of the cache.

Signature

The last part of the cache consists of the signature. The signature is created with the private key of the notary by signing the hash of concatenation of the header section and the list of target hosts by using the algorithm specified in the cache header. However, the signature does not tell anything about the validity or correctness of the cache, but only ensures, that the cache was not altered during transmission. From the perspective of an attacker, who is only capable of attacking the networking of the client, this makes things more difficult, as he must also attack the notaries. However, the TLS connection to the configuration of a notary is again verifiable using other notaries, which broadens the set of hosts, an attacker has to approach.

Generation of the key material and signing operations can be done automatically by the notary without any user interaction. The public key of the notary is stored in its configuration, while the private key is stored only in the memory. As neither the server operator, nor any other user must use this private key, it can be stored inaccessible to any other system or service user.

The signature is appended to the cache after the list of target hosts to enable the client to verify the integrity of the cache.

List of target hosts

The main part of the cache contains a list of target hosts, called entries. An entry in the cache is created according to the implemented caching strategy. Usually, an entry is created when a the notary receives the certificate of a target host during evaluation. The cache only consists of a limited number of entries, so that it is not getting bigger than really needed, thus consuming less storage on the hard disk, than a database, which holds every certificate ever received - even when it's already obsolete. If the maximum size of the cache is reached, then old entries must be dismissed or overwritten with new entries.

A cache entry contains the following fields:

IP+Port IP address and port of the target host. This information is important, as there are a lot of hosts, which resolve to multiple IP addresses, each containing a different certificate.

Hostname The declaration of the hostname or domain name during connection establishment is needed by the target server to differentiate between multiple virtual hosts on one physical server. The hostname is stated in the SNI extension of the TLS connection setup. If this information is missing, the target server is not able to decide, which certificate to return to the client, which usually results in responding with a default but probably wrong certificate.

Key Algorithm The algorithms of the public keys, which are carried in the received server and CA certificates. There are two cases, in which this information is necessary: The first case is, that the target server may implement cipher specs, which require different public keys [24]. The second case is, that the CA holds multiple certificates with different public keys. While old clients only support certificates (and thus cipher specs) containing RSA[7] keys, newer clients are also able to handle ECC [8] or DSA [9] key material. The selection of the certificate is done by the target server during TLS handshake protocol, as shown in section 2.1.3. For privacy reasons, version numbers of browser software or other libraries should not be disclosed to the notary, as this could enable user tracking. To identify the correct certificate chain to validate, the algorithm of each public key must be determined. Otherwise, one could validate a certificate containing an RSA key with the digest of a certificate containing an ECC public key, which lead to false positives.

Digests A list of digests of the certificates, that were received during connection establishment. One digest is made of each certificate, that was received.

Roles A list of roles which are implemented by the target server. This information is mainly used during bootstrapping, as it enables clients to discover yet unknown notaries. This field is optional, as it does require a complete HTTP connection setup and information processing on the notary, which consumes resources possibly needed for other operations.

Similar to the other notaries, the fields IP/Port and domain are used to identify the target host and thus must also be used to identify the entry, which holds further information for that target host. Additionally to the existing approaches, the key algorithm field must also be taken into account, because a domain can be assigned multiple public keys each created by another supported algorithm. To enable the client to use the cache during target host evaluation after all, each cache entry must contain the digests of the certificates. The digests are calculated by the notary operator during connection establishment to the target server. However, adding entries to and removing entries from the cache is governed by the caching strategy.

3.5.3 Caching strategy

While the cache is central to the processing on the endusers side, the caching strategy is the counterpart to the notary. Therefore, the caching strategy is responsible for controlling the actions of the cache operator, like adding and removing entries or issuing and distributing the cache. That makes the caching

[7] Rivest Shamir Adlemann, an asymmetric cryptosystem based on the factorization problem
[8] Elliptic Curve Cryptography, an asymmetric cryptosystem based on the discrete logarithm problem
[9] Digital Signature Algorithm, an asymmetrical cryptosystem based on the discrete logarithm problem and Schnorr-Signatures

strategy responsible for cache operator acceptance from a technical perspective. Moreover, the cache strategy is used to align the notary to the business of the notary operator. In this approach, the server operator is assumed to be the only entity, who has a business interest and the technical ability to run a notary, which makes them the cache operators. In general, the caching strategy gives answers to the following questions:

- Which conditions must be satisfied to add an entry to the cache?
- Which conditions must be satisfied to remove an entry from the cache?
- Which conditions must be satisfied to issue the cache?
- Which conditions must be satisfied to distribute the cache to active cache replicas?
- How to store information temporary?
- How to sort the cache entries?

Conditions can be categorized based on two categories: Technical conditions and business conditions. Technical conditions comprise of typical server constraints, like bandwidth or traffic limitations, that result from contractual constraints. Also, timely constraints can be seen as technical conditions, for example, when an operation must only run at night, because resource consumption would be too high during the day. In contrast, business constraints directly result from business incentives of the notary operator or the key endusers. That can be, for example, a limitation on target hosts or clients, who are allowed to add new entries to the cache.

3.5.4 Cache configuration

The cache has attached both a public and private configuration. The public configuration comprises of additional information about the notary, which can not directly be mentioned in the cache, its connected active replicas and the cache, that is issued by that notary. The configuration must only be accessible over a TLS-secured connection, because security-sensitive information is transferred. The configuration contains the following fields:

Cache Version The version of the cache structure.

Contact An email address, which can be used to get in contact with the notary operator.

KeyID The keyID of an arbitrary encryption system, which could be used to encrypt emails to the email address stated in the contact field. As an example, the PGP keyID could be inserted here.

Notary protocol The field indicates the protocol, that can be used to request the certificate fingerprint for a given target host.

Replicate URIs A list of active cache replicates, which offer a download of the current version of the cache. A replicate URI is a triple consisting of the ip, port and a domain, which is used during the HTTP protocol.

Replica Probability A numeric value between 0 and 1 indicating the probability, that a replicate is chosen by the client to download the cache instead of the notary operator.

Public key The public key, which is used to validate the signature of the cache. The signature of the cache must be created with the respective private key.

Public Key Validity Period The period, in which the public key is valid. This field should enable caching of the public key on the client side to lower the number of requests to the notary, which otherwise must be accessed regularly to gather the public key to verify the cache signature.

Only the notary itself is able to change any information other than contact, keyId, the replicate URIs and the replica probability. This restriction is set to simplify configuration and to prohibit non-expert notary operators to make decisions, which could harm the security of the notary. This should enforce context-sensitivity and context-alignment.

The notary configuration must also have a private component, which is responsible for notary operation. This part of the configuration and the processing of its attributes directly lead to server operator acceptance. This configuration is aligned to the system context, which is defined by the relation between the used and the free resources regarding network, storage and cpu. Usually these resources are moreover determined by the contractual relationship between server operator and, for example, server owners, network owners et cetera. While the currently used and free resources are determinable by the notary itself, the traffic limitations, whose exceeding would cost money, must be set by the server administrator in the public configuration. This configuration must not be difficult, as those metrics are usually known to the server operator.

The private configuration consists of attributes, that are calculated by the application or set by the server operator in order to control the notary. Only the application is able to access these attributes. There are three basic attributes, which must be private to not endanger the server operator. Only the first two attributes are alterable by the server operator.

Maximum Traffic The maximum traffic of the host running the cache operator. This is usually determined by the contract the server operator closes with the data center operator.

Maximum Bandwidth The maximum bandwidth of the host running the cache operator. Again, this value is usually determined by the contract the server operator closes with the data center operator.

Private Key The corresponding private key to the public key from the public configuration. This attribute can not be accessed by the server operator.

The following attributes could further be thought of in order to control time- and resource-intensive tasks:

Minimum Load Period This attribute states the period, in which the server is not busy with other tasks. This period could be used by the cache operator to update the cache by executing time- or resource-intensive operations, such as cache generation.

Minimum Traffic Period This attribute states the period, in which the network is not busy with other tasks. This period could be used by the cache operator to update the execute bandwidth-intensive operations, such as role mining of target hosts or distribution of caches.

Free storage This attributes states the remaining free storage the cache operator could use to store additional information. This attribute could be used to dynamically adapt to situations, where storage might be low.

It is essential, that the configuration neither does overburden the server operator, nor require much time. In this configuration, every alterable attribute is easily determinable by the server operator, while every non-alterable attribute is easily determinable by the notary itself. Moreover, the public part of the cache configuration is heavily used in client processing and thus must be planned with a high level of correctness. In letting the server operator off the hook, client processing does not rely on his decision anymore and cannot be disturbed by wrong configurations, which could result from missing expert knowledge.

3.5.5 Client processing

Basically, client processing can be divided into three phases: Initialization of the system, retrieval of additional caches and operation. Figure 3.4 shows the different transitions of the phases.

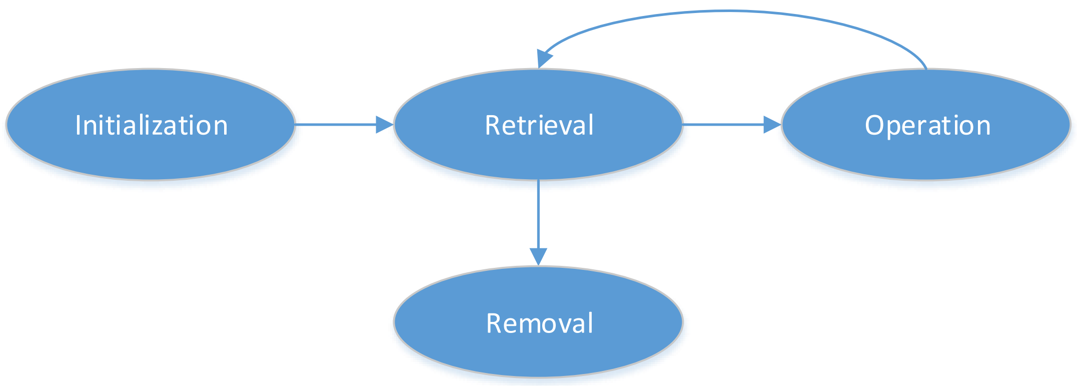

Figure 3.4: Client processing phases

Initialization

Client processing heavily relies on the secure initialization of the approach. It is crucial for the security of the system, that the first caches are retrieved by using a secure connection. It is therefore proposed to add a first set of appropriate caches to the browser software itself together with the TLS fingerprints of the notaries. Appropriate means, that other caches can be gathered and validated using these pre-installed caches. This enables the inclusion of a limited set of caches, that can be used to validate at least a limited set of target hosts. These target hosts can now be used to download new caches during retrieval. Alternatively, a list of fingerprints of notary operating target hosts could be pinned in the browser to enable the secure download of a small number of caches from those locations. Again, these cache are then further used to download new caches from locations, whose fingerprints are not pinned to the browser.

Retrieval & removal

Retrieving of a set of caches can also be termed as bootstrapping, because the system is extended from a basic set of preinstalled caches, which only enables the validation of further notaries, to a much more diverse set of caches able to validate any public TLS-enabled interface. Bootstrapping is repeated regularly, until no further caches are needed or available. The point of time of the retrieval must be further determined by the clients system and network load.

The retrieval phase is responsible for the discovery and download of new caches, which can be used in further bootstrapping attempts and during operation. The retrieval phase's discovery method depends on the assumption, that the caches and its cache operator services must be easily discoverable by the client. To keep service discovery efficient and to avoid the use of a central service, the well defined approach of "Well-Known Uniform Resource Identifiers" [56] is applied. This method relies on standardized prefixes, which are valid for the entire internet and defined in the "Well-Known URI Registry" maintained by the IETF. This approach is similar to the well-known ports, which is acknowledged by nearly all operating system vendors and service developers. With the completion of this thesis, the following prefixes will be requested:

- "notary-cache": The central point of downloading a cache from the cache operator.
- "notary-cache-config": The central point of accessing the cache configuration
- "notary-cache-replicates": The central point of downloading the cache from a cache replica. The answer contains a list of caches from arbitrary notaries, which can be provided to the client via /.well-known/notary-cache-replicates/*original-host*.
- "notary": A central access point to the notary implementation. The configuration indicates the protocol and the arguments for that interface.

The critical part of the retrieval phase refers to the download of the cache configuration from the cache operator. As already stated in section 3.5.4, the cache configuration provides the client with the public key, which is used to verify the integrity of the cache. As this information is critical to security - an attacker could alter the digests listed in every cache the client downloads - the configuration must be accessed via HTTP over TLS (HTTPS). To verify the fingerprints of the first set of additional cache operators, algorithm 6 is utilized, which is shown later. Moreover, availability of the notary must be given to download the configuration, which is a property only target servers and their server operators can assure.

Algorithm 1 shows the main retrieval process of a cache from a cache operator. The algorithm consumes a combination of the IP address, the port number and the hostname, which is used to identify the cache operator. It returns a valid cache file, which was either loaded from storage or received from either the cache operator or an arbitrary cache replicate. The algorithm further makes use of the the following functions:

- "retrieve-cache-from-operator", "retrieve-cache-from-replica": Retrieves the cache from the operator or the replica respectively.

```
Input: Potential cache operator host $operator_{ip}$, $operator_{port}$, $operator_{domain}$
Output: Cache $c$
Function retrieve-cache($operator$) begin
    retrieve-config-secure($operator$)
    if cache configuration retrieval successful then
        $pk \leftarrow$ Public Key from config;
        if cache for $operator$ already exists then
            $c \leftarrow$ load-cache($operator$);
            if validate-cache(c,pk)=True then
                return $c$;
            else
                delete($operator$);
            end
        else

        end
        $p_{replica} \leftarrow$ replica probability from config;
        $r \leftarrow random \in [0,1]$;
        $retrSuccess \leftarrow False$
        if $r <= p_{replica}$ then
            choose random replica $replica = \{ip, port, hostname\}$;
            retrieve-cache-from-replica($replica$, $operator_{ip}$) to $c$;
            if retrieval from replica successful then
                $retrSuccess \leftarrow True$;
            end
        end
        if $r > p_{replica}$ || $retrSuccess \neq True$ then
            retrieve-cache-from-operator($operator$) to $c$;
            if retrieval from operator successful then
                $retrSuccess \leftarrow True$;
            end
        end
        if $retrSuccess = True$ && validate-cache(c,pk) = True then
            return $c$;
        end
    end
    return error;
end
```

Algorithm 1: Retrieval of a cache

- “retrieve-config-secure”: Retrieves the configuration over HTTPS. Is is also assumed, that “retrieve-config-secure” makes use of algorithm 6 to validate the certificates, that are retrieved during the TLS handshake.
- “retrieve-digest-secure”: Retrieves the digest from a notary.
- “load-cache”: Stored caches are not downloaded again, until they become invalid. Instead, caches are loaded via the underspecified function “load-cache” from this storage.
- “delete”: Invalid caches are deleted by using the underspecified delete-function.

Algorithm 1 also makes use of the function "validate-cache", which is specified in algorithm 2. Each cache must be validated after retrieval to ensure integrity using the underspecified function "verify-cache-signature" and timely validity. The function is underspecified, as it depends on the actual algorithms used for signature and verification.

```
Input: Cache c; Public Key pk
Output: Boolean indication, whether the cache is valid
Function validate-cache(c, pk) begin
    $v_{start}$ ← validity start of $c$;
    $v_{end}$ ← validity start of $c$;
    $t_{now}$ ← timestamp now;
    if $pk$ is set then
        return (verify-cache-signature($c$, $pk$) = ok && $v_{start} > t_{now}$ && $v_{end} < t_{now}$);
    end
    return ($v_{start} > t_{now}$ && $v_{end} < t_{now}$);
end
```

Algorithm 2: Validation of a cache

Algorithm 1 only retrieves the caches. To utilize this function, a list of hosts has to be determined first, which is examined for their role in the system. If a host does offer any HTTPS-functionality or if the configuration is not found on the server, the host will be dismissed, as it is not needed anymore. Thus, only a list of arbitrary hosts must be determined.

There are two methods, which can be used to automatically obtain this list of hosts besides domain bruteforcing or configuration by an enduser, which are not suitable in regards to the requirements. The first approach is to look at existing caches and try to download caches from hosts, where the roles field indicate the cache operator or cache replicate roles. This method has the advantage, that all hosts are capable of HTTPS and are probably already evaluated by the cache operator.

```
Input: List of available caches $C_{available}$
Output: List of new caches $C_{new}$
Function retrieve-from-caches($C_{available}$) begin
    foreach Cache c in $C_{available}$ do
        retrieve-secure-config($operator$) if cache configuration retrieval successful then
            $pk$ ← Public Key from config;
            if validate-cache(c, pk) then
                foreach Entry e in list-of-hosts(c) do
                    if e is cache operator || e was not evaluated yet then
                        retrieve-cache($e$) and add to $C_{new}$;
                    end
                end
            end
        end
    end
    return $C_{new}$;
end
```

Algorithm 3: Retrieving new caches from existing caches

The second approach is to examine the browsing history of the client, which could reveal potential cache operators. This approach is much more expensive, as it forces the client to establish connections to all hosts in the history without any pre-sorting. However, it has the advantage to determine cache operators not known yet. Therefore, this method must only be executed, when the enduser is not interfered.

```
Input: List of recently accessed servers $H$
Output: List of new caches $C$
Function retrieve-from-history(H) begin
    foreach Host h in H do
        retrieve-cache($h$) and add to $C_{new}$;
    end
    return $C_{new}$;
end
```

Algorithm 4: Retrieval of caches from history

To optimize the speed of the bootstrapping process and to reduce the generated traffic, it is further proposed to facilitate caching of the cache configuration. The most critical attribute of the cache configuration, the public key, therefore owns a validity period. If this period is over, a potentially cached version of the cache configuration has to be withdrawn. For simplicity reasons, this functionality is dismissed in the algorithms.

The bootstrapping process is executed regularly to download new caches from different notaries. The caches are neither trusted by the client, nor must the information they contain be correct, because the notary itself could be ran by an attacker, who is able to create entries based on his own needs. That leads to the problem, that a cache could possibly contain information, which is not valid. However, it is not decidable, if a cache was issued by an attacker or if a target server, whose certificate digest was stored in the cache, is under attack. A client must therefore not rely on a little set of caches during operation phase, but must utilize a set as big as possible.

Operation

The caches are used to verify the certificate, which is received during connection establishment by the client. The client looks up the entry of the target host and extracts the digest of the certificate from this entry. This process is repeated with multiple caches from different notaries. The client now has the ability to decide on his own, whether to trust the certificate based on a number of cache hits and certificate digest accordances.

According to the threat model, it is important for the client to only choose caches from distant countries for the evaluation to overcome the problem of nation-state adversaries. There are different ways to examine the geolocating of an IP address. The most discussed approaches in research are measurements of different timings ("constraint-based geolocation") and network characteristics from different places, whose locations are known to the client. Tho goal of those approaches is to employ a triangulation-based technique, whereas the ICMP protocol plays an essential role. However, due to the lack of services to determine the location, those approaches are not applicable in this scenario. Other approaches comprise of whois- and DNS-data, or the examination of user-submitted information [53]. Geolocation databases often use those information for mapping ip address ranges to countries and cities. The problem with geolocation databases is, that they are not considered accurately [61] for the determination of location information, if the location should be determined on city-level. However, the authors claim, that at least the country can be determined accurately. In the case of NotaryCache, this accuracy is fully acceptable, because governments are considered as attackers. That assumption has the effect, that it makes no difference, if an arbitrary host, e.g. a target host or a notary, is located in one or another city in the same country. Thus, for the sake of location determination, geolocation databases, which map IP address ranges to countries, are used in NotaryCache. A lookup of an IP address will further be specified as *location(h)*. This procedure then looks up the IP in the database and returns the country, respectively the country code.

Based on this procedure, the origin of the caches can be determined by examining the location of the responsible cache operator, whose IP address is stated in the header of the cache. As already mentioned, it is essential, that the cache operator is neither located in the country of the target server, nor in the country of the client, to make the cache as independent as possible from potential attackers. The process of examining and selecting a cache is shown in algorithm 5.

```
Input: List of available Caches $C_{available}$
Input: Target host $h$
Output: List of appropriate Caches $C_{target}$
Function select-appropriate-caches($C_{available}$, $h$, $client_ip$) begin
    $l_{target} \leftarrow location(h)$;
    $c_{client} \leftarrow location(client_ip)$;
    foreach Cache $c$ in $C$ do
        if validate-cache($c$)=True then
            $l_{cache} \leftarrow$ location(IP-field from $c$-header);
            if $l_{cache} \neq l_{target}$ && $l_{cache} \neq l_{client}$ then
                add $c$ to $C_{target}$;
            end
        end
    end
    return $C_{target}$;
end
```

Algorithm 5: Cache selection

The selected caches are then utilized to evaluate the target host. The basis for the evaluation is composed of a list of digests. The first digest is calculated by the client from the server certificate. This digest will be called the original digest. The other digests are extracted from the caches, by looking up the target host in the cache and extracting the digest for that target host. The process of calculating the frequency distribution is shown in algorithm 6.

For every known digest, the probability is calculated, which results in a frequency distribution, that is later used to specify the current situation. In this frequency distribution, p_0 describes the probability of the original digest, which is calculated by counting the occurrences of this particular digest in the caches. In the optimal case, each cache supplies the same digest as the client, thus resulting in $p_0 = 1$ for the original certificate, which indicates $|P| = 1$, whereas P is the set of all probabilities p_i. In contrast, in the

```
Input: List of appropriate Caches $C$; Target Host $h$; Digests of certificates $D_{client}$; Index $N$ indicating
       digest to calculate frequency distribution
Output: Frequency distribution of digests from caches $p_i$ from total $n_{total}$ caches; List of notaries for
        potential live-requests $lr$
Function frequency-distribution(C, h, $D_{client}$, N) begin
    $d_0 \leftarrow$ N-th digest from $D_{client}$ /* List of digests, $d_0$ is the original digest */
    $n_0 \leftarrow 1$ /* List of counts of digests, $n_0$ is the count for the original digest */
    $p_0 \leftarrow 0$ /* List of probabilities of digests */
    $n_{total} \leftarrow 0$ /* Total number of caches, in which the target host was found */
    $lr \leftarrow 0$ /* List of notaries for live-requests */
    foreach c in Caches do
        if validate-cache(c)=True then
            $e \leftarrow lookup(c,h)$
            if e was found then
                /* An entry was found in cache for this target host → compare digests
                   */
                $digest_{cache} \leftarrow$ N-th digest from $e$;
                $n_{total} \leftarrow n_{total} + 1$;
                $l \leftarrow length(n)$;
                $found \leftarrow False$;
                if $digest_{cache} != d_0$ then
                    add $notary_{cache}$ to $lr$;
                end
                for $i = 1$ to l-1 do
                    if $digest_{cache} = d_i$ then
                        $n_i \leftarrow n_i + 1$;
                        $found \leftarrow True$
                    end
                end
                if $found = False$ then
                    /* Digest was not found → add new element to $n$ and $d$ */
                    $n_l \leftarrow 1$;
                    $d_l \leftarrow digest_{cache}$;
                end
            end
        end
    end
    for $i = 0$ to length(n)-1 do
        $p_i \leftarrow \frac{n_i}{n_{total}}$;
    end
    return $p, n_{total}, lr$
end
```

Algorithm 6: Calculation of the frequency distribution of a certificate

worst case, every other cache has supplied another digest, which results in a very small probability for p_0.

Moreover, the algorithm returns a list of notaries, whose caches did contain the domain, but with a different certificate digest. The domain could be requested from those notaries. If the notary then returns the correct certificate, the cache must be seen as invalid. This also leads to a correction of the respective fraction, as the number of occurrences of a certificate has changed.

It is also required, that n_{total} is high enough to justify a decision. Despite the number of existing approaches, no reasonable size for n_{total} could be found in existing literature yet. Similar to the number of notaries to be checked in Convergence[10], we thus assume at least three caches to be checked. If less then three caches were checked, the result must not be seen as meaningful. However, if the set of appropriate caches is bigger than n_{total}, live requests could also be issued to those notaries, whose cache didn't contain any information about that domain, to acknowledge the result.

Summarized, the algorithm calculates $P = \{p_0, p_1, ..., p_n\}$, whereas p_0 is the original digest from the certificate, that was received by the client, and p_1 to p_n are the probabilities of digests $\neq p_0$. That leads to four cases:

1. $|P| = 1$ and $p_0 = 1$: The digest, that was calculated from the certificate received by the client, conforms with the digests extracted from all appropriate caches. In this case, the client is assumed to be secure.

2. $\forall p_i \in P \backslash p_0 : p_i < p_0$: There are other digests found besides the original digest. However, the probability of all other digests is smaller than the probability of the original digest, that was received by the client. Again, the client is assumed to be secure.

3. $\exists p_i \in P \backslash p0 : \forall p_j \neq p_i : p_i = p_0 \pm \delta \wedge p_j \leq p_0 \pm \delta$: There are other digests found besides the original digest. At least one digest other than the original digest has the same probability as the original digest. That means, that there are other certificates, which are provided with the same probability. Since the caches are managed by notaries from other countries, that could denote to a content distribution network, which provides certificates based on the location of the client. As such, this case is assumed to be secure for the client.

4. $\exists p_j \in P \backslash p0 : p_j > p_0$: There are other digests found besides the original digest. In contrast to the other cases, there are also digests, which are more probable than the original digest, indicating an attack. This is especially the case, if the certificate was only received by the client and did not occur in any cache.

In case of the last three scenarios, the decision process can further be supported by live-requests to determine the topicality of the cached information. This can be useful to support or refuse a result calculated above using only cached information, which are not up-to-date when it comes to revocation or other reasons of certificate exchange. This is where the notary comes into play. A live-request is issued to a subset of notary components of the previously examined caches. This subset must consist of an equivalent number of caches responsible for approval and denial. Information about the notaries are gathered from the cache configuration. The process is shown in algorithm 7.

Correctness can also be calculated using cached digests, if $|P| > 1$: $c_{cached} = p_0 / (\sum_{i=1}^{n} p_i)$. The comparison of both numbers c_{cached} and $c_{realtime}$ can now be used to support or refute a result. In common, a higher $c_{realtime}$-value indicates, that some of the negative caches contain wrong information,

[10] Version 0.09

```
Input: Subset of previously used caches $C$, Target host $h$, Original digest $d_{orig}$
Output: Correctness $\in [0, 1]$
Function live-requests(C, h, $d_{orig}$) begin
    $numCorrectness \leftarrow 0$ /* notaries approving the original digest */
    $numIncorrectness \leftarrow 0$ /* notaries denying the original digest */
    foreach $c \in C$ do
        $operator \leftarrow$ IP, Port, Domain-Field $c$-header;
        retrieve-config-secure($operator$);
        if cache configuration retrieval successful then
            $notary \leftarrow$ notary protocol and version from cache configuration;
            if $notary$ supported then
                $d_{retr}$ = retrieve-digest-secure($h$,$notary$);
                if $d_{retr} = d_{orig}$ then
                    $numCorrectness \leftarrow numCorrectness + 1$;
                else
                    $numIncorrectness \leftarrow numIncorrectness + 1$;
                end
            end
        end
    end
    return $c_{realtime} = numCorrectness/(numCorrectness + numIncorrectness)$;
end
```

Algorithm 7: Realtime requests

which supports the decision to continue with connection establishment. In contrast, a lower $c_{realtime}$-value indicates wrong information in caches, which approved the original digest. This leads to refusal of the continuation. The more live requests are executed, the more precise the result will be and the better is the decision, which is planned upon checking the caches, is supported or refused.

However, the more requests are executed, the longer the connection establishment will last from a user perspective. This indicates, that the size of the subset of caches has to be aligned to the actual case. For example, there is no need to execute realtime requests in case one. In case two, the set should only contain a small number of caches, and so on. To further optimize connection establishment for future requests to that target host, the use of certificate pinning is proposed. This was already done in previous implementations [73]. As an advantage to previous work, pinned certificates could be automatically evaluated with caches and realtime requests and according to the results from a cache-based evaluation they could be renewed, dismissed or updated in an asynchronous way respectively.

4 Implementation

The implementation of NotaryCache is a Java 8 based application, which is delivered together with this thesis. However, an up-to-date version of NotaryCache is provided via `https://github.com/letzkus/notary-cache`. To simplify the build process of NotaryCache, Maven2 was chosen. Maven can be used to build both the client library and the server implementation, which contains a basic caching strategy and a basic implementation of the cache. Moreover, a simple notary implementation is given, as well as an example interface for the implementation of a Perspectives notary.

4.1 Design principles

NotaryCache follows various design principles, which are introduced in the next sections. The design principles were chosen in accordance to the requirements listed in chapter 3.3 to provide, for example, ease of use, adaptability, context-sensitivity and standardization.

4.1.1 Abstraction

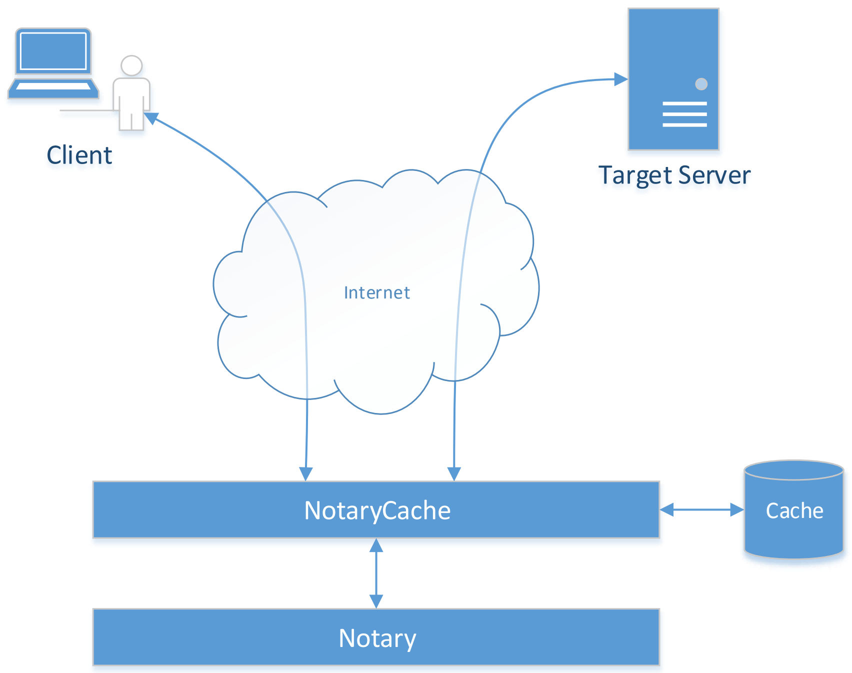

Figure 4.1: Abstraction

As shown, standardization is a big issue, since every notary vendor has implemented his own interface. NotaryCache positions itself in front of the notary by abstracting from the concrete implementation of the notary. This is done by implementing an overlay approach. In this approach, NotaryCache takes the role of a new layer between the client and the notary, as shown in figure 4.1. This transparent position enables NotaryCache to be able to receive both requests to the notary and the responses from the notary to the client. Thus, NotaryCache is able to manage the cache independently of the notary. That means, that the notary does not need to implement any caching functionality by itself. This makes NotaryCache easily implementable to notary operators without any changes to existing approaches.

The overlay approach also has the advantage, that NotaryCache is able to specify a common interface used to issue standardized requests to different notaries. Requests received on this interface can easily be translated to existing notary-specific requests. As shown above, most interfaces of existing approaches do not fit to the given requirements by ignoring different attributes, which leads to false-positives and erroneous responses. Instead, NotaryCache gives a standardized interface for clients, which fits to the requirement. This interface is used to both access the cache managed by NotaryCache, and the functionality of the notary by translating requests into notary-specific protocols. Thus, a standardized interface not only could enable interoperation between one client and various notaries, but also raises security of most notary approaches without directly changing them.

4.1.2 Modularization & event-based messaging

NotaryCache is composed of different modules, which communicate via a publish-subscribe-mechanism. This mechanism is implemented in the *Event Manager* module. In this mechanism, modules send specific requests to the *Event Manager*, which distributes the event to all other modules, that previously subscribed to that specific events. After receiving those events, the module can take specific actions based on the information provided together with that event. Sources for events are the *Server Interface* module, which provides interfaces, that can be used by clients to access the cache, the configuration or the notary itself, as well the *Hardware Monitor* module, which regularly monitors various hardware parameters and regularly sends events containing those information to the *Event Manager*. This enables arbitrary modules to adapt to various hardware measurements. The core architecture is shown on the left side of figure 4.2.

On the right side of figure 4.2, the interfaces to the cache and the notary are shown. The interface to the cache is presented by the caching strategy, which controls operations like adding or removing entries from the cache or the issuance of the cache. These modules are use case dependent and usually user-defined and are thus not part of the core architecture.

4.1.3 Ease of deployment

NotaryCache makes use of various existing and proven libraries. One example is jetty, a widely known servlet engine and HTTP server. Since NotaryCache implements jetty directly without relying on a dedicated instance of jetty, a system operator does not need to configure anything except NotaryCache's internal properties.

Moreover, NotaryCache is delivered as a simple ZIP-archive. Thus, installation does only consist of extracting the archive to an arbitrary directory and executing the scripts available for Microsoft Windows

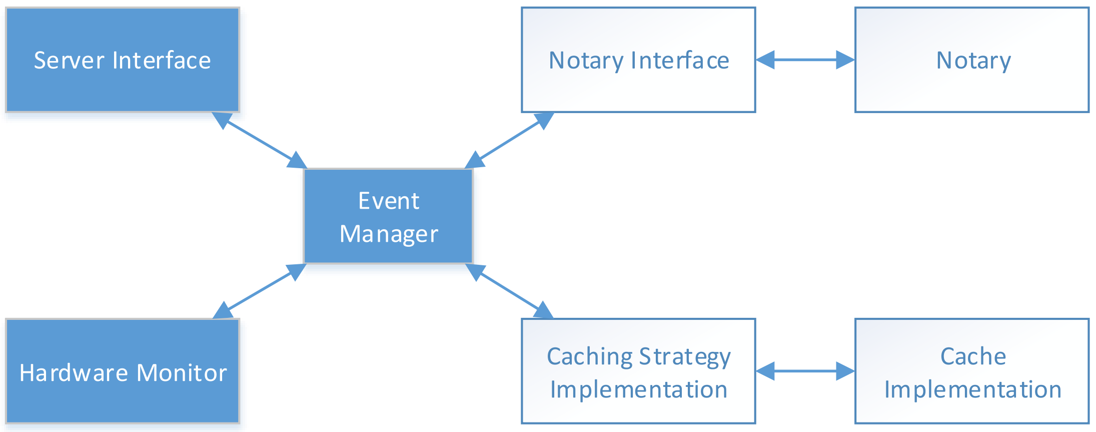

Figure 4.2: Modularization & event management

and Unix OS. Also, all needed resources and dependencies except an implementation of the Java 8 VM are delivered together with NotaryCache.

It is a foundational design principle of NotaryCache to introduce as few changes as possible to the underlying system configuration and the WebPKI as a whole. Basically, NotaryCache is designed to be operated behind an existing web server, which acts as a proxy. Most existing web server implementations already provide this feature, for example mod_proxy for the Apache Web Server [1] or the proxy module for nginx [2]. This has multiple advantages. First, no changes must be done to the firewall solution, if an existing website is operated at tcp port 80 (HTTP) and 443 (HTTPS). Second, features from those proxy implementation can simply be reused, such as authentication, logging or load balancing. Third, by using standardized uniform resource identifiers no conflict is created with existing software. Thus, NotaryCache can easily be implemented and operated in parallel to existing web applications.

A minimal configuration consists of six parameters to be set, each well documented in the configuration file and configurable without any expert knowledge about notaries or the WebPKI:

- *internal.interface*: The interface, on which NotaryCache listens.
- *internal.ip*: The internal IP of the interface, usually 127.0.0.1.
- *instance.port*: The TCP port, on which NotaryCache listens
- *external.ip*: The IP, where the host NotaryCache runs on is accessible.
- *external.hostname*: The domain or hostname that resolves to the IP of the host.
- *external.port*: The TCP port the web server listens on.

Advanced installations could also automate this configuration by simply setting those parameters based on automated system evaluation, for example routing tables or available network interfaces. Thus,

[1] http://httpd.apache.org/docs/2.2/mod/mod_proxy.html, Last access: 15.04.2015
[2] http://nginx.org/en/docs/http/ngx_http_proxy_module.html, Last access: 15.04.2015

NotaryCache could be deployed by distribution vendors directly without any configuration effort for the system operator.

4.2 Client

The algorithms of the client specification are implemented in Java 8 as a library to enable application developers to efficiently implement the approach in their own applications, as for example proposed by Fahl et al [35]. Basically, the library provides existing applications with a custom *X509TrustManager* object, that can easily be registered in the default or in a custom SSLContext. The application developers can simply add a list of hosts, which are then used by the NotaryCache client implementation to gather new caches. During the establishment of a TLS session, the *X509TrustManager*-object is automatically called by the Java Virtual Machine without any further consultation of the developer. Example code can be found in the official repository: `https://github.com/letzkus/notary-cache/`.

5 Evaluation

The evaluation is composed of two parts: The first part covers a statistical examination of NotaryCache when used by an individual user. By using a Monte Carlo Simulation, these statistical metrics are then extrapolated on a global scale and the benefits of NotaryCache will be shown in comparison to existing notaries. The second part covers the gap analysis regarding the requirements, which have been defined in chapter 3.3.

5.1 Simulation

In the following sections, the situation of an individual user is evaluated and then projected on a global scale.

5.1.1 Individual user

From an endusers point of view, NotaryCache consumes resources by downloading and storing the cache as well as its configuration on the client. Thus, the filesize is an essential metric, as it determines the amount of traffic and the diskspace on the client. Especially when dealing with a with 100 to 150 caches, diskspace must be considered as a main factor.

The cache structure consists of a three partitions: While header and footer are comparatively small, the list of hosts mainly determines the size of the cache. To determine the filesize related to a number x of entries, the first x hosts from the Alexa top 1 million list of websites are added to the proof-of-concept implementation. The generated cache is then downloaded to a client. In the provided implementation, the cache can be returned in a human-readable text format as well as in a LZ77 compressed format, which compresses the textfile to 30% to 35% of its original size. Table 5.1.1 shows the filesize of the cache in kilobyte in relation to the number of hosts that were added to the cache. Also the average size per entry is calculated for each list.

# of entries	Uncompressed	Compressed	Average size of an entry
100	22.362	7.918	0.220
250	54.539	17.717	0.216
500	104.466	33.519	0.208
750	157.400	50.196	0.209
1000	204.968	67.109	0.204
10000	2052.692	684.002	0.205

Table 5.1: Filesize of caches related to number of entries

The size of an entry mainly depends the number of received certificates. For each certificate, the implementation calculates a SHA256 digest and stores it in the cache. Thus, the more certificates are received, the bigger the entry will be. This property must also be considered when looking at the size of live requests and their responses. Here the number indicates, that it is slightly better to request hosts from lower parts of the Alexa top 1 million list, as higher entries tend to deliver more certificates, for example due to a better conformance with standards - not only the server certificate is delivered, but also the CA certificates to enable the client to fully reconstruct the certificate path - or due to a bigger certificate chain.

Another criteria, which affects the size of the cache, is the type of the IP address, as IPv4 addresses are usually shorter than IPv6 addresses. Only the minority of websites support IPv6, yet, but as the number of IPv4 addresses is strongly limited, it must be expected that the number of IPv6 capable target hosts (and clients) will rise in the near future. Comparison of a cache, which only contains IPv4 addresses with a cache containing only IPv6 addresses for the same hosts and certificates reveals a 11% increase of the filesize.

When it comes to network traffic, also the average size of an configuration must be taken into account. The size of the configuration mainly depends on the key generation algorithm and the configured keysize for the cryptographic material. If RSA with a keysize of 4096 is used, a configuration has a average filesize of 520 to 600 bytes. That makes the configuration insignificantly small when compared to the cache.

In general, the traffic, that is generated on client side, is determined by two factors:

- The number of caches and their configuration.
- The number of live requests, which must be issued, if a host could not be found in the caches.

In theory the number of caches is not restricted. However, in practice the client must restrict the number of caches to keep diskspace and traffic consumption low. Table 5.1.1 shows the storage and traffic consumption for a client, whichpram has to download and store an arbitrary number of hosts in relation to the number of entries per cache.

# of entries / # of caches	100	1000	10000
4	89,448	819,872	8210,768
16	357,792	3279,488	32843,072
32	715,584	6558,976	65686,144
64	1431,168	13117,952	131372,288
128	2862,336	26235,904	262744,576
256	5724,672	52471,808	525489,152

Table 5.2: Storage and traffic consumption related to filesize and number of caches

In addition to the traffic generated by downloading or updating the caches, traffic is also generated by creating live requests for hosts not contained in the list of available caches. In the proof-of-concept implementation, live requests are implemented as GET-requests with one to four parameters (ip, port, hostname, algorithm), which are send via HTTPS to the notary-interface, which forwards them to the actual notary implementation or the example notary implementation respectively. Thus, the size of a live request can be determined by summing up the TCP and TLS connection overhead, the HTTP parameters and the response from the notary, which consists of a status integer, a newline symbol and the response from NotaryCache. If the average size of a NotaryCache response is assumed to be 204 bytes - the current implementation of NotaryCache reformats the response from the notary to format of an entry -, both request and response consume around 3,7 kilobytes.

5.1.2 Global view

To determine the optimal cachesize and the benefits of NotaryCache in a large scale scenario similar to todays internet, requests are created based on a Zipf distribution and the metrics are calculated numerically using the Monte Carlo method.

Monte Carlo simulation

The idea behind the Monte Carlo method is to experimentally solve a deterministic problem[39]. While deterministic means, that the problem itself is not prone to any random process, other factors, such as input variables, may be. The Monte Carlo method is now used to calculate a meaningful result for these randomly distributed input variables. In general, the Monte Carlo method therefore consists of four steps:

1. Definition of input variables and their domain
2. Sampling of input data from a probability distribution
3. Using the input variables to solve the deterministic problem
4. Aggregating the different results

In this simulation, the optimal number of entries per cache is numerically converged from the number of hosts and users on the internet. The main goal of this simulation is, that, on the one hand, a cache with a realistic cachesize is determined, which contains a list of hosts, that are used by as many users as possible. The second goal is to calculate the benefits, which result from applying the cache to a real world scenario in contrast to a existing notary, that does not offer any caching, such as SignatureCheck.

The input variable consists of a fixed number of hosts and a fixed number of unrelated requests to that hosts. As mentioned above, it is assumed, that each host receives 1000 requests on average. However, the requests are randomly generated based on Zipf distribution. In general, the Zipf distribution approximates Zipf's law, which describes the property of an object that its frequency is inversely proportional to its rank. Therefore, the Zipf distribution belongs to the family of power law distributions. Mathematically, the Zipf distribution is formulated as

$$p(x) = \frac{1}{x^{\alpha}}, x > 0. \qquad (5.1)$$

The Zipf distribution is known to be valid for many scenarios. Breslau et al. [21] have investigated the Zipf distribution in web caching and discovered, that the relation between hosts on the internet and their request frequency also follows the Zipf distribution. The results were also confirmed by Adamic et al [2]. From a practical standpoint, the existing work shows, that there are many hosts on the internet, which are frequently requested, while there are others, which are nearly never requested. However, to our knowledge, there has been no agreement on α yet. While Breslau et al. assume $\alpha = 0.7$, Adamic et al. propose $\alpha = 1.0$. In the context of this simulation, both values are simulated and compared with each other.

To determine the average number of requests per user for a given host, the number of affected users is also given as input to the simulation. As the relation between human endusers and requests could not be found in existing literature, an uniform distribution is assumed. In the context of this simulation, that assumption implies, that an arbitrary host is accessed by x users, if the simulation reveals at least x requests for that host. It will be shown later, that the choice of the correct distribution is essential in order to achieve the full benefits of NotaryCache and that this topic definitely needs to be further investigated.

In the Monte Carlo method, the calculation of the number of entries in a cache for a given number of hosts, requests and users is the deterministic problem, that is to be solved for a randomly distributed list of requests. Moreover, the problem is solved for different numbers of users to examine the different effects, when there are more or less users than hosts. The sourcecode of the simulation is provided together with the implementation. For the results, which are shown in the next section, the following default parameters are set:

- hosts$= 100,000$
- requests$=100,000,000$
- users$=$hosts $*0.001$, hosts $*0.005$, hosts $*0.01$, hosts $*0.05$, hosts $*0.1$, hosts $*0.5$, hosts $*1$, hosts $*5$, hosts $*10$, hosts $*50$, hosts $*100$

The users are calculated in relation to the number of hosts. They will form the x-axis, whereas the y-axis is given by the size of the cache or the benefits respectively. To keep the results clear from any further modification, it is further assumed, that no key pinning mechanisms are implemented, which would additionally lower the number of live requests to notaries. That implies, that at least one request to a notary must be made for every request to a target host. For clarity reasons it is also assumed, that the caches only contain valid information.

Results

Figure 5.1 shows the result of a Monte Carlo simulation with $hosts = 100,000$ and $requests = 100,000,000$. In this graph, the blue plot states the number of entries in the cache for a given number of users and the red plot states the gained benefit at this point. For example, if looking at x$=10^4$, the cache contains hosts, which were accessed by 10^4 users. In this simulation, this list comprises of 692 hosts, which means, that around 20% of all requests could be saved, if NotaryCache was applied.

It is seen, that NotaryCache is powerful, if applied to $5 * 10^3$ to 10^4 users. In this scenario, a benefit of about 20% to 50% could be reached with a maximum cachesize of 1000 entries.

Additionally, figure 5.1 contains another graph, which relates the number of entries to the gained benefit without incorporating the users. Again, this graph shows, that about 50% benefit can be achieved with a reasonable sized cache.

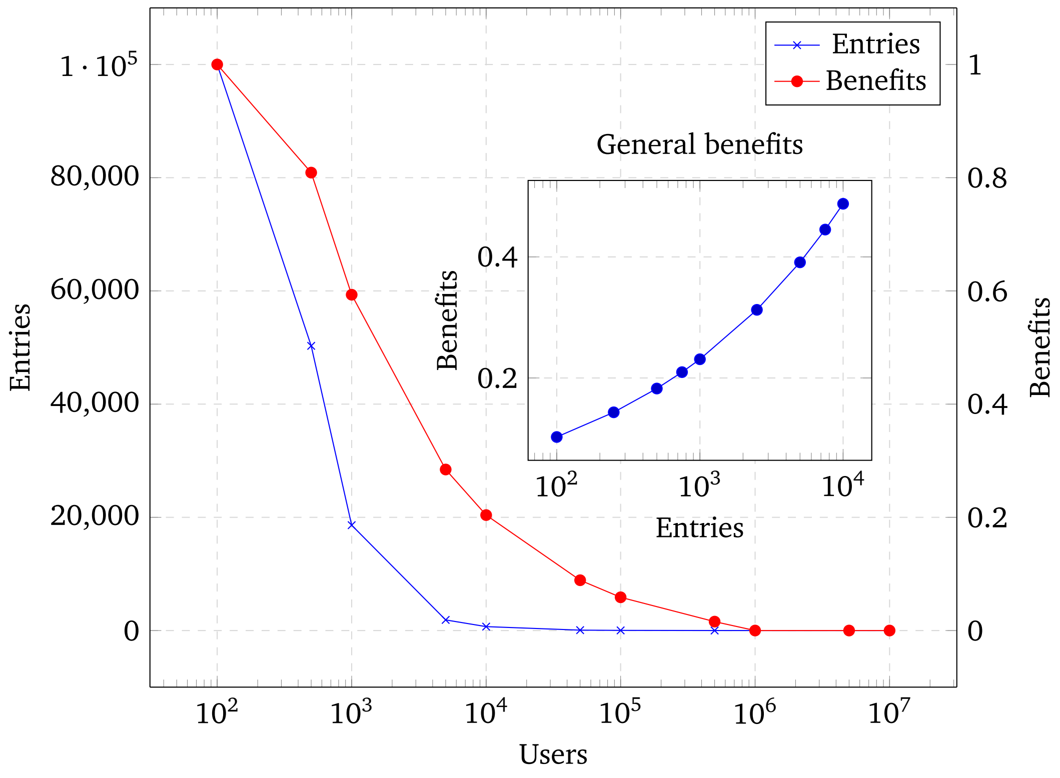

Figure 5.1: Plot for hosts=100k, requests=100m

The number of affected users can be increased by considering more than 1000 requests per host. An example is shown in figure 5.2. The left graph shows the original plot, as it is also shown in figure 5.1. The second graph is generated from the same input, except, that the number of requests is raised to one billion. One can see, that the entire graph is shifted to the right, which means, that more users are affected by the cache. Unfortunately, this does not increase the benefits of a reasonable sized cache.

The right graph shows the effects, if the distribution's parameter α is raised from 0.7 to 1.0. One can see, that the benefits are higher in general and that less entries are needed to affect the same number users. However, the α can not be adjusted by single person, but results more from the nature of the internet as a complex network.

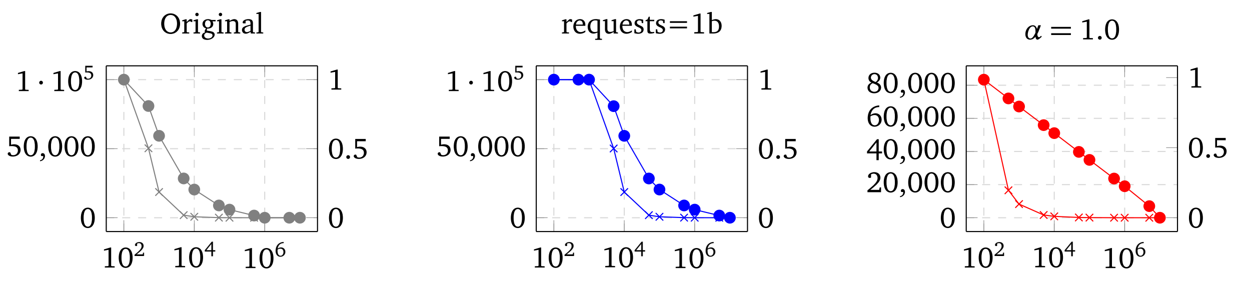

Figure 5.2: Effects of different configurations

Figure 5.3 shows the same graphs for $hosts = 1,000,000$ and $requests = 1,000,000,000$. It can be recognized, that the benefits are notably lower than in figure 5.1. Due to the high number of hosts, it is impossible to achieve a at least 20% benefit without increasing the cache to an unreasonable size.

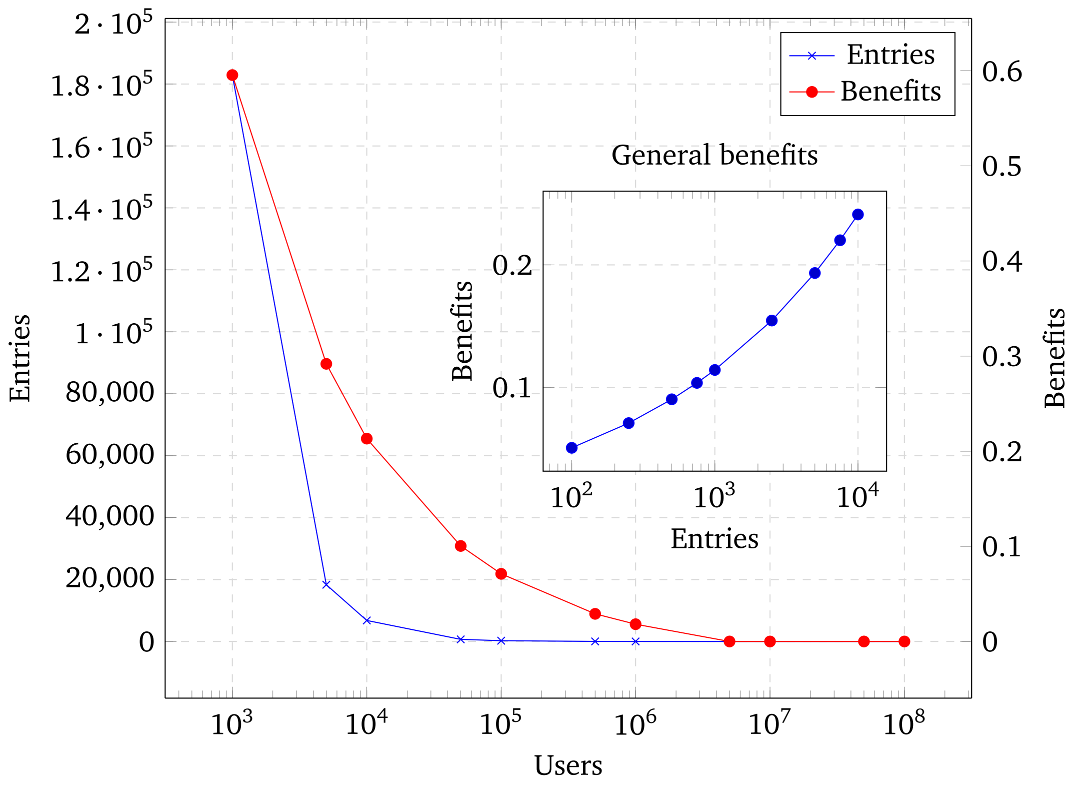

Figure 5.3: Plot for hosts=1m, requests=1b

These numbers, which were determined using the Monte Carlo method, must be seen in the context of one single cache. However, NotaryCache makes use of different caches, that could contain entries, which are not contained in any other cache. As such, the benefits could be achieved by splitting large numbers of hosts into smaller parts, which could simply be covered by one or more single caches. For example, if considering the results from figure 5.3, it would be possible to achieve 60% benefit with 20 caches, each holding 1000 entries, to affect 10^3 users.

Realtime requests

To calculate the additional traffic, that is caused by NotaryCache, the remaining requests, which are not covered by any cache, are taken into account. As shown in the last paragraph, nearly all requests could be covered by a suitable set of caches. However, that requires knowledge about all hosts and the distribution of requests to this hosts. As this does not exist, yet, only one cache will be considered in this evaluation. The cache is assumed to have the above mentioned configuration of $hosts = 100,000$ and $requests = 100,000,000$. Moreover, a maximum number of users of $maxUsers = 1,000,000$ is assumed. A similar host-user-ration can be found on the internet, where the ratio is about 1 : 8.

The result is shown in figure 5.4. One can now determine the traffic from choosing a number of users, whose common hosts should be covered by the cache. The red graph states the traffic generated by live requests for all users. The blue graph states the traffic for this number of users, that is generated by downloading the respective cache, which is calculated by multiplying the number of common hosts for these users with the average size of an entry. The green graph states the traffic, that is saved for a given number of users.

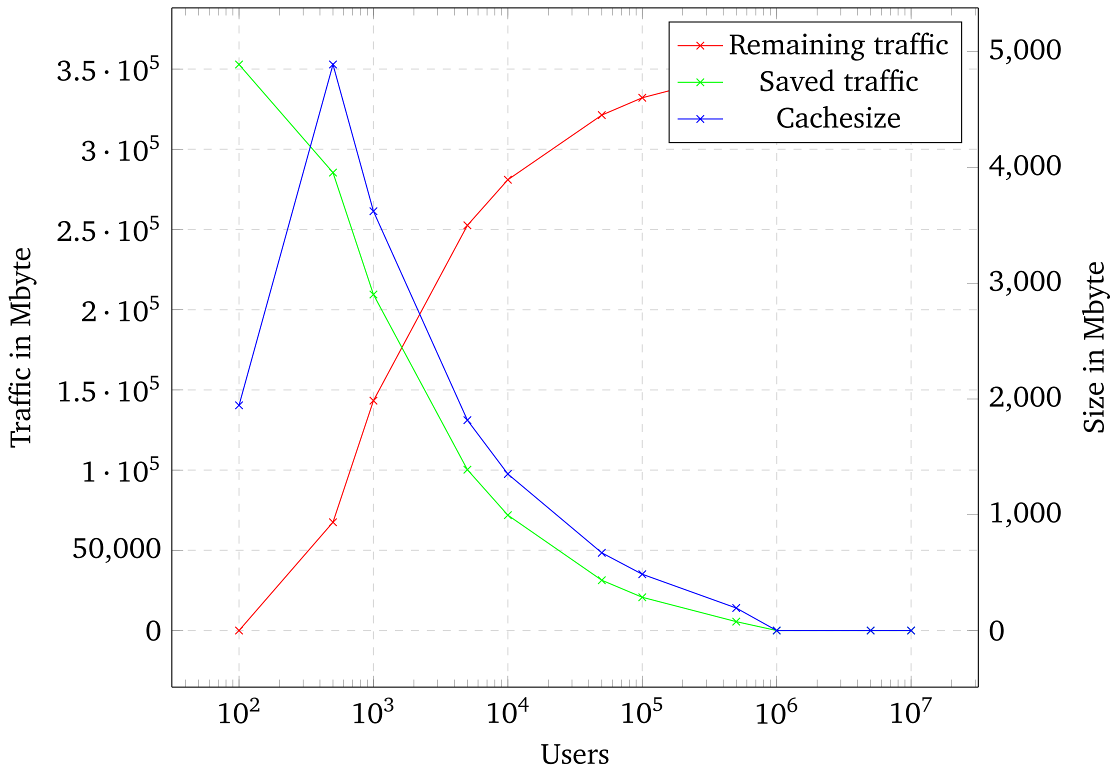

Figure 5.4: Additional traffic

For example, for 10^3 users, the additional traffic would comprise of

- approximately 150 Gbyte traffic generated by live requests
- approximately $3,500$ Mbyte traffic generated by downloading the cache

In contrast, approximately 200 Gbyte of traffic would be saved, that would otherwise be generated by live requests. If assuming, that 10^3 users are responsible for all requests to the covered hosts, that would decrease the number of requests by around 57%. In the worst case, the cache would be downloaded by all users, resulting in $3,500$ Gbyte of traffic for a cache, which is only used by 10^3 users. That means, that traffic would increase by the factor 10.

This evaluation shows, that the choice of the distribution of requests to users is important, when it comes to the actual calculation of the consequences of deploying NotaryCache to a real world scenario. Here, the assumption of a uniform distribution will reveal the worst case scenario, in which all clients must download the cache in order to achieve the calculated benefits. In contrast, by applying the Zipf distribution, the cache only needs to be downloaded by the users, who establish the majority of requests, in order to achieve the full benefits, which lowers the traffic consumption for cache downloads as compared to a uniform distribution. Nevertheless, in general the choice of the distribution does not raise or lower the benefits, that can be achieved by a fixed number of entries in a cache.

5.2 Requirement fulfillment

Requirement 1: Input NotaryCache defines the fields as input, which are used to identify the entry containing the target host. In contrast to existing approaches, not only the hostname, port or ip address can be given as an input, but also the algorithm of the public key, that is given in the server certificate, can further be used. That enables NotaryCache to also make use of future cryptographic algorithms and not yet widely distributed algorithms, such as ECC. Moreover, NotaryCache approaches the IP address and the hostname in different ways, making them less dependent on each other. Thus, they can be used in TLS SNI, which has a high deployment yet. Moreover, DNS spoofing attacks can be detected, for example by gathering the domain for an IP using reverse DNS and checking, if the IP resolves to the correct domain.

Requirement 2 If NotaryCache receives a request, it does simply response with the identified entry. The identified entry contains all information about that host, that are stored on server side, for example, IP address, hostname, port, key algorithm or the certificate chain. This allows the client to verify not only its received certificates, but also the hostname or resolved IP address in case of censorship mechanisms, that make use of the domain name system.

Moreover, NotaryCache is build to store arbitrary information about the identified host in its cache. This feature can be used to store the routing information to the target host, which can then be used by the client to determine, for example, the list of caches for evaluation.

Requirement 3 NotaryCache allows a fully automated infrastructure, which doesn't need any enduser interaction. By adding essential service information, the caches can be used for automated and efficient service discovery, which simplifies notary management at client side in contrast to existing notaries. Also, service discovery can be done asynchronously with a list of hosts, for example from browser history.

Moreover, automation enables the client to autonomously make decisions, whether to trust a given target hosts' certificate. Here, the ultimate goal must be, that every error message is fatal, which leads to the problem, that an attack must be detected accurately. NotaryCache enables the client to calculate probabilities based on a list of caches to indicate the correctness of faultiness of a given certificate. This mathematical description thus leads to accurate decisions without relying purely on trust mechanisms from the WebPKI or any enduser decisions.

Requirement 4 In NotaryCache, adequacy heavily relies upon the number of caches, which are investigated during target host evaluation. This aspect also directly results from the concept of multi-path notaries. However, in contrast to existing notaries, no traffic is generated to evaluate the certificates of a target host, if suitable caches were downloaded in advance. Also, due to the accurate identification of target hosts in caches, the adequacy of the results is also enhanced.

Requirement 5 NotaryCache implements mechanisms to ensure the integrity of the cache, if transferred over unencrypted HTTP connections. However, to receive the public key, that is needed for verification, the existing TLS infrastructure is used. The TLS certificates for these central NotaryCache deployments can be checked using preinstalled caches, certificate pinning or other measurements. As long as the measurements for these hosts succeed, NotaryCache is able to extend itself, as further TLS certificates can be checked using existing caches. Thus, additional measures are only needed for the first caches. Those could, for example, be deployed by central players of the internet, such as the Internet Society or Mozilla.

Requirement 6 It was shown in the last section, that NotaryCache heavily relies on the distribution of requests from users. If this problem could be solved, NotaryCache could improve scalability of existing notaries many times over. Moreover, NotaryCache introduces the roles of Cache Replicants, which can be used similar to content delivery networks to distribute the network load across the internet and usually near the clients location. This enables both fast download on client side and less traffic and network load on the server side. When thinking of big corporate networks or university networks, central fileshares in these networks could also be used to save traffic.

Requirement 7 The default notary, that is implemented in NotaryCache, is configured to gather the certificates of a target host shortly after the request was received. If other notary deployments are integrated, then NotaryCache could be used to delay requests to notaries in order to fulfill this requirement.

Requirement 8 NotaryCache does not expose any security-sensitive information, which can be used for an attack on the target host. Every certificate is only stored as a digest, so that an attacker could not find any certificates containing weak keys using NotaryCache. Also, NotaryCache does not keep track of target hosts, which do not offer any encryption HTTPS connections and thus do not differentiate between closed ports and unencrypted HTTP ports.

Requirement 9 Although the caches are distributed to clients as files, NotaryCache relies on a high availability. To successfully evaluate caches, the client must connect to NotaryCache's configuration. Since TLS is a hard requirement for this connection, it is difficult to integrate content distribution networks to act as support in case of downtime. Availability must also be given to issue live requests to NotaryCache.

However, in contrast to existing notaries, NotaryCache bears small outages without any problems, as traffic at NotaryCache deployments should not be as high as directly at notary deployments.

Requirement 10 NotaryCache does minimize the risks of a server operator to a minimum by implementing an event based architecture, which also includes hardware monitoring to react on changes in CPU or network load. Moreover, NotaryCache runs without administrative permission, nor does it require any additional infrastructure or frameworks besides the Java framework, which could also be delivered together with NotaryCache to administrators, who do not want to install Java system-wide. This independence enables system administrators to further restrict NotaryCache's traffic consumption.

Requirement 11, 12 NotaryCache includes mechanisms for hardware monitoring as well as mechanisms for adaption to different scenarios, such as increasing system load. NotaryCache clients also include location determination based on the given IP using location databases. This information is used during target host evaluation.

NotaryCache does not directly address risk in its context. Nevertheless, it would be possible to determine a minimum and maximum number of caches based on the endusers risk level. This would at least ensure, that certificates are evaluated from a minimum number of different places worldwide.

Requirement 13 NotaryCache introduces three standardized interfaces using the .well-known-registry, which is defined in RFC5785. This registration assures a conflict-free location, where NotaryCache can be accessed. Moreover, by giving a standardized protocol, a standardized and comprehensive interface to arbitrary notaries could also be a benefit for a widespread deployment, as a client doesn't need to communicate using a specific notary protocol anymore. Thus, arbitrary notaries could be incorporated in the notary infrastructure with less effort than before.

Requirement 14 NotaryCache is designed with respect to privacy. Thus, the cache does not contain any information, which could identify individual users. It also lies in the nature of the cache, that the operator of a NotaryCache deployment is neither able to determine the number of requests for a target host, nor is he able to see all target hosts of one client, as, in the optimal case, most requests would be covered by the cache. An operator of a NotaryCache deployment could only see the live requests. However, at client side, randomization of caches could be implemented, such that not every request would be send to the same NotaryCache deployment.

Requirements 15, 16 Because NotaryCache is designed to work without any user interaction, neither ease of use with the core product, nor usefulness of the core product are harmed. Also, NotaryCache must only be deployed in addition to existing measures like certificate path evaluation and certificate validation checking.

Requirement 17 A list of caches enables the client to simply lookup certificate digests offline, hence increasing the efficiency and minimizing the latency of target host evaluation. Moreover, caches and configurations can be obtained asynchronously from different sources, such as USB pen drives, content delivery networks or browser/operation system updates. However, if a target host could not be found in the caches, than live requests must be issued, which leads to the same latency issues, that were already found for existing approaches.

Requirement 18 Expert knowledge is needed in order to get to know about the problem domain and to get to know NotaryCache. Besides that issue, installation and maintenance of NotaryCache is simple, as it is delivered as a ZIP-file, which must only be extracted. Starting works by simply starting the bash- or cmd-file, which is delivered together with NotaryCache. No other actions are needed. The concept of NotaryCache could also be implemented as a web server module or as a component of the operating system.

Requirements 19, 20 The architecture of NotaryCache supports various business cases. The main business cases are given by monitoring capabilities offered by the Cache Monitor role, which behaves similar to the monitoring capabilities of Certificate Transparency. Here, the generic business case could be, that business operators could monitor a huge number of caches to notify its customers of faulty certificates for their domain. This service could be offered by CAs, as they are issuing certificates and do have an interest in certificate management practices, which are sold to customers to improve their certificate management. Another use case could be the detection of fraudulent CAs by browser vendors or public authorities.

6 Conclusion

It was shown, that existing notary implementations have various issues, which prevented a widespread deployment yet. Not only the technical challenges were investigated, but moreover the socio-technical challenges and the requirements from ubiquitous computing and security economics were reviewed. The examination revealed significant issues regarding enduser inclusion as well as missing economic incentives. NotaryCache was developed to provide valuable services to increase the number of notary deployments. NotaryCache adds a new layer of abstraction, which can be used by application developers to include arbitrary notaries. By downloading caches asynchronously to the actual requests and by shifting the evaluation to the client, additional delay can be avoided, which is a strong requirement of enduser-related software. Moreover, service discovery and bootstrapping can be done fully automatically without any enduser interaction. That also supports the proposal of various researchers, not to include endusers in security related decisions. Also, the installation and maintenance of the server component of NotaryCache is easy. NotaryCache also gives incentives to participants of the WebPKI by adding monitoring and replication capabilities. For example, a CA could implement monitoring functionality, which is sold as additional services to customers. In the evaluation, it was shown, that NotaryCache can lower the number of requests to notaries with a small amount of entries.

It was also shown, that initialization of NotaryCache is essential to its security. That means, that software vendors have to come up with a concept of how to integrate NotaryCache in their product and how to initialize NotaryCache securely. Also, NotaryCache could overburden small computers, as traffic and storage consumption increases due to the download of additional caches. Nevertheless, the basic problem, server operators and software developers have to be aware of the current issues of the WebPKI in order to implement NotaryCaches or notaries in common, persists. To approach this issue, the client component of NotaryCache is delivered as Java 8 library, which could easily be integration in existing Java software.

Moreover, the evaluation revealed various issues with NotaryCache, which must be investigated in future work. The first issue is the determination of the parameter α for the Zipf distribution, which describes the relation from requests to target hosts. The evaluation showed, that it makes huge differences, if $\alpha = 0.7$ or $\alpha = 1.0$. In general, one could say, that the lower α, the higher the number of entries in the cache to achieve a targeted benefit.

Moreover, the evaluation illustrated the problem, that there are currently no mechanisms to decide, whether to download a given cache or not. That leads to the problem, that caches are downloaded, which could be fully useless to the client, because the cache doesn't contain the hosts, that the client accesses. From the endusers perspective, the cache would not provide any benefit. Its download rather produces a huge amount of traffic, that could otherwise be saved with a suitable strategy.

7 Summary

This thesis discussed the various drawbacks of multi-path notaries related to todays WebPKI. First, an introduction to the WebPKI and its drawbacks was given to show the current issues of internet security. Then, multi-path notaries were introduced and it was shown, how multi-path notaries could solve those issues. However, todays notaries have different drawbacks and practical issues, which result in the fact, that notaries are not widely deployed yet.

To overcome these issues, NotaryCache was developed based on various requirements from computer science and ubiquitous computing, as well as from socio-technical and economical engineering. Central elements of NotaryCache are caching structures, which contain information about hosts, that were recently requested from notaries. These cache structures are to be downloaded by clients to evaluate received certificates offline without any interaction with the notaries. Moreover, a strategy for this evaluation was given based on the attacker model. In this strategy, only caches from NotaryCache deployments are chosen to evaluate a host, which were not created in the same country as the target host.

After sketching the proof-of-concept implementation, an evaluation of NotaryCache was done to determine the storage and traffic consumption for an individual user as well as for the global scenario. It was shown, that NotaryCache could improve the deployment of notaries and could significantly lower the number of requests to notaries. However, due to the missing of important distributions, only a rough estimation of the optimal cache size could be given. Also, an evaluation based on the requirements, that were defined beforehand, was done, to show the improvements to existing issues.

8 Glossary

(End-) User The enduser is the entity who has the intention to establish a connection to a target server, usually because of the contents of the website the server returns. The enduser relies on the correctness of the certificates as he is interested in the authenticity, integrity and confidentiality of the connection and the returned contents. In the context of the thesis, the enduser is assumed to be unreliable and is such not able to make security-related decisions. RFC 3647 describes the enduser as the relying party “who acts in reliance on that certificate and/or any digital signatures verified using that certificate”.

Client The client is the application which is used by an enduser to connect to a given target server. It therefore must support essential protocols like HTTP and TLS. The client possesses a dedicated root trust store or uses the trust store offered by the operating system, which it uses to evaluate trust to deduced root certificates. Usually the client is a web browser like Mozilla Firefox or Google Chrome.

(Root) Trust Store The root trust store is the place within an application or operating system, which is used to store trust information about root CA certificates. If a root CA certificate is stored in this trust store, it is assumed to be trusted.

Certificate Authority A Certificate Authority usually consists of two entities: The certificate authority responsible for the issuance of certificates and in possession of the private key and the registration authority which is responsible for approving or denying certificate requests from server operators. This makes more actions possible like identification, authorization, revocation, and so on. In this thesis, certificate authority comprises both authorities for simplicity reasons.

(Target) Server The target server which receives the HTTP requests from the client initiated by the enduser, executes them and returns contents back to the client. The target server must be able to communicate via TLS. Otherwise no connection via TLS is possible. A target server comprises of one or more virtual hosts and is identified by one or more IP addresses. The target server may also be accessible by a domain, when using the Domain Name System. During the TLS session, the concrete host is identified by the SNI extension as described above.

(Server) Operator The server operator describes one or more people responsible for the operation of the target server. They may have economical and legal incentives to operate the server or additional services.

Bibliography

[1] Marc Abrams, Charles R Standridge, Ghaleb Abdulla, Edward A Fox, and Stephen Williams. Removal policies in network caches for world-wide web documents. In *Acm sigcomm computer communication review*. Volume 26. (4). ACM, 1996, pages 293–305.

[2] Lada A Adamic and Bernardo A Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002.

[3] George A Akerlof. The market for"lemons": quality uncertainty and the market mechanism. *The quarterly journal of economics*:488–500, 1970.

[4] Devdatta Akhawe, Bernhard Amann, Matthias Vallentin, and Robin Sommer. Here's my cert, so trust me, maybe?: understanding tls errors on the web. In *Proceedings of the 22nd international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2013, pages 59–70.

[5] Devdatta Akhawe and Adrienne Porter Felt. Alice in warningland: a large-scale field study of browser security warning effectiveness. In *Usenix security*, 2013, pages 257–272.

[6] Nadhem J AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob CN Schuldt. On the security of rc4 in tls. In *Usenix security*, 2013, pages 305–320.

[7] Mansoor Alicherry and Angelos D Keromytis. Doublecheck: multi-path verification against man-in-the-middle attacks. In *Computers and communications, 2009. iscc 2009. ieee symposium on*. IEEE, 2009, pages 557–563.

[8] Bernhard Amann, Robin Sommer, Matthias Vallentin, and Seth Hall. No Attack Necessary: The Surprising Dynamics of SSL Trust Relationships. In *Proceedings of the 29th annual computer security applications conference*. ACM, 2013, pages 179–188. DOI: `10.1145/2523649.2523665`. URL: `syncii:///NoAttackNecessaryTheSurpr.webarchive`.

[9] Bernhard Amann, Matthias Vallentin, Seth Hall, and Robin Sommer. Extracting certificates from live traffic: a near real-time ssl notary service. Technical report. Technical Report TR-12-014, ICSI, 2012.

[10] Bernhard Amann, Matthias Vallentin, Seth Hall, and Robin Sommer. Revisiting ssl: a large-scale study of the internet's most trusted protocol. Technical report. Technical report, ICSI, 2012.

[11] Ross Anderson. Why information security is hard-an economic perspective. In *Computer security applications conference, 2001. acsac 2001. proceedings 17th annual*. IEEE, 2001, pages 358–365.

[12] Ross Anderson and Tyler Moore. Information security economics–and beyond. In, *Advances in cryptology-crypto 2007*, pages 68–91. Springer, 2007.

[13] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Dns security introduction and requirements. RFC (4033). `http://www.rfc-editor.org/rfc/rfc4033.txt`. RFC Editor, 2005. URL: `http://www.rfc-editor.org/rfc/rfc4033.txt`.

[14] Hadi Asghari, Michel Van Eeten, Axel Arnbak, and Nico Van Eijk. Security economics in the https value chain. *Available at ssrn 2277806*, 2013.

[15] James Ball, Julian Borger, and Glenn Greenwald. Revealed: how us and uk spy agencies defeat internet privacy and security. *The guardian*, 6, 2013.

[16] Adam Bates, Joe Pletcher, Tyler Nichols, Braden Hollembaek, and Kevin RB Butler. Forced perspectives: evaluating an ssl trust enhancement at scale. In *Proceedings of the 2014 conference on internet measurement conference*. ACM, 2014, pages 503–510.

[17] Tony Bates, Philip Smith, and Geoff Huston. Cidr report for 10 may 15. `http://www.cidr-report.org/as2.0/`. [Online; accessed 10-May-2015]. 2015.

[18] Gordon Baxter and Ian Sommerville. Socio-technical systems: from design methods to systems engineering. *Interacting with computers*, 23(1):4–17, 2011.

[19] Kevin Benton, Juyeon Jo, and Yoohwan Kim. Signaturecheck: a protocol to detect man-in-the-middle attack in ssl. In *Proceedings of the seventh annual workshop on cyber security and information intelligence research*. ACM, 2011, page 60.

[20] Johannes Braun and Gregor Rynkowski. The potential of individualized trusted root stores: minimizing the attack surface in the light of ca failures. *Iacr cryptology eprint archive*, 2013:275, 2013.

[21] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: evidence and implications. In *Infocom'99. eighteenth annual joint conference of the ieee computer and communications societies. proceedings. ieee*. Volume 1. IEEE, 1999, pages 126–134.

[22] Michael Coates. Revoking trust in two turktrust certificates. `https://blog.mozilla.org/security/2013/01/03/revoking-trust-in-two-turktrust-certficates/`. [Online; accessed 09-Fabruary-2015]. 2013.

[23] Comodo. Comodo fraud incident, mar. 2011. `https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`. [Online; accessed 09-Fabruary-2015]. 2011.

[24] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC (5280). `http://www.rfc-editor.org/rfc/rfc5280.txt`. RFC Editor, 2008. URL: `http://www.rfc-editor.org/rfc/rfc5280.txt`.

[25] Cristiano Andre da Costa, Adenauer C Yamin, and Claudio Fernando Resin Geyer. Toward a general software infrastructure for ubiquitous computing. *Ieee pervasive computing*, 7(1):64–73, 2008.

[26] Fred D Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *Mis quarterly*:319–340, 1989.

[27] Antoine Delignat-Lavaud, Martın Abadi, Andrew Birrell, Ilya Mironov, Ted Wobber, and Yinglian Xie. Web pki: closing the gap between guidelines and practices. In *Proceedings of the 21st annual network and distributed system security symposium*, 2014.

[28] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC (5246). `http://www.rfc-editor.org/rfc/rfc5246.txt`. RFC Editor, 2008. URL: `http://www.rfc-editor.org/rfc/rfc5246.txt`.

[29] Zakir Durumeric, David Adrian, James Kasten, Drew Springall, Michael Bailey, and J. Alex Halderman. Poodle attack and sslv3 deployment. `https://zmap.io/sslv3/`. [Online; accessed 28-April-2015]. 2014.

[30] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. Analysis of the https certificate ecosystem. In *Proceedings of the 2013 conference on internet measurement conference*. ACM, 2013, pages 291–304.

[31] D. Eastlake. Transport layer security (tls) extensions: extension definitions. RFC (6066). `http://www.rfc-editor.org/rfc/rfc6066.txt`. RFC Editor, 2011. URL: `http://www.rfc-editor.org/rfc/rfc6066.txt`.

[32] Peter Eckersley and Jesse Burns. The (decentralized) ssl observatory. In *Invited talk at 20th usenix security symposium*, 2011.

[33] Kai Engert. DetectTor. `http://detector.io/DetecTor.html`. [Online; accessed 09-Fabruary-2015]. 2013.

[34] Chris Evans, Chris Palmer, and Ryan Sleevi. Public key pinning extension for http. Internet-Draft (draft-ietf-websec-key-pinning-21). `http://www.ietf.org/internet-drafts/draft-ietf-websec-key-pinning-21.txt`. IETF Secretariat, 2014. URL: `http://www.ietf.org/internet-drafts/draft-ietf-websec-key-pinning-21.txt`.

[35] Sascha Fahl, Marian Harbach, Henning Perl, Markus Koetter, and Matthew Smith. Rethinking ssl development in an appified world. In *Proceedings of the 2013 acm sigsac conference on computer & communications security*. ACM, 2013, pages 49–60.

[36] Kurt Geihs, Jan Marco Leimeister, Alexander Roßnagel, and Ludger Schmidt. On socio-technical enablers for ubiquitous computing applications. In *Applications and the internet (saint), 2012 ieee/ipsj 12th international symposium on*. IEEE, 2012, pages 405–408.

[37] Google. Issue 65533: no method to check thumbprint of certificate received from website. `https://code.google.com/p/chromium/issues/detail?id=65533`. [Online; accessed 11-May-2015]. 2015.

[38] Alexandra C Grant. Search for trust: an analysis and comparison of ca system alternatives and enhancements, 2012.

[39] John Michael Hammersley and David Christopher Handscomb. *Monte carlo methods*. Volume 1. Methuen London, 1964.

[40] Cormac Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the 2009 workshop on new security paradigms workshop*. ACM, 2009, pages 133–144.

[41] J. Hodges, C. Jackson, and A. Barth. Http strict transport security (hsts). RFC (6797). `http://www.rfc-editor.org/rfc/rfc6797.txt`. RFC Editor, 2012. URL: `http://www.rfc-editor.org/rfc/rfc6797.txt`.

[42] P. Hoffman and J. Schlyter. The dns-based authentication of named entities (dane) transport layer security (tls) protocol: tlsa. RFC (6698). `http://www.rfc-editor.org/rfc/rfc6698.txt`. RFC Editor, 2012. URL: `http://www.rfc-editor.org/rfc/rfc6698.txt`.

[43] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. The ssl landscape: a thorough analysis of the x. 509 pki using active and passive measurements. In *Proceedings of the 2011 acm sigcomm conference on internet measurement conference*. ACM, 2011, pages 427–444.

[44] Ralph Holz, Thomas Riedmaier, Nils Kammenhuber, and Georg Carle. X. 509 forensics: detecting and localising the ssl/tls men-in-the-middle. In, *Computer security–esorics 2012*, pages 217–234. Springer, 2012.

[45] Lin Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. Analyzing forged ssl certificates in the wild. In *Security and privacy (sp), 2014 ieee symposium on*. IEEE, 2014, pages 83–97.

[46] A. Langley. Revocation checking and Chrome's CRL. `https://www.imperialviolet.org/2012/02/05/crlsets.html`. [Online; accessed 23-Fabruary-2015]. 2012.

[47] Adam Langley. Playing with the Certificate Transparency pilot log. `https://www.imperialviolet.org/2013/08/01/ctpilot.html`. [Online; accessed 09-Fabruary-2014]. 2013.

[48] Ben Laurie. Certificate transparency. *Queue*, 12(8):10, 2014.

[49] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, Tao Wan, and Jianping Wu. When https meets cdn: a case of authentication in delegated service. In *Security and privacy (sp), 2014 ieee symposium on*. IEEE, 2014, pages 67–82.

[50] Netcraft Ltd. SSL Survey. `http://www.netcraft.com/internet-data-mining/ssl-survey/`. [Online; accessed 02-Fabruary-2015]. 2013.

[51] William R Marczak, John Scott-Railton, Morgan MARQUISBOIRE, and Vern Paxson. When governments hack opponents: a look at actors and technology. In *Proceedings of the 23rd usenix security symposium*, 2014.

[52] Moxie Marlinspike. An agile, distributed and secure strategy for replacing Certificate Authorities. `http://convergence.io/`. [Online; accessed 03-Fabruary-2014]. 2011.

[53] James A Muir and Paul C Van Oorschot. Internet geolocation: evasion and counterevasion. *Acm computing surveys (csur)*, 42(1):4, 2009.

[54] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.

[55] Eila Niemelä and Juhani Latvakoski. Survey of requirements and solutions for ubiquitous software. In *Proceedings of the 3rd international conference on mobile and ubiquitous multimedia*. ACM, 2004, pages 71–78.

[56] M. Nottingham and E. Hammer-Lahav. Defining well-known uniform resource identifiers (uris). RFC (5785). `http://www.rfc-editor.org/rfc/rfc5785.txt`. RFC Editor, 2010. URL: `http://www.rfc-editor.org/rfc/rfc5785.txt`.

[57] Mark O'Neill, Scott Ruoti, Kent Seamons, and Daniel Zappala. Tls proxies: friend or foe? *Arxiv preprint arxiv:1407.7146*, 2014.

[58] ZEIT ONLINE. Selbst ssl-verschlüsselung ist nicht vor nsa-spionage sicher. `http://www.zeit.de/digital/datenschutz/2013-09/nsa-gchq-private-internet-verschluesselung`. [Online; accessed 28-April-2015]. 2013.

[59] Maarten Ottens, Maarten Franssen, Peter Kroes, and Ibo Poel. Systems engineering of socio-technical systems. In *Incose international symposium*. Volume 15. (1). Wiley Online Library, 2005, pages 1122–1130.

[60] Henning Perl, Sascha Fahl, and Matthew Smith. You won't be needing these any more: on removing unused certificates from trust stores. In, *Financial cryptography and data security*, pages 307–315. Springer, 2014.

[61] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. Ip geolocation databases: unreliable? *Acm sigcomm computer communication review*, 41(2):53–56, 2011.

[62] JR Prins and Business Unit Cybercrime. Diginotar certificate authority breach'operation black tulip'. 2011.

[63] Eric Rescorla. *Ssl and tls: designing and building secure systems*. Volume 1. Addison-Wesley Reading, 2001.

[64] Steven B Roosa and Stephen Schultze. The "certificate authority" trust model for ssl: a defective foundation for encrypted web traffic and a legal quagmire. *Intellectual property & technology law journal*, 22(11):3, 2010.

[65] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 internet public key infrastructure online certificate status protocol - ocsp. RFC (6960). `http://www.rfc-editor.org/rfc/rfc6960.txt`. RFC Editor, 2013. URL: `http://www.rfc-editor.org/rfc/rfc6960.txt`.

[66] Ryan Singel. Law Enforcement Appliance Subverts SSL. `http://www.wired.com/2010/03/packet-forensics`. [Online; accessed 05-Fabruary-2015]. 2010.

[67] Christopher Soghoian and Sid Stamm. Certified lies: detecting and defeating government interception attacks against ssl (short paper). In, *Financial cryptography and data security*, pages 250–259. Springer, 2012.

[68] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying wolf: an empirical study of ssl warning effectiveness. In *Usenix security symposium*, 2009, pages 399–416.

[69] EL Trist and KW Bamforth. Some social and psychological consequences of the longwall method. *Human relations*, 4(3):3–38, 1951.

[70] International Telecommunication Union. Ict facts and figures - the world in 2014. `http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx`. [Online; accessed 28-April-2015]. 2014.

[71] Nevena Vratonjic, Julien Freudiger, Vincent Bindschaedler, and Jean-Pierre Hubaux. The inconvenient truth about web certificates. In, *Economics of information security and privacy iii*, pages 79–117. Springer, 2013.

[72] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the acm*, 36(7):75–84, 1993.

[73] D Wendtandt and D G Andersen. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing, in *Proceedings of the usenix 2008 annual technical conference (atc 2008), usenix association, berkeley, ca*, 2008.

[74] Alma Whitten and J Doug Tygar. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *Usenix security*. Volume 1999, 1999.

[75] K. Wilson. Distrusting new cnnic certificates. `https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/`. [Online; accessed 28-April-2015]. 2015.

[76] Barbara H Wixom and Peter A Todd. A theoretical integration of user satisfaction and technology acceptance. *Information systems research*, 16(1):85–102, 2005.

[77] Kim Zetter. Saudi telecom sought us researcher's help in spying on mobile users. `http://www.wired.co.uk/news/archive/2013-05/15/saudi-telecom-sought-spy-help`. [Online; accessed 28-April-2015]. 2013.

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 19. Mai 2015

(Fabian Letzkus)