Privacy Threats in the Mobile Web & Social Media

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Informatik und Angewandte Kognitionswissenschaft
der Universität Duisburg-Essen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften

genehmigte Dissertation

von
Lorenz Schwittmann
aus
Krefeld

Gutachter: Prof. Dr.-Ing. Torben Weis
Gutachter: Prof. Dr. Arno Wacker

Tag der mündlichen Prüfung: 24.05.2019

DOI: 10.17185/duepublico/70228
URN: urn:nbn:de:hbz:464-20190625-105440-3

Abstract

In this dissertation, we investigate selected privacy threats in the mobile web and present means to mitigate them. The mobile web does not mitigate preexisting privacy threats of the web. We therefore initially analyze the security of domain validation processes performed by Certificate Authorities (CAs). Vulnerabilities in this process can be used to obtain an illegitimate certificate suitable for attacking HTTPS connections. We present a methodology to probe CAs for weaknesses without jeopardizing productive systems and apply it to a large portion of commercial CAs. We show that it is possible for a network-level attacker to obtain such certificates despite availability of technical measures suited to prevent this. We provide suggestions for both CAs and domain owners to mitigate such attacks.

A secure connection alone is not sufficient to guarantee privacy as transmitted data can be processed arbitrarily by the remote server. We discuss this privacy threat for online social networks (OSNs) which caused privacy debates due to accumulation of large amounts of private information at profit-oriented companies. A survey of decentralized OSN systems which aim to mitigate privacy issues is provided. Based on this, we present a privacy preserving OSN featuring end-to-end encryption and social graph obfuscation. As this hides large potions of (meta-)data from the provider, the potential for privacy breaches is substantially reduced.

The mobile web introduced several additional APIs allowing unrestricted access to certain sensors without notifying the user. We discuss privacy threats arising from this in the last part of this work. First, we present an approach for identifying videos being played back in the proximity of the user. The approach relies on ambient light sensor readings perceiving minimal illumination changes caused by the playback device. As these changes can be correlated to reference signals extracted from source videos, an identification is possible. In our evaluation we demonstrate feasibility as accuracy is above 93% for professional productions. The second sensor-related privacy threat covered in this work consists of disclosing the user's location. We present an approach to covertly measuring the sun's position which can be used to determine the user's position using astronomic calculations. Our evaluation shows it is possible to achieve an accuracy of up to 146 km. For both sensor-based threats we discuss mitigation techniques and propose countermeasures.

Zusammenfassung

In dieser Dissertation werden Privatsphärengefährdungen bei der mobilen Nutzung des Webs untersucht und Möglichkeiten aufgezeigt, diese zu beseitigen. Zu Beginn wird die Sicherheit von Zertifizierungsstellen (CAs) bei der Domainvalidierung untersucht. Schwachstellen bei dieser erlauben es Angreifern in Besitz von Zertifikaten fremder Domains zu gelangen, welche genutzt werden können, um die Sicherheit einer HTTPS-Verbindung anzugreifen. Es wird eine Messmethodik vorgestellt, welche es erlaubt, Zertifizierungsstellen auf solche Schwachstellen zu untersuchen. Experimentell wird nachgewiesen, dass es einem Angreifer möglich ist, illegitim Zertifikate zu erhalten, obwohl es technische Maßnahmen gäbe, die dies verhindern könnten. Es werden konkrete Schritte vorgeschlagen, welche sowohl CAs als auch Domainbesitzer umsetzen können, um diese Art von Angriffen zu verhindern.

Auch wenn die Verbindung zu einem Webserver gesichert ist besteht die Möglichkeit, dass bei diesem die übertragenen Daten unzulässig verwendet werden. In dieser Arbeit wird dies anhand des Anwendungsfalls der sozialen Netzwerke behandelt. Gerade die Ansammlung großer Mengen privater Daten bei einem Anbieter birgt ein Missbrauchspotential. Es wird daher ein Überblick über alternative Ansätze gegeben, welche die Privatsphäre der Nutzer besser schützen können sollen. Darauf aufbauend wird im Detail ein eigener Ansatz eines verteilten sozialen Netzwerkes mit Ende-zu-Ende Verschlüsselung beschrieben. Da der Anbieter bei diesem keinen Zugriff auf die Klartextdaten hat und der soziale Graph in Teilen verschleiert wird, reduziert sich die Gefahr eines Datenmissbrauchs erheblich.

Die mobile Nutzung des Webs geht einher mit der Möglichkeit des unbeschränkten Zugriffes auf bestimmte Sensoren von Mobilgeräten ohne Wahrnehmung des Nutzers. Im letzten Themengebiet werden die damit einhergehenden Risiken aufgezeigt. Dazu wird zunächst ein Verfahren vorgestellt, mittels dem Videoaufnahmen identifiziert werden können, die in der Nähe des Nutzers abgespielt werden. Dies ist durch den Umgebungslichtsensor möglich, welcher geringe Beleuchtungsunterschiede messen kann. Diese Unterschiede wiederum können Referenzsignalen zugeordnet werden. Es wird gezeigt, dass diese Zuordnung bei professionellen Videoproduktionen in über 93% der Fälle korrekt erfolgen kann. Abschließend wird untersucht, inwiefern der Nutzer über Sensoren geortet werden kann, die über Webapplikationen ausgelesen werden können. Durch Bestimmung des Sonnenstands ist es mittels astronomischer Berechnungen möglich, die Position der Beobachtung zu folgern. Es wird gezeigt, dass abhängig vom Nutzerverhalten eine Bestimmung der Position mit einer Genauigkeit von bis zu 146 km möglich ist.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advent of the mobile web has fundamentally changed the way large parts of the population interact with information. Smartphones provide constant access to news, weather reports, digital multimedia encyclopedias and a broad variety of entertainment. The wide shift towards a mobile web can be seen in two numbers: In 2018, 20% of U.S. adults accessed the Internet primarily via their smartphone without having a broadband service at home [93] while in Asia the mobile portion of Internet traffic exceeded 68%. [15] That increase of mobile access affects the way the web evolves as a platform for applications. Specific web APIs were introduced to utilize features unique to mobile devices such as battery levels, various sensors, vibration feedback, touch events or device orientation.

Concurrently, the role of user-generated content intensified. In 2017, the growth of content in the web originated in large parts from online social networks with Facebook as the largest of these platforms. The spread of users between these services is unbalanced with a trend to concentrate on very few large platforms. [125] This causes personalized data to accumulate at large cooperations.

While this centralization has sparked privacy debates, in combination with the mobile web it extends classical web privacy threats with new dimensions. This thesis illustrates these threats with selected examples and shows mitigation techniques.

The body of this work starts with a focus on privacy-providing architectures. In **Chapter 3**, we analyze security properties of Certificate Authorities during domain validation. We present a passive measurement method that allows us to check CAs for a list of technical weaknesses during their domain validation procedures. Our results show that all tested CAs are vulnerable in one or even multiple ways, because they rely on a combination of insecure protocols like DNS and HTTP and do not implement

existing secure alternatives like DNSSEC and TLS. We verified our results through experiments and disclosed these vulnerabilities to CAs. Based upon our findings, we provide recommendations to domain owners and CAs to close this fundamental weakness in web security. This work has been presented at the *4th IEEE European Symposium on Security and Privacy* [106] in June 2019.

Even if a secure connection has been established between client and server, privacy properties depend on the provided service as a whole. In Online Social Networks for example personal data is stored on the premise of the service provider and customers have to trust the provider to utilize this in an ethical way. In **Chapter 4**, we therefore survey a broad range of distributed online social networks regarding their privacy properties. This survey has been published in the *IEEE Internet Computing* magazine [109] in March 2014.

In **Chapter 5**, we describe one of the federated online social networks mentioned in Chapter 4 in detail. All user content is encrypted and decrypted on end-user devices, hiding the content from the OSN providers. The social graph is hidden from the OSN provider by employing a novel aliasing approach and using secure algorithms for mutual friendship establishment. Usernames are mapped to friend-specific aliases, which reduces the amount of information a provider can gather from analyzing these identifiers. Users authenticate to each other without revealing their identities to a potential attacker. The proposed system allows for user interactions between independent OSN providers. To improve data availability we use a replication scheme which does not jeopardize the obfuscation of the social graph. This work has been presented at the *International Workshop on Hot Topics in Peer-to-peer computing and Online Social neTworking* [108] in 2013.

The second part of this work focuses on privacy issues stemming from sensors exposed to web APIs. **Chapter 6** shows how an ambient light sensor can be used to infer which TV channel or on-demand video a smart device owner is currently watching. A video playing on a television screen emits a characteristic flickering, which serves as an identification feature for the video. We present a method for video recognition by sampling the ambient light sensor of a smartphone or wearable computer. The evaluation shows that, given a set of known videos, a recognition rate of up to 100% is possible by sampling a sequence of 15 to 120 seconds length. Our method works even if the device has no direct line of sight to the television screen, since ambient light reflected from walls is sufficient. A major factor of influence on the recognition is the number of video cuts that change the light emitted by the screen.

The work in that chapter is based on a best paper candidate presentation on the *IEEE International Conference on Pervasive Computing and Communications* [105] in March 2016 and on an invited extended version published in the *Pervasive and Mobile Computing* journal [107] in July 2017.

In **Chapter 7** we describe an approach which uses ambient light sensor, magnetometer, gyroscope and accelerometer to turn a mobile device into a digital sextant. We show that these sensors are sufficient to determine the rough geographical location of a user by turning the mobile device into a digital sextant. As these sensors do not require extended permissions they can be read without the user's awareness. Despite low-quality sensor data, our approach is able to determine the position of the sun and thereby the geographical area where the user is located. Our approach works even if the user holding the device does not cooperate in being located or employs location-disguising techniques such as a VPN. We analyze in detail the different error sources and show in which settings and situations our approach works best. The location accuracy was at best 146 km with a medium accuracy below 500 km. Truncating the positional sensor readings minimizes the privacy threat, while truncation of the ambient light sensor has almost no effect. This work has been presented on the *International Conference on Information Systems Security and Privacy* [110] in February 2019.

In **Chapter 8**, we conclude this work with a summary of results and provide an outlook for future developments that are based on the findings of this thesis.

Chapter 2

Fundamentals

We begin this work with some basics insofar as as they are necessary for understanding the following chapters.

2.1 Security Goals

The National Institute of Standards and Technology (NIST) defines information security as "the protection of information and systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability." [88] This definition refers to confidentiality, integrity and availability as the objectives of protection mechanisms which are consequentially called *security goals*:

Confidentiality guarantees that information is available only to those authorized to obtain it.

Integrity provides certainty that only authorized data changes are performed.

Availability assures that a system actually performs services to authorized entities.

These three goals are also called *CIA triad* [118], emphasizing their fundamental nature. However, these goals are not complete in being insufficient to express all desired security properties of a system. They are commonly complemented by the following goals:

Authenticity allows to prove genuineness, applicable to both data and entities.

Accountability assures that previous action by an entity cannot be denied retroactively.

A *threat* is defined as the potential violation of one or more security goals caused by a series of events. An *attack* is an actual realization of this potential risk. [101]

2.2 Privacy

Depending on the context it is used in, *privacy* has different definitions. In cryptography privacy is synonymous to confidentiality and viewed as a property of an encryption scheme. [81] As an example let us assume an adversary with control over an insecure channel wants to obtain some transmitted information. As the sender is aware of this threat the data is encrypted before being transmitted. Privacy is breached if the attacker is still able to decipher that information.

In context of the Internet there is a strict distinction between the concepts of confidentiality and privacy. While the Internet Security Glossary defines (data) confidentiality analogous, privacy is explicitly described as "the right of an entity (normally a person) [...] to determine [...] the degree to which the entity is willing to share information about itself with others." [114] As this is a right of an entity, the term of *data privacy* is congruously deprecated as misleading. A similar definition is given by the International Telecommunication Union (ITU) though it is supplemented with an acknowledgment of its inherent impreciseness due to referring to the right of an individual. [121]

Privacy is also its own field of research in statistical databases where information about a population are to be disclosed while at the same time the privacy of individuals should be protected. A fundamental privacy requirement for such systems was phrased by [33] as the inability to learn anything about an individual by having access to such a database. While intuitively appealing this requirement also mandates to preserve the privacy of non-participants of that database, even if there is auxiliary information available. For example, let us assume a person is unwilling to share her body height but she is known to be 10 cm larger than the average human being. In this case access to a database yielding the average human height will breach that person's privacy — regardless of being actually included in that data set. In a formalized form this strict requirement from 1977 was therefore proven to be unsatisfiable if the database is to provide any utility. Instead the notion of *differential privacy* has been

proposed. [43] It requires that the output of a statistical database should not allow an adversary to definitely conclude whether an individual is included in the data set or not. This approach is implemented by adding certain amounts of noise to outputs. The magnitude of this noise is a trade-off between utility and privacy. Although the concept of differential privacy is too database specific to apply it to Internet users we will come back to this trade-off in chapter 6.

In the scope of this thesis we adopt the privacy definition from the Internet Security Glossary [114]. Consequentially, a privacy violation is the distribution of personal information without explicit or implicit consent of the user. By this definition a cryptographic confidentiality breach is also a privacy violation as an adversary gains information without the user's awareness. Consent is crucial to this definition and not the extent to which information are disseminated. Publicly sharing one's exact location is therefore not covered by this definition whereas deduction of the user's rough position by a company would be considered a privacy breach — even though from an objective point of view the quality of information and distribution magnitude is larger. Although this definition bears impreciseness, as noted by the ITU, it will suffice for the scope of this work. For comprehensibility, assumptions about the user's intentions will be disclosed throughout the text.

2.3 Internet Public Key Infrastructure

To transmit a message securely over an insecure channel various encryption ciphers can be used. All require that there is some exchanged key material available to both sender and receiver. As this material cannot be transfered in plain text over the insecure channel several approaches have been proposed to solve this *key exchange problem*.

In the Internet, Public Key Infrastructures (PKI) are commonly used to address this issue. Trust is conveyed by a Certificate Authority (CA) which guarantees for participants' identities. This is realized using asymmetric cryptography with the CA's public key distributed to all participants. Trust is transitive in this model: CAs can delegate their power – with or without restrictions – to another entity called *Subordinate CA* in this context.

```
Certificate  ::=  SEQUENCE  {
    tbsCertificate       TBSCertificate,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValue       BIT STRING
}
TBSCertificate  ::=  SEQUENCE  {
    version         [0]  EXPLICIT Version DEFAULT v1,
    serialNumber         CertificateSerialNumber,
    signature            AlgorithmIdentifier,
    issuer               Name,
    validity             Validity,
    subject              Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                         -- If present, version MUST be v2 or v3
    subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                         -- If present, version MUST be v2 or v3
    extensions      [3]  EXPLICIT Extensions OPTIONAL
                         -- If present, version MUST be v3
}
```

Figure 2.1: X.509 v3 certificate structure according to [27].

2.3.1 X.509 Certificates

Cryptographic certificates extend the concept of asymmetric cryptography by binding information regarding an entity to a public key. X.509 [27] defines the certificate format used predominantly in the Internet (Fig. 2.1).

The binding of an entity to a public key takes place via `subject` containing its name and `subjectPublicKeyInfo` set to the cryptosystem-depending public key. For example, when certificates are used to identify a web server `subject` contains the host name[1]. The entity declared in the `issuer` field signed the whole `TBSCertificate` with the resulting signature stored in `signatureValue`. This construction causes the actual binding; any modification of name or public key will invalidate the signature.

To validate a certificate signature the relying party requires the issuer's certificate. As this certificate itself can be issued by another entity a *certificate chain* has to be traversed up to a well-known trusted certificate which is signed by itself. If all signatures in this chain are valid, the current date is within the `validity` attribute

[1]For reasons of brevity extensions – especially *Subject Alternative Name* – are not discussed here.

of all certificates and potential constraints caused by `extensions` are met, then the validation succeeds.

Possession of such a certificate chain alone is not sufficient to provide authenticity since it can be copied trivially. Protocols relying on X.509 certificates therefore include some interactive proof involving the private key corresponding to the end entity's certificate. For example, in the Transport Layer Security Protocol version 1.3 (TLS, [99]) this is implemented during handshaking by mandating a signature over all messages exchanged previously. As this includes high entropy random numbers by both parties this proof is strictly tied to the current TLS session. The common term *owning* or obtaining a certificate therefore always includes possession of the associated private key.

2.4 The Mobile Web

From a user's point of view the mobile web is characterized by interacting with web pages via a web browser on a mobile device. These web pages are downloaded from a remote web server via the Hypertext Transfer Protocol (HTTP).

2.4.1 TLS

For a sender, the path of its IP packets traversing the Internet is unpredictable since routing tables are subject to constant changes. Unknown entities pose therefore a threat to the confidentiality of transmitted data. While this challenge preexisted in stationary environments, mobile devices are unique in their varying Internet access points. Especially public Wi-Fi networks allow adversaries to eavesdrop or temper with traffic.

To protect web traffic, HTTP requests are usually secured by a TLS session wrapping the connection. TLS has been designed to provide the security goals confidentiality, integrity and authenticity. The latter is implemented by presenting an X.509 certificate to the client during handshake. As this certificate is signed directly or indirectly, i.e., using a certificate chain, by a trusted certificate authority (CA) the client assumes the owner of this certificate and its associated private key to be the authentic remote host. During the handshake symmetric algorithms and session keys are negotiated to be used for actual payload data. Such a *hybrid cryptosystem* has the advantage of combining the convenience of a public-key scheme with the performance

of a symmetric one. [103]

2.4.2 WebAPI

Nowadays most web pages do not only contain text and images but require the browser to run interpreted ECMAScript code. Brunelle et al. [18] reported that 54.5% of examined web pages required ECMAScript to load embedded resources emphasizing that scripting has become a fundamental feature for web pages to function. Scripting is not limited to loading embedded resources but allows to access local *WebAPI*s provided by the browser. Features of these APIs include manipulation of the page's Document Object Model (DOM), performing dynamic HTTP requests to servers, rendering 3D scenes or storage of local data. With these features the web became a platform for applications which were previously only developed for desktop environments.

Likewise *Device API*s were defined to expose mobile device specific features to browsers, enabling usage of the web as a platform for mobile applications. These APIs allow to access the ambient light sensor [65], read battery status [67], obtain proximity sensor values [11] or determine geolocation [95] and screen orientation [22]. Similarly to desktop environments, these APIs are not sufficient to implement arbitrary mobile applications as web applications but allow to utilize the web's cross-platform properties for certain use cases.

As these sensors reveal more information than in a stationary environment — especially due to mobile-specific usage patterns — there is a higher potential for privacy breaches. Access to data deemed privacy relevant such as revealing the position via GPS or providing access to microphone or camera, is therefore limited by browsers. If a website requires these information users are prompted to confirm access.

2.5 Online Social Networks

The phenomenon of Online Social Networks (OSNs) can be traced back to the middle of the 2000s when several websites with *social* features started to reach a general public. These features are, according to a popular definition by boyd and Ellison from 2007 [34], the possibility for users 1. to create profiles, 2. add references to other users and 3. reach other user's profiles by following such references. References between profiles are called *friends*, *followers*, or simply *contacts*, depending on the actual OSN implementation.

This definition has a strong focus on mapping and interacting with the social network but does not mention the actual communication between users which is a main incentive for users on OSNs according to [63]. Depending on the actual OSN, users can communicate via chat, instant messages, photo- or video-sharing, blogs, commenting profiles or videoconferencing. [85, 34] Ellision and boyd provided an updated definition in 2017 [42] addressing that shortcoming. The focus shifted to user-supplied content which is available in profiles and presented to users as streams to interact with. While this definition does not include means of real-time communication it can be applied to nearly all asynchronous communication present in current OSNs.

In academic literature, OSNs are also referred to as *social networks*, *social networking*, *social networking sites* or *social network sites*. While all describe the same phenomenon, the term *social networks* is discouraged as it has a long history in social sciences describing human relations in general. Likewise the term *networking* is bearing a focus to a specific activity, rendering it unsuitable for a general term. [42]

The term *social network sites* is well suited to describe currently deployed large platforms. Such a website contains the profiles and links of all users registered on it with all user-generated content. As this has sparked privacy concerns of aggregating all information at a single entity we focus on distributed networks in this work. Therefore, we prefer the term online social networks over social network sites as it is not restrictive to a singular website.

Chapter 3

Domain Validation Security

The security of the WWW relies on cryptography and certificates, which are issued by certificate authorities (CAs). Security-aware users can take to their browsers to learn about the cryptography and key length involved in securing an HTTPS connection. However, the entire cryptography is pointless if the browser trusts in the wrong certificates. Even for a security-aware user it is difficult to judge whether a given certificate should be trusted or not. Therefore, browser vendors like Mozilla or Apple compile lists of CAs, which are considered to be trusted.

Trust in a CA is based upon their commitment to issue certificates to rightful domain owners only. There have been cases like Symantec [20] where a CA has been shown to issue certificates to unauthorized entities. CAs offer different domain validation (DV) procedures, which in turn rely on the security of other protocols and infrastructure like DNSSEC, DANE or HTTP. This poses the following research question: how secure is domain validation and what countermeasures are in use to fend off attackers? Another aspect is the appearance of Let's Encrypt as a new CA, which disrupted the market in 2015 by issuing certificates for free with a fully automated procedure. This raises the additional question whether Let's Encrypt is able to achieve a security level comparable to traditional CAs.

In this chapter, we design a passive measurement methodology that probes CAs for vulnerabilities in their different DV procedures. We analyze the DV issuance process, identify potential security mitigations and probe their existence passively while requesting a certificate from 15 CAs, which cover 96% of the certificate market. Our research shows that all major CAs expose weaknesses when validating whether a signing request was issued by the rightful domain owner or not. This is despite the fact that secure countermeasures already exist. CAs either do not employ all available

security measures or fail to implement them properly. We confirm the validity of our results by demonstrating successful man-in-the-middle attacks on three CAs for a test domain under our control.

Our method searches for indications of countermeasures in use; an absence of appropriate countermeasures thus yields vulnerabilities under our threat model. An attacker can exploit the detected vulnerabilities to falsely convince the CA of owning the domain, resulting in a certificate owned by an attacker which is trusted by all major browsers. Such a certificate can then be used in a man-in-the-middle attack to compromise the authenticity or encryption between browsers and legitimate web servers. For a network-level attacker, attacking the certificate issuance is much easier than breaking HTTPS encryption. This type of attack requires neither breaking cryptography nor a lot of computing power. Thus, the security of the web depends substantially on the domain validation practice.

The contributions of this chapter are: (1) a security analysis of domain validation, (2) based upon which we develop an approach for the detection and classification of domain validation weaknesses, (3) which we applied on major certificate authorities to get insights about their DV practices for the first time.

This chapter is organized as follows. Section 3.1 describes the certificate issuance process and our threat model. We elaborate on the domain validation methods, their security issues and potential countermeasures. In Section 3.2 we present our measurement methodology that detects these countermeasures and classifies vulnerability against attacker types. Section 3.3 lists the certificate authorities that we tested in practice with the results given in Section 3.4. We demonstrate practical attacks in Section 3.5. Section 3.6 describes the disclosure of results to the CAs and their responses. Section 3.7 discusses related work. Section 3.8 gives operational recommendations based on our findings.

3.1 Certificate Issuance

The process of certificate issuance begins with an *applicant* generating an asymmetric key pair. While the private key remains with him, the public key is bundled with the fully qualified domain name (FQDN) in a certificate signing request (CSR) and is sent to a CA. The CA checks whether this request is approved by the *domain owner*, who might be a different entity than the applicant. Domain Validation (DV) is the process of confirming whether the applicant has control over the domain name. Depending on

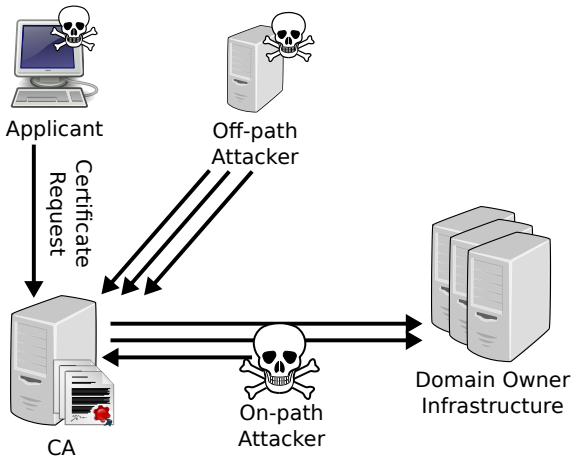


Figure 3.1: Threat model: an attacker requests a certificate for a domain that he does not own.

the validation method this involves passing a *challenge* defined by the CA. Once the validation has been completed successfully, the applicant receives a signed certificate.

The actual implementation of these steps depends on the CA and is usually not disclosed to the public. An exception is Let's Encrypt. To fully automate the whole certificate issuance process, the Automatic Certificate Management Environment (ACME) protocol is in the process of being specified in a public draft [7].

Besides DV there are the procedures of Organization Validation (OV) and Extended Validation (EV), which additionally verify and include the name of the domain owner's organization. EV certificates cause web browsers to prominently show that name in the address bar instead of the regular HTTPS indicator, e.g. a green padlock.

In this chapter we focus on DV only, as we assume the average user will not be able to tell the difference to an OV or EV certificate. This assumption is backed by a study by Jackson et al. who *"did not find that extended validation provided a significant advantage in identifying the phishing attacks tested in this study"* [62].

3.1.1 Threat Model

An attacker attempts to obtain a certificate for a domain name not possessed by him. The attacker acts as the applicant in the certificate issuance (Figure 3.1), whereas the legitimate domain owner does not intend to interact with the CA. The attacker interferes with the subsequent domain validation to trick the CA into believing that

the domain owner approves the certificate issuance.

We consider two types of network-level attacks: *off-path* and *on-path attacker*. Off-path attackers have the capability to spoof IP packets with a source address claiming to originate from the domain owner, but do not see the network traffic between the CA and the domain owner's servers. On-path attackers are capable of passive eavesdropping or performing an active man-in-the-middle attack. The validity of this threat model has been demonstrated by prior work and can be achieved, e.g., by redirecting network traffic via BGP attacks [13]. In any case, the attacker has the capability to choose the CA involved in the domain validation (*CA selection attack*), as he initiates the certificate issuance process.

3.1.2 Validation Methods

The *"Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates"* [21] specifies methods allowed for domain validation. It is published by the the CA/Browser Forum which is a consortium of CAs and browser vendors negotiating the set of minimal practices that a publicly trusted CA must employ. For example, the Mozilla Root Store Policy as of version 2.5 [84] explicitly requires CAs to employ a subset of the validation methods defined in the baseline requirements version 1.4.1.

The validation methods include out-of-band validation such as fax, SMS, phone, or postal mail, as well as contacting the domain name registrar. While these methods are valid ones, they are rarely used for domain validation in practice. As we will see in Section 3.3 none of the considered CAs offered them for domain validation. The following Internet-based validation methods are used in practice:

1. **DNS Change.** The CA generates a random token and instructs the applicant to publish it in the DNS zone file as a `TXT`, `CNAME` or `CAA` resource record.

2. **Agreed-Upon Change to Website.** The CA generates a random token and instructs the applicant to publish it under a specific URL within the domain.

3. **TLS Using a Random Number.** The CA generates a random token and instructs the applicant to generate a TLS certificate containing that value and serve it on that domain.

4. **Email to Domain Contact.** The CA sends a random token via email to

the email address stored in the domain's WHOIS record. The applicant has to submit this token, usually via a website.

5. **Constructed Email to Domain Contact.** Like (4), but the email address is constructed by using 'admin', 'administrator', 'webmaster', 'hostmaster' or 'postmaster' @ `domain`.

All methods include transfer of a random token whose actual implementation is CA-defined. The baseline requirements define this as either a randomly generated value of at least 112 bit entropy or as a request token cryptographically derived from the CSR.

To assess the attack resilience, we have a detailed look at each validation method and discuss potential weaknesses as well as mitigation strategies. An overview of the following discussion is shown in Table 3.1.

3.1.3 DNS-based Validation

If a CA offers this validation method, it typically prompts the applicant to add a specific `CNAME` or `TXT` record containing the token to the domain's zone file (cf. Figure 3.2a). After the applicant has made this change to the DNS zone, the CA queries the record using a DNS resolver to verify whether the applicant has control over the domain.

From an attacker's point of view this method is prone to DNS spoofing. Depending on the attacker's capabilities the response has to be spoofed with (on-path attacker) or without (off-path attacker) knowledge of the actual request. DNSSEC protects from both types of attackers, provided that the domain is signed.

If the domain is not signed with DNSSEC, there are several best practices and protocol extensions available to mitigate off-path spoofing by increasing the entropy in DNS requests: DNS Cookies [44], 0x20 encoding [32], source port number randomization [61] or TCP-based transport. Multiple, redundant queries with a consistency check can also be used to decrease likelihood of a successful attack. Off-path spoofing requires a massive amount of forged responses to match the guessed entropy in the request. Another mitigation strategy is to detect spoofing attempts by monitoring the number of incoming responses, thereupon triggering additional countermeasures or human intervention.

Apart from DNSSEC validation there are few effective mitigation strategies against on-path attackers. The ACME draft [7] suggests to send DNS queries from multiple

Table 3.1: Validation methods, attacks and associated countermeasures.

Validation Methods	Attack	Countermeasure
All	Off-Path DNS	Detect mass-spoofing Source port randomization DNS Cookies 0x20 encoding DNS via TCP DNSSEC DNS Multipath DNS Multiserver
All	On-Path DNS	DNSSEC DNS Multipath DNS Multiserver
HTTP	HTTP Active	HTTP Multipath DANE
TLS-RND	TLS Active	TLS Multipath DANE
Email	SMTP Eavesdrop	STARTTLS End-to-end encryption
	SMTP Active	DANE End-to-end encryption MTA-STS

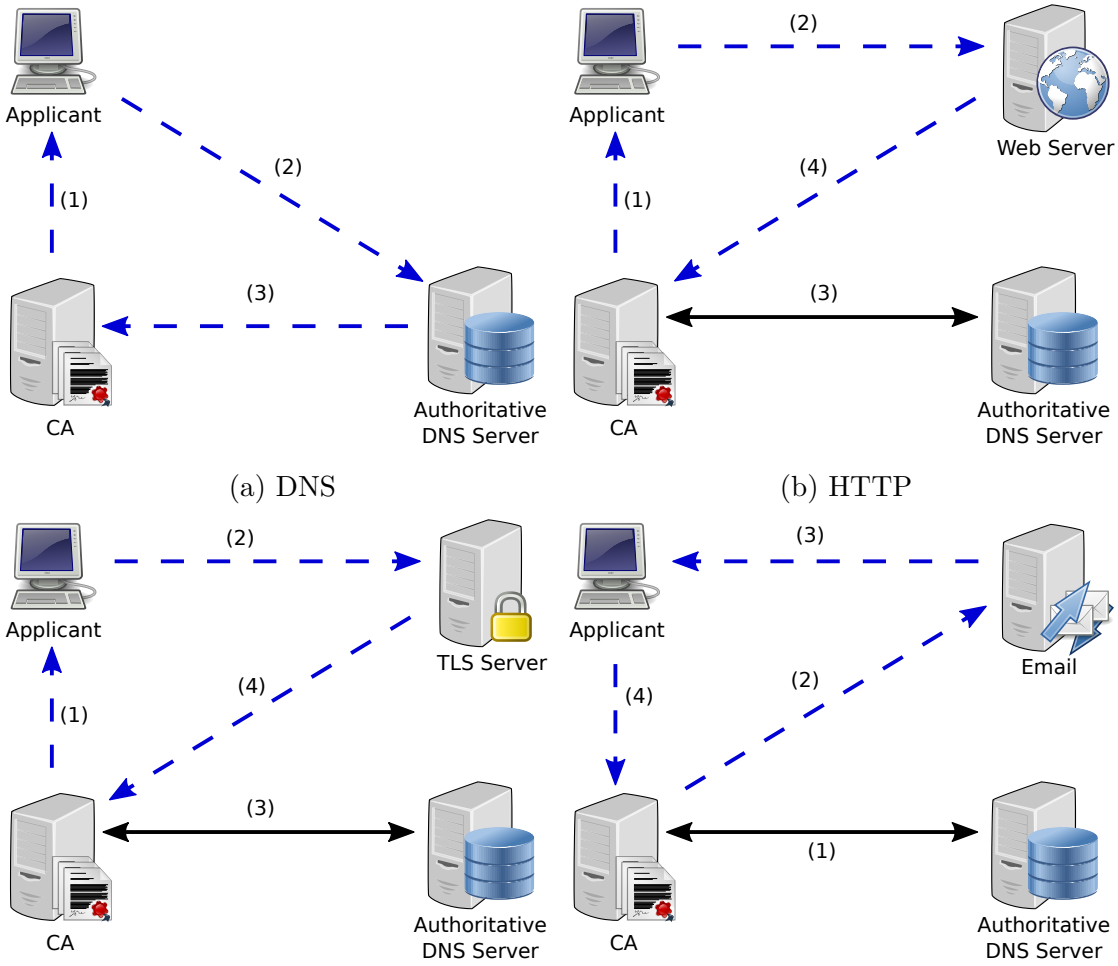


Figure 3.2: Flow of random tokens (dashed lines) and supporting DNS lookups (solid lines) for different validation methods.

vantage points (*multipath* queries). Similarly a CA could send redundant queries to all authoritative DNS servers for that domain (*multiserver* queries). The idea behind both strategies is that an on-path attacker has the capability to poison a few Internet paths between the CA and domain owner, but probably not every possible one. Sending redundant queries over diverse paths increases the likelihood to receive an untainted response, which exposes a potential spoofing attempt.

3.1.4 Web-based Validation

We sum up the HTTP-based (Agreed-Upon Change to Website) and TLS-RND-based (TLS Using a Random Number) methods as web-based validation because they have similar security properties.

3.1.4.1 HTTP

The CA asks the participant to place a token under a well-known URL with the applied-for domain name (cf. Figure 3.2b). Once the token has been placed, the CA verifies if the token is indeed in place. Before the HTTP request occurs, the CA has to resolve the domain name to obtain the web server's IP address by querying for `A` or `AAAA` DNS records. This is susceptible to the DNS attacks explained in the previous Section 3.1.3. If the attacker successfully spoofs a forged DNS response, he can redirect the CA to a web server under his control to pass the challenge. Thus, the DNS-specific considerations and countermeasures apply for the web-based validation as well.

The HTTP transaction provides an additional potential target for the attacker, because spoofing either the DNS or the HTTP response suffices to succeed. As the DNS and web server are typically located on different hosts, potentially in different networks, this constitutes another path for on-path spoofing. Again, the CA could employ redundant requests (*HTTP multipath*) to mitigate this vulnerability. A cryptographic proof of authenticity would be required to securely prevent man-in-the-middle attacks on HTTP. In the web context authenticity is usually provided by a trusted certificate—which a CA cannot expect to be available if the applicant is currently applying for one.

Independent of this bootstrapping problem, even if the CA attempts an HTTPS connection to the target web server, an on-path attacker can deny HTTPS availability and force a downgrade to cleartext HTTP. To avoid a downgrade attack, the

domain owner needs a secure channel to advertise the HTTPS capability along with his identity to the CA. DNS-based Authentication of Named Entities (DANE, [56]) provides such a measure by binding the domain name to a certificate or public key to be used in TLS connections. The certificate used in the HTTPS connection may be a self-signed certificate, if announced as such via DANE. Combined with DNSSEC this approach prevents man-in-the-middle attacks and downgrade attacks.

3.1.4.2 TLS-RND

Presenting a self signed certificate with a CA-defined random number is comparable to HTTP(S)-based validation (cf. Figure 3.2c). The main difference here is that after DNS resolution and performing a TLS handshake no further application protocol is used. Due to ongoing specification of ACME, there are two protocol variants for this approach called *TLS-SNI* and *TLS-ALPN*.

TLS-SNI is defined in the ACME draft up until version 9 [6]. It uses the Subject Alternative Name (SAN) field to encode the random token as a subdomain of the non-existing `acme.invalid.` zone (SAN A). Likewise, a CSR-specific key is encoded as a second alternative name (SAN B). After deployment of the certificate by the applicant, the CA initiates a TLS handshake with the Server Name Indication (SNI, [45]) set to SAN A and considers the challenge to be passed if a certificate with both SAN A and SAN B is presented.

This approach turned out to be exploitable on hosting providers which allow user-provided certificates to enable HTTPS on their websites[1]. As the Subject Alternative Names have no direct connection to the actual domain names, the providers could not enforce a strict domain separation. Users at the same provider could therefore pass TLS-SNI challenges for any other hosted domain. As a consequence TLS-SNI was deprecated and disabled in Let's Encrypt. TLS-ALPN [115] does not have this issue as the domain name to be validated is kept unchanged as Subject Alternative Name. Instead, the random token is encoded in a newly specified certificate extension. Additionally the Application Level Protocol Negotiation (ALPN, [46]) extension is used during TLS handshake with a fixed identification sequence—although no actual application data is sent over that TLS channel. This should avoid confusion about the semantics of these certificates and effectively make this validation method an opt-in option for hosting providers.

[1] `https://www.zdnet.com/article/lets-encrypt-disables-tls-sni-01-validation/`, Accessed 2018-06-27

Despite the protocol differences between TLS-SNI and TLS-ALPN our security considerations apply to both of them. As long as the certificate is not trusted—either by a valid CA or by DANE—the CA has no means to assert the authenticity of the domain owner.

3.1.5 Email-based Validation

The CA sends an email to the domain contact according to the WHOIS database or to an address constructed from the applied-for domain name (cf. Figure 3.2d). Some CAs allow the applicant to choose the email address, but only from a small set of addresses and never freely. The email contains a token, which the applicant has to submit on the CA website to prove control over the mailbox.

To send an email—disregarding whether the address is constructed or looked up—the CA has to query `MX` and `A/AAAA` DNS records to locate the mail server. This is susceptible to the DNS attacks mentioned in Section 3.1.3 and the countermeasures discussed there apply as well. If the attacker is able to redirect the SMTP connection to his server, he can easily intercept the token.

While all validation methods demonstrate control over the domain, email-based validation differs on a conceptual level as the applicant proves ability to read rather than write on the domain. The token thereby becomes a secret, as anyone who can obtain it is allowed to obtain certificates for this domain. This is not the case for the other validation methods where the challenge explicitly consists of publishing the token. By design email-based validation is the only method where a purely passive attacker can succeed by eavesdropping the SMTP connection. Thus, the email must be encrypted to prevent eavesdropping, and the applicant must not be allowed to choose the key as he is the attacker in our threat model.

One way to achieve this is to upgrade the cleartext SMTP connection to TLS with the STARTTLS extension. Similar to HTTP, a man-in-the-middle attacker can perform a downgrade attack by stripping the STARTTLS signals. Again, using DANE/DNSSEC with a self-signed certificate provides authenticity without CA involvement and prevents downgrade attacks on SMTP connections [38]. Mail Transfer Agent Strict Transport Security (MTA-STS) is another work-in-progress mechanism for DNS-based signaling of STARTTLS support for email domains [79]). Other than DANE, MTA-STS requires a CA-issued certificate for the mail server, which implies the applied-for domain either cannot use it or must rely on a third-party email

provider, which already has a certificate. MTA-STS does not require DNSSEC, which makes it easier to deploy but in this case also vulnerable to DNS-induced downgrade attacks.

Another way to hide the secret token is by using end-to-end encryption with S/MIME or OpenPGP. In this case the CA needs a secure way to look up the public key of the domain owner, who is not necessarily identical with the applicant. There are experimental DNS-based approaches to achieve this objective for both, S/MIME [57] and OpenPGP [130].

3.1.6 Additional Countermeasures

In addition to the security measures discussed above, there are additional countermeasures that are independent of the chosen domain validation method. While these are not direct countermeasures to the attack sketched in section 3.1.1, they are suited to mitigate it.

Certification Authority Authorization (CAA) [54] is a DNS record that lists the CAs that are permitted to issue certificates for a domain. The domain owner may optionally put such a record into his DNS zone file, thus prohibiting unauthorized CAs from certificate misissuance. Each CA is required to check the existence of the `CAA` record during domain validation according to the baseline requirements. This prevents the CA selection attack, where an adversary choses the CA with the least security measures or iterates through all CAs until the attack succeeds with one of them. As this method relies on the DNS, the attacks and countermeasures discussed for DNS-based domain validation do apply here as well.

Certificate Transparency (CT) [73] is an approach for publishing all certificates issued by trusted CAs in logs with cryptographic proofs of inclusion. Using Merkle Hash Tree the existence of a once-submitted certificate cannot be denied. The goal of this approach is to force CAs to publish every issued certificate as clients will refuse certificates that do not bear a log inclusion proof. When all valid certificates are visible to the public, misbehaving CAs and attacks can be discovered more easily.

Extended Validation (EV) includes a process to verify the domain owner's identity in addition to his control over the domain. Unlike plain DV, this involves human interaction which has the potential to discover an ongoing attack. Still, effectiveness depends on CA-specific realization of this process.

3.2 Approach

In this section we present our approach for finding vulnerabilities in CA domain validation practices in the wild. We acquire certificates for a domain under our control from a broad range of CAs. Our test setup is designed to complete the domain validation successfully while allowing us to observe the security measures that the CA employs. All network traffic is recorded to perform a post-mortem analysis after the certificate has been issued.

Design rationale. The rationale for a purely passive approach is to get a complete view of the whole domain validation process without endangering the reproducibility or validity of the results. Intentional misconfigurations or simulated attacks may raise suspicion and trigger human intervention, which would influence the outcome of further tests under the same domain and applicant identity. Even more important, the passive measurement and error-free setup bears no risk of interfering with the normal CA operations. This justifies to experiment with productive systems without prior consent from the CAs, which again might endanger the validity of the results. From a CA's point of view our experiments are regular certificate issuances.

Limitations. We assume optimistically that if there are indications for a security measure, then it has been implemented correctly. Thus we cannot rule out whether the measures in use are ineffective due to implementations faults. However, the absence of a security measure is a definite vulnerability according to our threat model.

3.2.1 Setup

Our test setup is shown in Figure 3.3. All servers run on one host and serve a newly registered second-level domain D. The authoritative DNS server accepts TCP and UDP-based queries. The domain is DNSSEC-signed using RSA/SHA-1 with a 1024-bit zone signing key (ZSK) signed by a 2048-bit key signing key (KSK). There are two name servers defined in the top-level zone ($ns1.D$, $ns2.D$), each with glue records for distinctive IPv4 and IPv6 addresses. Secure delegation takes place by `DS` records with both SHA-1 and SHA-256 digests of the KSK in the parent zone.

An `MX` record set up for the domain points to a mail server with STARTTLS support. `A` and `AAAA` records point to a web server that accepts both HTTP and HTTPS. The mail and web server use self-signed certificates, which are secured by DANE via `TLSA` records [56]. All servers are reachable via IPv4 and IPv6.

We did not configure MTA-STS, DNS Cookies or DANE-based end-to-end email

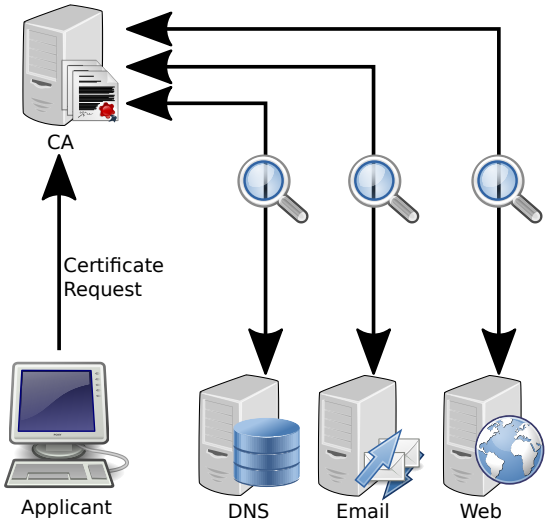


Figure 3.3: Measurement setup.

encryption, but our setup allows us to observe if the CA attempts to use it. Our DNS zone is arranged to ensure the CA will have to send the corresponding DNS queries even if they utilize aggressive negative caching [47], which is a DNS cache optimization that omits queries under certain conditions.

3.2.2 Detection of Countermeasures

We start capturing all network traffic in our setup before applying for a certificate. Additionally we retain all log files of involved services as these provide further insights. After passing the CA's challenge and obtaining the certificate we filter obviously unrelated traffic (vulnerability scanning by third parties, other scientific measurements) and automatically analyze the remaining data with a custom software. Analyzing this data consists of two steps: First we determine which countermeasures the CA uses and then we derive which attacks are possible under our threat model.

We classify each countermeasure from Table 3.1 as *fully*, *partially* or *not* implemented. As noted above not all measures can be detected passively with certainty. Criteria are defined as follows:

3.2.2.1 DNS

Some of the DNS countermeasures are clearly detectable from one DNS message, including DNS Cookies, 0x20 encoding and TCP transport. We consider them to be *fully* implemented if all relevant queries show them and *partially* if only some show

them. The set of relevant queries depends on the validation method. In some cases multiple DNS queries are necessary, which implies that all such queries are relevant and must be protected against spoofing.

Source port randomization, multipath and multiserver queries require an evaluation of more than one query. To accurately recognize source port randomization a large number of queries has to be observed. As these do not appear during a regular issuance we assume source port randomization to be implemented *fully* unless we find two DNS queries with identical source addresses and ports. Detecting multipath and multiserver queries is performed by grouping relevant queries by query name, class and type. If each group contains more than one distinct IP source (multipath) or destination (multiserver) address this mitigation is *fully* implemented. We classify it as *partially* implemented if only part of the group fulfills the requirement.

We define DNSSEC validation as *fully* implemented if the relevant DNS queries have the `DNSSEC OK` flag and a `DNSKEY` query has been observed within *TTL* seconds before or after that query. Otherwise we define DNSSEC validation as *not* implemented. If redundant queries are observed (same name, class and type) we consider the flag as set if at least one query has it set. This definition may result in a misclassification in favor of the CA if the CA uses multiple resolvers, of which only some are validating, or if the CA fetches the `DNSKEY` but fails to validate correctly.

The countermeasure of mass-spoofing detection cannot be observed passively and thus must be omitted from our analysis.

3.2.2.2 Web

HTTP multipath is considered to be implemented *fully* if we observe redundant requests for the same CA-defined URL from different source IP addresses. DANE support is indicated by a DNSSEC-secured query for `TLSA` under domain name `_443._tcp.`D, followed by an HTTPS request.

The time stamp of the last HTTP request allows to group DNS queries. As this definitely marks completion of the name lookup process DNS queries performed afterwards are unrelated to this. We therefore consider DNSSEC validation of the lookup part of HTTP-based validation to be *not* implemented if there is no `DNSKEY` query before the last HTTP request recorded.

For TLS-RND the same classifications apply.

3.2.2.3 Email

Countermeasures against SMTP attacks revolve around encryption. Whether the sender mail transfer agent requests STARTTLS is determined by analyzing our mail server logfile.

Additional countermeasures are determined by looking for DNS-based queries: DANE queries for `TLSA` under `_25._tcp.`D, MTA-STS for `TXT` under `_mta-sts.`D and end-to-end encryption by queries for `SMIMEA` under `_smimecert.`D or `OPENPGPKEY` under `_openpgpkey.`D. For each of these countermeasures DNSSEC validation is checked individually. Similar to HTTP-based validation,delivery of the email provides a time restriction. If a `DNSKEY` query is observed after the email has been sent it is unrelated to establishment and securing of the SMTP connection.

3.2.3 Attack Vulnerability

Based on the observed implementation state of each countermeasure, we classify each validation method as either *vulnerable* against an attack, *mitigated* or found *no vulnerability*. As a result of our classification of countermeasures, *vulnerable* implies that an attack is definitely feasible, because we demonstrated the absence of an appropriate countermeasure. *Mitigated* implies that the attack is potentially feasible, but countermeasures exist that might mitigate it. Found *no vulnerability* implies that we did not find evidence for a feasible attack.

If the CA implements DNSSEC validation then there is *no vulnerability* against DNS off-path attacks. We consider an off-path attack as *mitigated* if the CA implements at least one of the applicable countermeasures (source port randomization, DNS cookies, 0x20 encodig, TCP transport, multipath, multiserver). Otherwise, we consider the CA as *vulnerable* against DNS off-path attacks.

DNSSEC validation also protects against DNS on-path attacks, i.e. there is *no vulnerability*. Most other DNS countermeasures are ineffective against an on-path attack. Only DNS multipath and multiserver have the potential to *mitigate* it, otherwise the CA is *vulnerable*.

If DANE is used then the CA is *not vulnerable* to an active HTTP attack. Employment of HTTP multipath *mitigates* such an attack under certain circumstances. If neither is implemented, the CA is *vulnerable* to this attack. The same countermeasures are effective against an active TLS attacker, i.e. TLS multipath causes a *mitigated* and DANE a *not vulnerable* rating.

To be *not vulnerable* to an SMTP eavesdropper STARTTLS or end-to-end encryption has to be used. Otherwise the CA is *vulnerable* to this attack. If no countermeasures against an active SMTP attack are implemented the CA is *vulnerable* to this attack Usage of DANE, end-to-end encryption or MTA-STS causes a CA to be *not vulnerable*.

3.3 Tested Certificate Authorities

Table 3.2: List of tested CAs and their validation methods. Price is minimum of all validation methods (differences due to promotions and exchange rates).

CA	Tested Validation Methods	Trusted Root CA	Price
AlphaSSL	Email, DNS	GlobalSign	17 €
Amazon	Email, DNS	Starfield Technologies	0 €
Certum	Email, DNS, HTTP	Certum	15 €
Comodo	Email, DNS, HTTP	Comodo	0 €†
DigiCert	Email[1] with identity validation	DigiCert	148 €
GeoTrust	Email	GeoTrust	0 €†
GlobalSign	HTTP[2]	GlobalSign	107 €
GoDaddy	Email, DNS, HTTP	Go Daddy Group	54 €
Let's Encrypt	DNS, HTTP, TLS-SNI	IdenTrust	0 €
Network Solutions	Email	USERTRUST	71 €
RapidSSL	HTTP[3]	DigiCert	7 €
SSL.com	Email, DNS, HTTP	USERTRUST	41 €
Starfield Technologies	Email, DNS, HTTP	Starfield Technologies	51 €
StartCom	Email	–	0 €
Thawte	DNS, HTTP	DigiCert	30 €
Thawte	Email	Thawte	30 €

Further available validation methods: [1]HTTP, DNS; [2]DNS, Email; [3]Email
† Obtained free trusted trial certificate.

We selected a set of CAs issuing the most certificates on the market. As data sources we used [41] and W3Techs.com. StartCom is included although it recently lost trust by major browser vendors[2]. We attempted but could not test WoSign, because it targets the Chinese market and we were unable to provide one of the payment options supported by WoSign.

[2]`https://blog.mozilla.org/security/2016/10/24/distrusting-new-wosign-and-startcom-certificates/`, Accessed 2018-06-20

As we investigate the DV certificate issuance, we omitted CAs that provide OV or EV certificates only. In one case we inadvertently purchased an OV certificate, which we noticed only after the payment: DigiCert asked for a proof of personal identity and put the personal name as "Organization" attribute in the subject field. We leave DigiCert in the results, as it still provides insights about the domain validation practice. Note however that the applicant had to identify himself in this case, whereas this was not necessary for any of the DV certificates.

Table 3.2 lists the 15 CAs considered in our evaluation, their respective validation methods and the price paid. As of January 2018 this list covers 96.0% of all publicly trusted certificates used on Alexa's TOP 10 million websites[3]. We test all domain validation methods offered by each CA (except where noted). Therefore we need multiple domain names, one for each tested validation method. However, we can reuse the same domain name when acquiring certificates from different CAs, because CAs do not collate issued certificates with each other. The second-level domain names we used for the evaluation consist of two or three randomly chosen words from an English dictionary, registered under `.com` or `.net`.

As shown in Table 3.2, not every entity that we consider as CA issues certificates under a trusted root CA certificate with its name. While there are currently 152 root CA certificates in the Mozilla store[4], this number does not directly relate to the number of trusted CAs. On the one hand CAs use more than one trusted certificate for operational reasons (e.g. DigiCert alone owns 29 trusted root certificates), on the other hand there are companies which sell certificates without being present in root stores. The latter case is possible due to trusted intermediate CA certificates effectively granting that company CA capabilities. In case of Thawte the trusted root CA certificate surprisingly depends on the chosen validation method. We do not differentiate these cases but define a CA as an entity that issues certificates under its own brand.

Amazon is a special case, as it is the only CA that does not support the applicant to generate or retrieve the private key of the certificate. Instead, the private key is deployed on Amazon's TLS load balancers, which forward cleartext requests to cloud instances.

[3] `https://w3techs.com/technologies/overview/ssl_certificate/all`, Accessed 2018-01-29

[4] `https://wiki.mozilla.org/CA/Included_Certificates`, Accessed 2018-01-29

Table 3.3: Vulnerabilities found for DNS-based validation. Vulnerable (●), mitigated (◐), found no vulnerability (○).

CA	CAA		DNS	
	On-path	Off-path	On-path	Off-path
AlphaSSL	○	○	●	◐
Amazon	●	◐	◐	◐
Certum	○	○	●	◐
Comodo	○	○	○	○
GoDaddy	◐	◐	●	◐
Let's Encrypt	○	○	○	○
SSL.com	○	○	○	○
Starfield Technologies	◐	◐	●	◐
Thawte	○	○	◐	◐

Table 3.4: Vulnerabilities found for HTTP-based validation.

CA	CAA		DNS		HTTP
	On-path	Off-path	On-path	Off-path	Active
Certum	○	○	○	○	●
Comodo	○	○	●	◐	●
GlobalSign	○	○	◐	◐	●
GoDaddy	●	◐	○	○	●
Let's Encrypt	○	○	○	○	●
RapidSSL	○	○	○	○	●
SSL.com	○	○	○	○	◐
Starfield Technologies	●	◐	○	○	●
Thawte	○	○	○	○	●

3.4 Results

We tested the DNS-based, email-based validation in November and December 2017 and the HTTP-based validation in May 2018. The median time it took between submission of a certificate request till receipt of the signed certificate was 7:10 minutes (P_{25} = 4:21 min, P_{75} = 9:22 min).

We present the results of our security evaluation separately for each tested validation method: DNS-based validation in Table 3.3, web-based validation in Table 3.4 and

Table 3.5: Vulnerabilities found for TLS-SNI-based validation.

CA	CAA		DNS		TLS
	On-path	Off-path	On-path	Off-path	Active
Let's Encrypt	○	○	○	○	●

Table 3.6: Vulnerabilities found for email-based validation.

CA	CAA		DNS		SMTP		
	On-path	Off-path	On-path	Off-path	Passive	Active	TLS ver.
AlphaSSL	○	○	○	○	○	●	1.2
Amazon	●	◐	◐	◐	○	●	1.0
Certum	●	◐	●	◐	○	●	1.0
Comodo	○	○	○	○	○	○	1.2
DigiCert	○	○	○	○	○	●	1.2
GeoTrust	●	◐	●	◐	○	●	1.0
GoDaddy	●	◐	●	◐	○	●	1.2
Network Solutions	○	○	●	◐	○	●	1.2
SSL.com	○	○	●	◐	○	●	1.2
Starfield Technologies	●	◐	○	○	○	●	1.2
StartCom	●	◐	●	◐	●	●	none
Thawte	●	◐	●	◐	○	●	1.0

Table 3.5, email-based validation in Table 3.6. The results are broken down according to the classification from Section 3.2.3 as vulnerable (●), potentially mitigated (◐) or not vulnerable (○) against specific attack classes. Detailed lists of detected security measures are given in the appendix.

We consider vulnerability against DNS attacks also for the web and email-based validation, as DNS attacks suffice to undermine any validation method. One might assume the CA would achieve the same security rating against DNS attacks across all validation method, but that is not the case. For example, AlphaSSL, Certum, GoDaddy and Starfield Technologies use DNSSEC validation during HTTP or email-based validation, but strangely not during DNS-based domain validation.

3.4.1 CA Selection Attack

Similarly, we present vulnerabilities of the CAA lookup separately from other DNS lookups, even though CAA is basically a DNS-based mechanism. In most cases the CAA and other DNS ratings are identical, but there are a few discrepancies where CAA is more secure and even a few cases where CAA appears less secure than the other DNS lookups. As a positive note, all CAs perform a CAA lookup as required by the CA/B Forum[5] since September 2017. Despite being strongly recommended [54], not all CAs authenticate the CAA record via DNSSEC. These CAs are vulnerable to a CA selection attack by a DNS on-path attacker, and potentially vulnerable to

[5]https://cabforum.org/2017/03/08/ballot-187-make-caa-checking-mandatory/

off-path attackers.

3.4.2 On-path DNS Attack

To effectively protect from on-path DNS attacks, DNSSEC must be used. Several CAs have shown indications for DNSSEC validation according to our classification criteria, but not all. Without DNSSEC, redundant queries from multiple vantage points (*DNS multipath*) or to multiple authoritative DNS servers (*DNS multiserver*) have a chance to mitigate (◐) an on-path attack. Several CAs show indication for such a countermeasure: Amazon/DNS (two different IPv4 resolvers), Amazon/Email (18 unique IPv4 resolvers querying MX records), GlobalSign/HTTP (6 identical queries by the same GeoTrust IPv4 address, additional lookup via Google Public DNS), Thawte/DNS (usage of both IPv4 and IPv6). However, we cannot confirm with our passive methodology whether this is actually a security measure, or whether the redundant queries are due to operational reasons or functionality unrelated to the domain validation.

3.4.3 Off-path DNS Attack

Since an on-path attacker is more capable than an off-path attacker, all countermeasures against an on-path DNS attack apply to an off-path attack as well. Thus the off-path attack will exhibit at most the same vulnerabilities as the on-path attack, but not more than that.

In fact, all CAs seem to have appropriate mitigations against off-path attacks in place. None of the CAs showed indication for the lack of source port randomization. Some CAs had countermeasures beyond that in place, e.g. 0x20 encoding by Let's Encrypt and DNS Cookies by Amazon, RapidSSL and Thawte.

3.4.4 HTTP Attack

Preventing HTTP attacks requires opportunistic HTTPS and authentication with DANE. Although some CAs use DNSSEC, none attempted to query for our `TLSA` record during HTTP-based validation. Thus, every HTTP-based validation was vulnerable to an active man-in-the-middle attacker (Table 3.4).

For Certum we observed an anomaly as we were instructed to place our validation token v in a file under `/.well-known/pki-validation/`v`.html`. As *"the Request*

Token or Random Value MUST NOT appear in the request" [21, Section 3.2.2.4.6], this is a violation of the baseline requirements. A web server configured to echo the extension-less filename of every requested file would pass any such challenge.

Comodo and SSL.com allowed the applicant to specify whether HTTP or HTTPS should be used. This choice does not increase security, because the attacker posing as applicant would simply choose HTTP. We do not see a benefit of exposing such security-relevant option to the applicant, since a fall-back approach sketched in Section 3.1.4 provides the same flexibility with less potential for misuse.

A potential mitigation (◐) consists of performing multiple HTTP requests from different vantage points (*HTTP multipath*). We observed indications for this behavior for SSL.com, although it is unclear whether this is a security measure or an operational artifact.

Starfield Technologies queried three URLs one after another: first a file path that the applicant was not asked for to use[6] over HTTP, followed by HTTPS. Only then in a third HTTP request[7] the CA was able to obtain the requested token. As Starfield Technologies is a subsidiary of GoDaddy, this indicates a brand-unaware backend software trying multiple well-known paths until one succeeds.

3.4.5 TLS-SNI Attack

TLS-SNI (Table 3.5) has only been supported by Let's Encrypt. As Let's Encrypt does not use DANE, this leaves it vulnerable to man-in-the-middle attackers in the same way as HTTP. We tested TLS-SNI in November 2017 before TLS-ALPN was drafted. However, the protocol changes between TLS-SNI and TLS-ALPN do not affect the security assessment under our threat model.

3.4.6 SMTP Attack

Most CAs allow the applicant to choose a specific WHOIS or constructed email address. The set of constructed addresses was always restricted to the five well-known local parts. Amazon, DigiCert, Godaddy and Starfield Technologies sent separate emails to all five constructed addresses over separate SMTP connections. This might increase the attacker's chance to intercept the token, but it also increases the chance for the domain owner to notice an unauthorized certificate request.

[6] HTTP query path `/.well-known/pki-validation/godaddy.html`
[7] HTTP query path `/.well-known/pki-validation/starfield.html`

Email-based validation has the unique property of being vulnerable to passive attackers, which requires encryption to render this attack impossible. All CAs except StartCom used STARTTLS to upgrade SMTP to an encrypted connection. Thus only StartCom is vulnerable against passive attackers (Table 3.6). For informational purposes we also list the established TLS protocol version. Several CAs negotiated the obsolete TLS 1.0, which is not recommended due to security concerns [113].

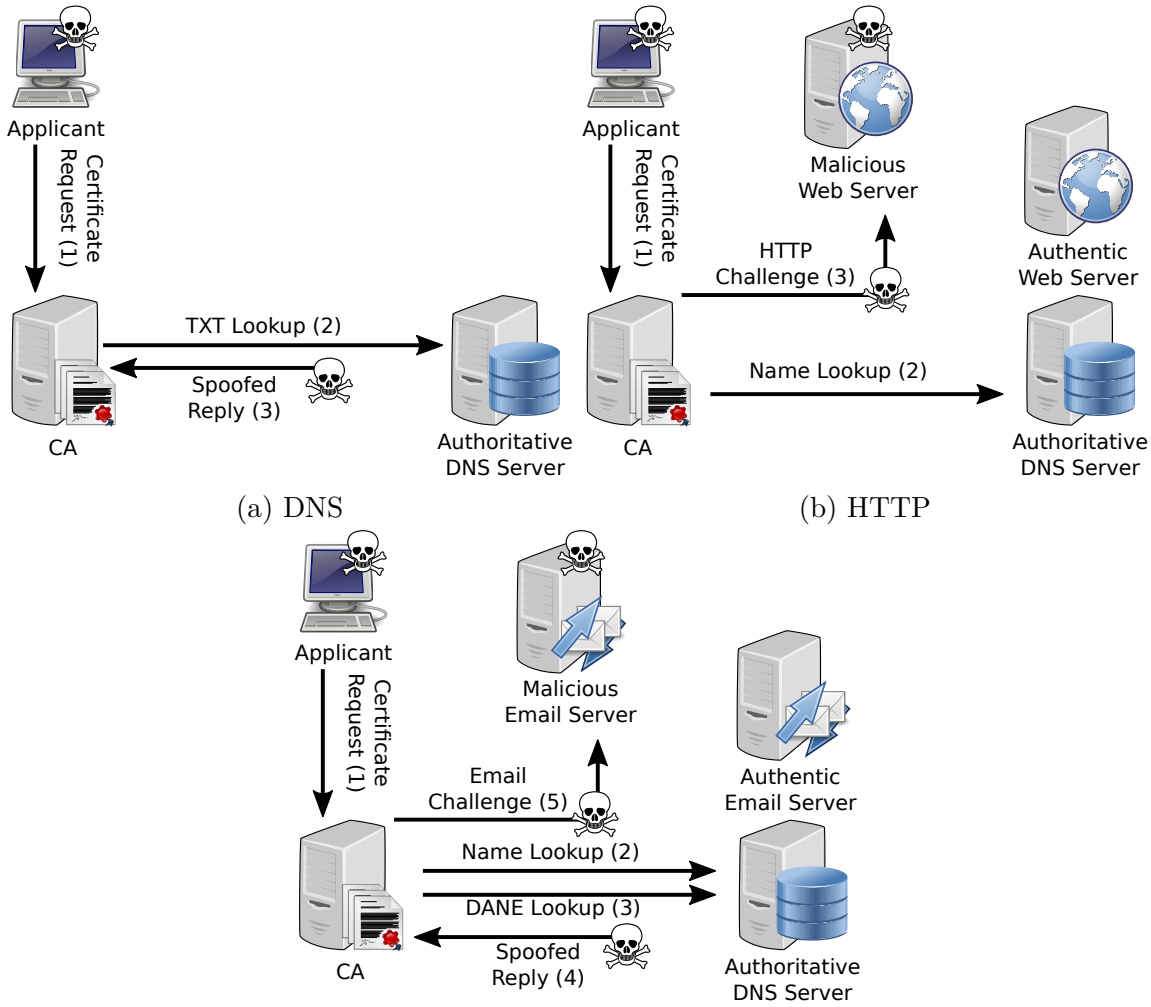
An active attacker could impersonate the destination mail transfer agent or deny STARTTLS capabilities to force a downgrade. Unlike with HTTP, we observed support for SMTP with DANE with Comodo, Network Solutions and SSL.com, which could prevent active attacks against SMTP. However, only Comodo queried the `DNSKEY` record in time before sending the mail, which is necessary for DNSSEC validation. Thus, although Network Solutions and SSL.com retrieved the `TLSA` record, the record is unusable due to a lack of validation [56] and the CA remains vulnerable. Other CAs did not use DANE at all and are thus vulnerable against active SMTP attackers.

We did not observe any support for MTA-STS nor support for end-to-end email encryption.

3.4.7 Discussion

The inconsistent security ratings raise the question of what causes the diverse results within a single CA. In case of the CAA mechanism, which became mandatory only recently, it makes sense to assume that best current practices are implemented while old processes remain unchanged. But as we have seen, in some cases the CAA validation was less secure than other DNS lookups. The inconsistency of security measures may be the result of a diligent security assessment, which leads under careful consideration of operational realities to diverse security requirements. Or they may be the result of technological legacies, grown infrastructures and ad-hoc implementations with the lack of an overall security strategy.

In a couple of cases the CA relies on Google Public DNS, which is a DNSSEC-validating resolver service. This is a useful addition to increase the number of vantage points, as long as the CA validates DNSSEC signatures additionally by itself. However, there are justifiable doubts about this assumption. Consider for example the following CAs, which showed DNSSEC indications only for a subset of validation methods: Certum (DNS and HTTP), GoDaddy (HTTP) and Starfield Technologies



(a) DNS (b) HTTP

(c) Email with DANE denial

Figure 3.4: Attacks performed on validation methods.

(HTTP and email). In each of these cases the CA relied on Google Public DNS for name resolution, but we never observed a `DNSKEY` query from the resolvers residing in the CA's network. A potential explanation why the other validation methods are not DNSSEC-protected is thus: the CA does not support DNSSEC validation and any indication for DNSSEC support is an artifact of the CA's decision to outsource part of their name lookups and blindly trusting Google Public DNS.

3.5 Experimental Validation

We have found vulnerabilities for all tested CAs using our conservative method. To verify our findings we attempt to obtain certificates by performing a network-level attack on our infrastructure while requesting certificates via different validation methods. We select GoDaddy/DNS, Thawte/HTTP and Network Solutions/Email for these attacks. For our setup we use a dedicated domain name and sign its zone with DNSSEC like a legitimate domain owner would do. DANE records for HTTPs and SMTP with STARTTLS are generated accordingly.

Figure 3.4 sketches the approaches. A malicious applicant requests a certificate from the CA (1) followed by a validation method-depending attack. For DNS-based validation (Figure 3.4a) we spoof responses to TXT queries using the packet sniffer/generator kamene[8] to complete the challenge. The original query is not modified, which causes the authentic response to eventually reach the CA as well and reveals that an attack has occured.

DNS queries of HTTP-based validation (Figure 3.4b) are not tampered with. Instead all HTTP traffic is tunneled to a malicious web server (3), which responds with the correct token to the CA's request.

As explained in Section 3.4.6, Network Solution performs DANE queries without DNSSEC validation during email-based validation. We exploit this vulnerability by tampering with DNS and SMTP (Figure 3.4c). The attacker actively denies existence of DANE records (4) and tunnels SMTP traffic to a malicious mail transfer agent (5) without STARTTLS support.

All attacks were performed in September 2018 and succeeded. For DNS-based validation we observed a singular TXT query which was spoofed accordingly. HTTP-based validation resulted in one HTTP-request for the actual domain and a second one for its `www` subdomain. In both cases the malicious web server served the validation token containing file. Email-based authentication caused a lookup for DANE records which was answered with a spoofed name error response. The subsequent SMTP connection was tunneled to the malicious mail transfer agent and the email was transmitted in plain text.

We obtained trusted certificates after these validation steps in all cases, i.e., all attacks were performed successfully. We did not observe additional countermeasures other than those already revealed by our passive measurement approach, which sub-

[8] `https://github.com/phaethon/kamene`

stantiates the validity of our method.

3.6 Disclosure of Results

We disclosed our findings directly to the CAs. We informed AlphaSSL, Comodo, Thawte, SSL.com and Starfield Technologies in April 2018 about inconsistent infrastructure behavior and Certum about its HTTP CA/Browser Forum Baseline Requirements violation. After a refinement of our method we informed the remaining CAs in July 2018 as they were vulnerable via at least one validation method. We reported the successful practical attacks from Section 3.5 to affected CAs in September 2018. Reactions greatly varied between CAs.

Starfield Technologies replied that DNSSEC was not mandated by the CA/B Forum Baseline Requirements and is therefore not supported.

Similarly Thawte stated that implementing DNSSEC was not a priority from a security point of view.

Let's Encrypt acknowledged the vulnerabilities, but justified that the DANE approach for securing HTTP and TLS-based validation is too complex. Instead Let's Encrypt favors the restriction of validation methods via the CAA record, which is currently under specification[9].

Certum acknowledged the baseline violation and reported that they deployed a correction in July 2018. Implementing DNSSEC validation was said to be under consideration.

GlobalSign acknowledged the vulnerabilities and announced a new infrastructure with DNSSEC support. They provided voucher codes for us to repeat the analysis. We were able to confirm consistent DNSSEC support for HTTP, email and DNS-based validation in August 2018 which eliminates all vulnerabilities except for active HTTP and SMTP attacks.

The remaining CAs made no factual statements.

3.7 Related Work

Scheitle et al. [102] surveyed the adoption of the CAA record and compliance to it. Compared to our work they examine the CAA mechanism only, but in greater depth

[9]https://tools.ietf.org/html/draft-ietf-acme-caa-05

as they used deliberately broken CAA configurations to uncover protocol violations. Although CAs did perform CAA validation, their implementations were flawed and they misissued certificates in corner cases.

Bhargavan et al. [10] formally modelled and verified the ACME protocol used by Let's Encrypt. They discovered a cross-CA attack possible with ACME, where one misbehaving CA forwards a certificate issuance request to another CA and succeeds. While this is a valid attack, we did not consider it in our threat model as the impact is moderate.

Borgolte et al. [16] demonstrated the problem with residual trust in domain names that point to unused IP addresses in the cloud. An attacker can grab the IP address and succeed with domain validation although the domain is not under his control. A similar problem are mistyped nameserver addresses and outdated WHOIS records [127], which an attacker can exploit to pretend control over a domain. This demonstrates the risk when the attacker can freely choose the validation method and when the CA does not support appropriate countermeasures to harden the validation.

Certificate studies. Various studies examine the certificates found in the wild [58], their trust relationship to intermediate and root CA certificates [41], and forged certificates encountered [60, 28]. By inspecting a large body of deployed certificates Delignat-Lavaud et al. [35] identified numerous violations of the baseline requirements in 2012–2013. Kumar et al. [69] followed up in 2017 and found that the percentage of misissued certificates decreased to 0.02%, but a long tail of small authorities still issued non-conformant certificates. They developed a certificate linter that checks for errors in certificates but not *"whether the destination domain was correctly validated"* [69], which is the research gap that we address in this chapter.

Certificate authority model. Arnbak et al. [3] surveyed the market share and price of DV (avg. $81), OV (avg. $258) and EV (avg. $622) certificates in 2013. They argue that the certificate market is driven by brand reputation or feature bundles, but not security. As the actual CA security is largely unobservable for the potential buyer, she has to make her decision based on the perception of security or other incentives. Our work sheds light on the domain validation practices and discloses weaknesses in that part of the system.

Matsumoto and Reischuk [80] suggested to incentivize CAs for careful identity validation by making them financially accountable. In case of a security incident, an insurance payout should be triggered automatically to the domain owner. Some

CAs like Comodo[10], Thawte[11] or GlobalSign[12] in fact offer a warranty bundled with a certificate. However, the security of the system depends on the weakest CA that persists in the browser trust stores, whose security mishaps are not covered by the warranty plan. Several approaches attempt to fix this structural flaw by rethinking the public-key infrastructure, either as addition to the existing CA model [119] or as fundamental alternative [135, 9, 8, 132, 31].

3.8 Recommendations

For every tested CA we found at least one attacker model that allows certificate misissuance under at least one domain validation method. One of the oddities in our results are varying DNS countermeasures subject to the validation method. However, secure domain name lookups are required for all Internet-based validation methods, including email and HTTP. The requirement for performing DNSSEC validation should be codified in the baseline requirements. This would provide domain owners with an opt-in way of enhancing security while at the same time maintaining compatibility with non-DNSSEC domains. As DNSSEC signing has not been adopted universally [128], CAs should consider using a combination of additional DNS mitigations listed in Section 3.1.3.
Recommendation: use DNSSEC signing (as domain owner) and DNSSEC validation (as CA).

While DNSSEC support is a necessary prerequisite to prevent attacks on domain validation, it is not sufficient. HTTP, TLS-RND and email-based validations require further measures to provide application layer security. The application layer protocol can benefit from using TLS, but requires a mechanism for downgrade resilience against active attackers. In case of email, Opportunistic DANE [38] prevents downgrade attacks on TLS-secured SMTP connections. DANE could be used in principle to secure HTTP or TLS-RND as well. However, the current ACME draft [7] mandates all HTTP-based validation to be performed without HTTPS due to concerns of improperly configured virtual hosts on shared web servers[13]. Similarly we observed CAs to allow applicants (including the attacker according to our threat model) to

[10] `https://www.instantssl.com/compare-ssl-certificates.html`, Accessed 2018-06-27
[11] `https://www.thawte.com/ssl/`, Accessed 2018-06-27
[12] `https://www.globalsign.com/en/ssl/compare-ssl-certificates/`, Accessed 2018-06-27
[13] `https://www.ietf.org/mail-archive/web/acme/current/msg00524.html`

choose whether HTTP validation should be performed using HTTP or HTTPS.
Recommendation: use a downgrade resilient signaling mechanism like DANE or CAA to chose secured validation channels when available.

Using `CAA` records reduces the potential for certificate misissuance. In its most simple form, the domain owner uses an empty `issue` property to lock the domain from certificate issuance or renewal when not needed. To protect from DNS spoofing, DNSSEC should be used. Only if all CAs performed DNSSEC validation, on-path attackers could be deterred effectively from obtaining illegitimate certificates. This would empower the domain owner to effectively control which CAs may issue certificates for her domain in the presence of attackers. Otherwise restricting a domain to a high-security CA will be moot, if an attacker is able to convince a less secure CA of a false CAA response.
Recommendation: use `CAA` records with DNSSEC.

Certificate authorities can utilize additional CAA parameters to allow restriction to a certain subset of domain validation methods. Combined with DNSSEC this achieves a downgrade-resistant signaling, preventing CA selection attacks as well as fallbacks to insecure protocols. This is especially important, because the validation methods have varying security properties.
Recommendation: use `CAA` records for authorization of the allowed domain validation methods.

3.9 Conclusion

Our results have shown that attacks on domain validation are within reach of a network-level attacker. All tested CAs proved to be vulnerable under our threat model via at least one validation method. In each of these cases a secure countermeasure exists already, but was not supported by the CA. The web-based validation in particular proved to be prone against man-in-the-middle attackers. In one case the CA violated the baseline requirements of the web-based validation. We showed experimentally that the domain validation vulnerabilities found in our analysis can actually be exploited.

Following up on our research question about the security of Let's Encrypt, we can conclude that its domain validation is at least as secure as that of traditional CAs. Let's Encrypt uses preventive security measures like DNSSEC where a couple

of other CAs do not. The HTTP and TLS-SNI validation methods are nevertheless vulnerable to man-in-the-middle attackers.

In general, a higher price for a certificate did not correlate with an increase in deployed security measures. For example, the GlobalSign certificate cost 107 €, but we observed less security measures than with Let's Encrypt, which is free for domain owners. This is however a purely technical view, as we did not consider additional buying incentives like bundled warranty, brand trust or logo availability.

Another core finding is that HTTPS is not enough as sole provider of web security. Before HTTPS can be called into action, DNSSEC is required to secure domain validation and obtain a certificate without the hazard of a man-in-the-middle attack. This applies to all Internet-based validation methods, as they all require a secure domain name lookup. On the other hand, setting up a domain with DNSSEC relies on HTTPS to interact securely with the domain name registrar. Ultimately, both systems complement each other and close their mutual security gaps that exist during setup.

Future work should follow up on our optimistic classification and test whether the indications for a security measure reflect that the measure is actually in use. This could be achieved with deliberate misconfigurations. Sending inconsistent DNS responses, invalid DNSSEC signatures or mismatching DANE records could provide further insights about the domain validation reality.

We did not consider wildcard certificates in our study and leave it for future work. Apart from domain validation, a security assessment of the extended validation processes is also of interest. Furthermore, browser vendors should assess whether the visual cues for extended validation are effective to convey its meaning to the user.

Chapter 4

Privacy Preservation in Decentralized Online Social Networks

Even if domain validation has been performed in a secure fashion and users interact with authentic web pages privacy issues exist.

During the last years the number of users of *Online Social Networks* (OSN) has sharply increased. These networks are incompatible with each other: if a user wants to get in contact with friends on different OSNs he has to register in each network separately. This circumstance may be a more important OSN selection criterion than e.g. the quality of service, potentially impeding diversity and innovation. As a consequence, this had lead to an accumulation of huge user bases on a few OSN providers.

Researchers have investigated various technical solutions for solving the privacy concerns that arose from centralized OSN.

In this chapter we survey state of the art decentralized OSN systems and compare their characteristics. Two such systems are Diaspora and OneSocialWeb which both use a federated architecture of independent servers. Another class of systems uses end-to-end client-side data encryption to hide the data content from the storage providers. This includes Persona, Vegas and SoNet. The third class of systems – PeerSoN, Safebook, LifeSocial and Cachet – utilize a distributed hash table (DHT) instead of federated servers. The fundamental characteristic which we compare is the privacy benefit of the decentralized approach. Apart from personal content this includes more subtle data and metadata, e.g. the time at which the user was active or the *social graph* the user is interacting with. A closely related question to that of

privacy benefit is the cost that it causes. While some users may be willing to accept performance penalties to a certain degree, a privacy preserving system has to compete with its centralized forerunners in terms of efficiency and functionality. In March 2013, Facebook registered 751 million active mobile users. These users will unlikely give up mobile access if they switch to a privacy enhanced OSN. Therefore, a succeeding OSN needs to run efficiently, including on mobile devices with limited hardware resources.

The contribution of this chapter are (1) a survey and classification of state of the art OSNs and (2) an identification of open research challenges regarding privacy and efficiency in OSNs.

This chapter is organized as follows. The next section surveys federated OSNs and compares their privacy properties. Section 4.2 evaluates encryption schemes, their application in OSNs and limitations regarding performance and confidentiality of metadata. Section 4.3 examines the implications of peer-to-peer OSNs originating from architectural design choices. Section 4.4 concludes this chapter with an identification of operational obstacles new OSNs face.

4.1 Server Federations

A server federation is a decentralized system of loosely connected servers, each being run by an independent data-holding entity (Figure 4.1). This should prevent accumulation of large amounts of personal data in one place.

One famous representative of this class of OSNs is Diaspora [37] which received a large crowdfunding and has been featured by the New York Times.[1] With about 380,000 users,[2] it is the largest decentralized OSN. Diaspora consists of independent servers (called *pods*) which communicate with each other. Users can either register on an existing pod or create their own. In the later case, the user stays in control of his data because the pod runs on a machine administrated by himself. Diaspora has a fine grained access control scheme to share content with specific contacts or contact groups. The server-to-server protocol specifies transfer of encrypted and signed messages providing security comparable to SSL/TLS. End-to-end encryption does not take place.

This is similar to OneSocialWeb, which is another approach for decentralized

[1] http://www.nytimes.com/2010/05/12/nyregion/12about.html
[2] https://diasp.eu/stats.html

online social networking.[3] It aims to enrich the Extensible Messaging and Presence Protocol (XMPP) with OSN features. XMPP itself is an instant messaging protocol with a federated architecture. OneSocialWeb leverages the existing XMPP messaging infrastructure with enhancements for creating profiles, expressing social relationships and sharing content.

Although Diaspora and OneSocialWeb allow users to set up their own servers, personal data will not always stay there. If friends from other servers request this information, it will be transfered to their servers and become accessible in plaintext to the service provider. Consider the current deployment status of Diaspora: the largest server hosts 70% of all users [12]. One service provider has thus access to the data of 70% of Diaspora users plus their friends' data stored on other servers. Therefore, a theoretical decentralized architecture itself is not enough to prevent aggregation of user data in one place. This shows that a user has to trust his service provider as well as the friends' service providers. Furthermore, one service provider can see every interaction of its users since every user is uniquely identifiable (i.e., *username*@*hostname*). This enables the service provider to determine the social graph of its hosted users.

This shows that merely distribution of user data on different servers combined with server-side enforced access control is still prone to attacks on users' privacy. The users' privacy depends on providers not disclosing their data without authorization. In the next section we discuss decentralized OSNs that use encryption to hide content from service providers.

4.2 Encrypted Data Storages

To cope with the issue of curious or malicious service providers, researchers suggested the use of encryption on the end hosts. With encryption and decryption taking place at the edges of the network, service providers become ordinary storage providers for opaque data blobs. Several challenges arise from using cryptographic user-to-user security in an OSN. First of all there is the well-known key exchange problem: How can users exchange cryptographic keys over an untrusted server without initial knowledge of authentic data?

This can be solved by an out-of-band method, e.g. face-to-face meeting. While this is generally cumbersome, methods like Bluetooth, NFC or QR-codes can make it more

[3] http://onesocialweb.org/

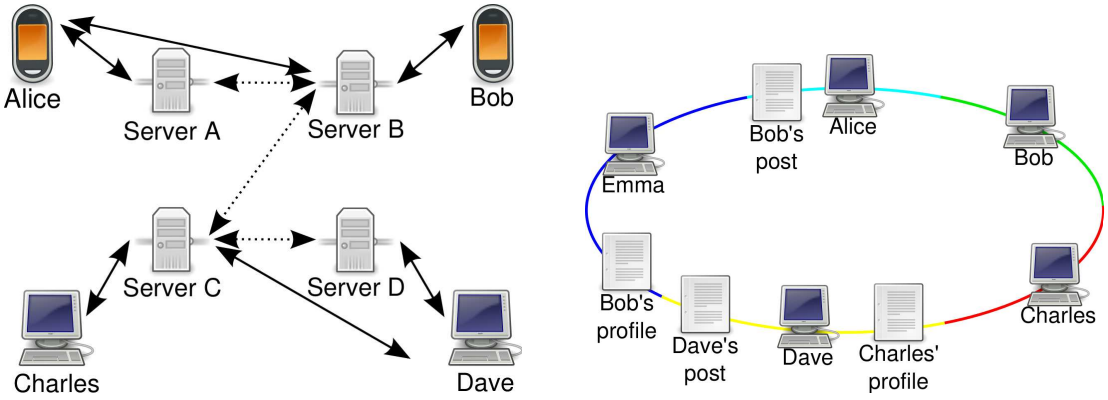


Figure 4.1: Examples of OSN architectures: A federated system with interconnected servers (left) and a DHT-based structured peer-to-peer system (right). Indicated by color, files are associated with their responsible peer.

comfortable, e.g. as used in Vegas [40]. Inviting new friends to a privacy enhanced OSN by mail is discussed in [39]. An in-band authentication approach is the Socialist Millionares Protocol which prevents a malicious OSN provider from interfering with exchanged messages by using a shared secret. This shared secret can be a common information in natural language, e.g. location where two acquaintances met first [108].

As it is not possible to crawl a friend's list for new contacts, Vegas specifies a coupling mechanism. A user can initiate friendship authentication between two of his trusted friends and to guarantee for their correct identity. During this coupling procedure new keys are exchanged so the initiating user cannot read future messages exchanged between the coupled friends.

4.2.1 Encryption Schemes

In an unencrypted OSN it is the responsibility of the server to enforce access control lists. With user-to-user encryption, the end host becomes responsible to enforce access control via cryptography.

Traditionally, a hybrid encryption scheme is used. Every user has a public/private key pair. Messages are encrypted with a symmetric key, which is in turn encrypted with the public key of each recipient. The idea is to reduce the runtime of the computationally expensive asymmetric encryption and to be able to reuse the symmetric key for future messages. In particular, the runtime of the symmetric encryption does not depend on the number of recipients. To revoke a user's access to a group key, a new symmetric key must be rolled out to the remaining group members. This requires

again to perform asymmetric cryptographic operations, but removing a user from a group is expected to happen less often than publishing new content.

Persona [5] uses a different encryption scheme called Attribute Based Encryption (ABE). With ABE, the symmetric content key is encrypted to be shared with pre-defined groups or combinations of groups ("co-worker $\wedge$ friend"). This allows to encrypt a new group key with one ABE operation for intersections of existing contact groups. This is more efficient than the traditional hybrid approach which would encrypt the group key n times for each of the n recipients. While saving space, it comes with a higher computational cost: ABE operations are about 100-1000 times slower than those of RSA.

A similar encryption scheme is Identity Based Broadcast Encryption (IBBE) [96]. IBBE is more flexible than ABE as it addresses individual recipients instead of pre-defined groups. It also supports removing recipients from groups with no further cost. This differs from traditional hybrid approaches and ABE which must re-run the generation and roll-out new keys.

4.2.2 Metadata

At a glance, the idea of encrypted content seems to obsolete the necessity for traditional access control lists. However, world readable storages of encrypted data may still leak metadata like timestamps, size of data objects, data structures or header information. Data lengths can be used to identify object types (texts, images, videos, likes) and header information leak group communication patterns and allow social graph inference [51]. A newly appearing data object implies OSN activity at a certain time, which can be of interest to employers for example. To prevent a third-party attacker from harvesting such metadata, server access control can be used in addition to end-to-end encryption.

Persona takes a different approach and does not offer public lists of data objects. Any user that can name a data object may retrieve it, but references to other data objects are encrypted together with the content. This hides metadata from public but still allows server providers to observe interactions in the social graph. Friends are granted write access by signing a token with their public key, which is a unique identifier. With the IBBE encryption scheme, each message contains identifiers of the recipients, including those who do not write back. This leads to the question whether the social graph can be hidden even from storage server providers.

4.2.3 Hiding the Social Graph

An approach to hide the social graph from server providers is aliasing in the SoNet OSN which is described in detail in the next chapter. Users communicate exclusively with their storage service which is also used as proxy for accesses to other users' storages. Upon friendship establishment between Alice@ServerA and Bob@ServerB both sides generate random identifiers, i.e., ServerB sees Alice as j@ServerA and ServerA sees Bob as i@ServerB. These identifiers are unique per friendship, so if Charles@ServerB becomes friends with Alice, ServerB sees Alice as k@ServerA ($i \neq j \neq k \neq i$). Each storage server knows all aliases of their users and can therefore forward messages to the correct recipient. A storage cannot resolve aliases of other storages to cleartext usernames. Due to the uniqueness of aliases, the storage does not see if two of its users are friends with the same user on another storage.

Aliasing attempts to disguise user identities in an aggregated user base on a storage server but fails with imbalanced user distributions. Consider e.g. a storage server with 70% of all users on which all relations among each other are known to the provider, or communicating with a storage which has only one user. Colluding storage providers can as well unveil relations between their users. An ideal setup would be a homogeneous distribution of users on independent servers but this cannot be enforced by technical measures.

The OSN Vegas takes a more rigorous approach to hide the social graph [40]. Viewing a friend's contact list is deliberately not a supported feature. Vegas uses public storage servers but with random file names and hidden directory structures. One user can run multiple storage servers and point each contact to a separate server. For each contact, different cryptographic keys are used to avoid identification by public keys. This hides the social graph from server providers and from the friends but leads to limitations which are discussed in the following section.

4.2.4 Performance and Limitations

Encryption and signing on end user devices necessarily imposes technical constraints. If storage providers are not able to read user data, they cannot easily assist the user at processing this data. This includes e.g. searching, creating image thumbnails, or computing summaries over large data sets. In a worst case scenario clients have to download all data, decrypt it, and process it locally. While there is research on homomorphic encryption allowing server-side operations on encrypted data it is currently

limited to rudimentary operations and not practical for real applications. Nevertheless, the performance of basic OSN functionality does not need to be considerably inferior to OSNs without end-to-end encryption, even on mobile devices. In SoNet, the time overhead for the cryptographic operations of a typical group message was 28 ms on an average Android device[108]. While this shows feasibility for symmetric ciphers and classic asymmetric ciphers like RSA, more computational complex schemes like IBBE are not suitable for mobile devices.

All of the discussed encrypted OSN approaches prevent leakage of metadata to third-parties. SoNet attempts to obfuscate the social graph from server providers by communicating with aliases. The aliasing has a low network and CPU overhead but hides the graph only partially. Vegas effectively hides the social graph from server providers but does not support OSN features like discussion groups or comments readable by other contacts. Avoiding those communication types which reveal social relations consequently yields 1:1 messaging. Sending one message to n recipients is possible but expensive because it requires $2n$ asymmetric cryptographic operations. Posting a status message to 100 friends would take more than 2 seconds on a mobile device.

4.3 Peer-to-Peer Approaches

Peer-to-peer based OSNs have been proposed that differ from federated approaches by using shared resources of connected clients for storing user data that is encrypted and can only be accessed by users that have the necessary keys, e.g. friends. The idea is to waive dedicated third party servers and to have a system which scales automatically with the number of participating users. The common element is a distributed hash table (DHT) which is used to find data in a peer-to-peer system with logarithmic routing complexity.

Distributed hash tables assign every participating peer and every data object that has to be stored an ID within the DHT key space. Every peer is responsible for the data stored in a part of the key space (e.g. closest IDs), as sketched in Figure 4.1. The allocated hard disk space can thus be much higher than the data a user has actually stored in the DHT himself, which may lead to a perception of unfairness. As peers sign on and off, the key space responsibilities of peers have to be adjusted, which requires a constant exchange between peers. This poses an overhead compared to server federations, especially when the DHT is used as payload storage and not

just as a data index.

This approach is used in LifeSocial [50] which stores all user content in a Pastry DHT. Data objects can contain actual payload data or references to other data objects. References represent a distributed linked list and allow efficient storing, e.g. a photo object being linked from different albums and from the user profile. However, this has implications for data retrieval. Each reference has to be resolved by a DHT query which will usually be routed to different peers. Since the assignment of data objects to peers is seemingly random, it does not take authorship or network proximity into account. Even if resolution of references is parallelized, the total time to fetch a subtree is still linear in its depth. While the authors evaluated performance for a small network of 30 nodes, practicability for a large network consisting of millions of peers remains unclear.

As this may be a major obstacle for practical deployment, Cachet [89] attempts to improve lookup times by caching data at friends. In addition to maintaining a DHT network, Cachet establishes direct connections to friends to push data objects to them. Friend connections are secured with SSL/TLS which is faster than decrypting many small objects individually. If a user is offline, common trusted friends can be used to retrieve content without costly DHT lookups. According to the Cachet authors, the time required to display an OSN newsfeed decreased from several hundred to around 10 seconds in their software prototype. While this is a significant improvement, it is still a perceptible delay compared to server-based systems.

In the related PeerSoN [19] approach content is always transfered directly between peers. The DHT serves as index to track the list of peers which hold a copy of each object. Each peer downloading data from a friend can enter its address into the DHT and share the cached data. This keeps the DHT thin, similar to the proven trackerless BitTorrent DHT, but what does this mean for privacy?

4.3.1 Privacy and Security Implications

In PeerSoN, the online status of all users and their IP addresses are world-readable from the DHT. The social graph can be easily derived from peers which cache data for their friends. In LifeSocial, social relationships are not reflected in the network topology because data-to-peer allocations are random. Nevertheless, each user is identified uniquely by his public key. Each data object in the DHT contains the cleartext IDs of the recipients, allowing recovery of the social graph as well.

Cachet makes use of object references to hide the recipients of data objects. The public keys are not saved with the data objects but at its parent objects in encrypted form. Given a well-known object as entry point, only authorized friends see the recipients of the referenced objects. This yields a privacy level comparable to federated, world-readable servers.

The main privacy difference between a federated and a DHT-based online social network is the distribution of responsibility and power. In a server federation a user chooses a provider and trusts him to not delete his encrypted data or attempt deanonymization by analyzing network traffic and metadata. This provider remains fixed until the user decides to move to another provider. In a DHT the responsibility and power is dispersed to a varying set of peers, assuming that the majority is trustworthy or at least not interested in decloaking the users identity. Due to the openness of the system, a DHT provides weak spots to increase the power that one peer should have. Examples include deleting or modifying data, denying access to data (DoS), choosing a peer ID to control a specific part of the DHT key space or placing oneself in DHT routing tables to control large parts of network traffic [126].

4.3.2 Friend-to-Friend Networks

The use of existing real-life trust has been suggested in Safebook [29] as foundation for an online social network. Each user connects directly to trusted friends to store data and forward messages. This constitutes a friend-to-friend network with concentric circles around each user called *matryoshkas*. That way it is possible to route from the outermost to the innermost circle by only traversing strong trusted links as seen in Figure 4.2. The outermost peer signs into a Kademlia DHT as entry point for a user's matryoshka network. These paths are created randomly during bootstrapping. Using this scheme, it is possible to communicate with a peer without knowing his IP address or disclosing his identity. The peers on the innermost circle, which are connected directly to the user, are used to mirror his profile data. These mirrors also store comments from other users and present them to the user as soon as he is online.

The matryoshka network reflects the social graph but as social links are trusted connections, outsiders cannot disclose the graph. Like in other peer-to-peer-based systems, it is again the DHT that offers weak spots: adversaries can sign into the DHT as entry point for any existing user or attempt to control the DHT routing or key space. The principle to replicate data only to friends and not into the DHT puts

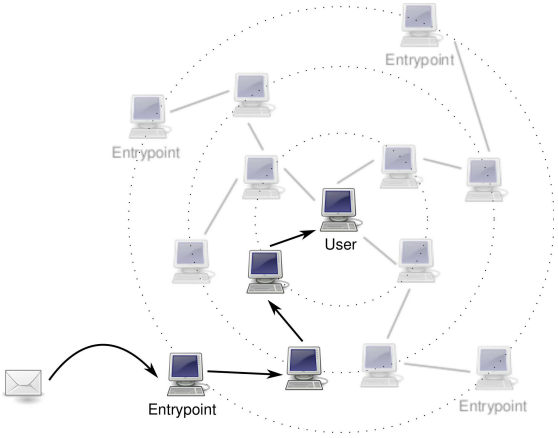


Figure 4.2: Message routing through matryoshkas circles. Lines between circles represent a friend relation.

constraints on the usability. Most friends can be expected to share a similar circadian rhythm and turn off their devices when they go to bed. Friends staying in other parts of the world will need to share online time to send messages and vacation pictures. While this can be alleviated by keeping some devices online around the clock, it questions the suitability of a pure peer-to-peer network when it is in fact a network of servers that keeps the system running.

4.3.3 Mobile Devices in P2P Systems

The effort to maintain a DHT or another type of peer-to-peer network causes a continuous basic system load. While this may be negligible for desktop devices, it renders peer-to-peer technology unsuitable for mobile devices whose battery will drain and data volume might be exceeded. This is an additional load besides the previously discussed cryptography on mobile devices. Mobile devices do not necessarily need to become part of the peer-to-peer network to access the DHT: desktop peers can act as gateways. This saves resources on mobile devices and increases the network quality since only the more reliable desktop peers maintain the DHT. On the other hand, it may lead to an unbalanced distribution of resource consumers versus contributors. As the number of mobile devices increases, the scalability advantage of peer-to-peer systems diminishes.

4.4 Conclusion

There are different understandings about the objectives of privacy preservation in online social networks. An overview of the features of the discussed systems is given in Table 4.1. Most authors agree on the need to encrypt content before passing it into the storage service. OSN system designs are divergent about hiding metadata like online status, IP address, recipients, message size and length. We conclude that peer-to-peer-based systems, while evading the need for dedicated storage servers, are especially prone to leak metadata. Compared with encrypted data storages this can be seen as a cost/privacy tradeoff. Together with functional constraints with regard to the growing mobile user population, federated approaches are a better fit for future privacy enhanced OSNs.

Server federations without encryption are the only class of OSNs which allows some kind of server side data utility. While these are a privacy threat this approach could provide a advertisement based business model for free OSN hosting. Privacy is only provided by choosing a trustworthy OSN provider.

Besides the challenges every new OSN has to face (e.g. convincing users of established OSNs to switch, although there are none/few of their friends), a privacy enhanced OSN has additional obstacles: Higher loading times (Cachet), increased resource requirements due to advanced encryption schemes, key exchange related actions (SoNet, Vegas), unsuitability for mobile devices and a reduced feature set (Vegas) are examples of downsides observed in this chapter.

Furthermore, systems without end-to-end encryption have to be discarded due to missing confidentiality. This confidentiality can be achieved by using hybrid encryption schemes which have been shown to be feasible for mobile devices. Leakage of metadata to third parties can be prevented by read access control or disallowing directory listing. However, a partly unsolved problem persists in obfuscating the social graph in front of the storage provider. Vegas proposes a “one storage per friend” solution which solves this problem, but it results in poor multicast performance. On the other hand, SoNet is efficient but only hides the graph partially and under certain preconditions in terms of user distribution. A solution realizing both, generally hiding the social graph while being efficient, is yet to be found.

	Diaspora	OneSocialWeb	Persona	Vegas	SoNet	PeerSoN	Safebook	LifeSocial	Cachet
End-to-end encryption			+	+	+	+	+	+	+
DHT-based						+	+	+	+
Federated servers	+	+	+	+	+				
Graph hidden				+	(+)		+		
Replication					+	+	+	+	+
Mobile support	+	+	+	+	+	+			
Hide activity from 3rd party	+	+		+					

Table 4.1: Overview of characteristics.

Chapter 5

SoNet – A Federated Online Social Network

After providing a general overview of decentralized OSNs we describe *SoNet*, a federated OSN approach, in detail in this chapter.

In SoNet, users may become their own data storage provider or choose a commercial one to rent storage capacity. Storage servers exchange data via a network protocol, enabling user interactions which span across different OSN providers.

A fundamental property of SoNet is end-to-end encryption of all user content. With such technical measures, the OSN provider serves as online data storage for encrypted private information without access to the private keys. Besides the actual OSN content like messages or images we also consider the social graph as private information. The social graph reveals with whom a user interacts which potentially discloses common interests. Even more, it allows the identification of users who did not sign up with their real name [131]. In SoNet, this is addressed by an aliasing scheme which obfuscates the social graph.

Based on the experience of Diaspora, we consider server availability as a key issue in federated OSNs: Bielenberg et al. discovered that 50% of the Diaspora servers have more than 50% downtime [12]. Although Diaspora is different by design than our approach, it also uses a federated server infrastructure. We are therefore using data replication between OSN providers to cope with server downtimes and to reduce query delays.

If a privacy-aware OSN is to succeed, it has to provide a similar level of comfort and usability as an existing centralized one. Even if employing cryptographic mechanisms on the end-user device, the system must be efficient enough to run on mobile

phones. For further reference, our requirements are privacy (**R1**), availability (**R2**) and efficiency (**R3**).

The contributions of this chapter are (1) design of a caching mechanism which does not jeopardize privacy and (2) a novel social graph obfuscation scheme.

This chapter is organized as follows: The next section introduces the basic architecture of our approach and describes how OSN features are implemented on top of this. In Section 5.2 we define a novel approach to obfuscate parts of the social graph. Section 5.3 discusses SoNet's privacy properties under different attacker models followed by an evaluation of its cryptographic performance in Section 5.4. We conclude this chapter in Section 5.5 by summarizing the main features of our approach.

This chapter is based upon a publication from 2013 and reflects the state of the art at this time (cf. [108]).

5.1 Architecture

The basic architecture of our system consists of independent servers which communicate with each other using a federation protocol. Users choose one of these servers and connect to this host exclusively. If two users are to exchange a message, it has to pass one or two servers depending on whether both users chose the same provider or not.

This approach is similar to existing federated services like email or the Extensible Messaging and Presence Protocol (XMPP). Users are identified likewise by a *useridentifier*@*serverdomain* scheme. Figure 5.1 illustrates an example of such a system.

All data objects exchanged between users are end-to-end secured by cryptographic operations. Servers therefore are not able to interpret them but merely forward them as opaque binary blobs. Hence we separate the system into two layers: a federation protocol used between servers to exchange data and the actual OSN protocol used between clients. Since clients do not connect to each other, the OSN protocol is tunneled through the federation protocol. With this separation in place, the storage could be used for other collaborative applications in parallel with the OSN.

5.1.1 Storage Servers

Servers offer clients methods for storing and retrieving data objects. HTTPS is used to provide authenticity, privacy and integrity between client and server.

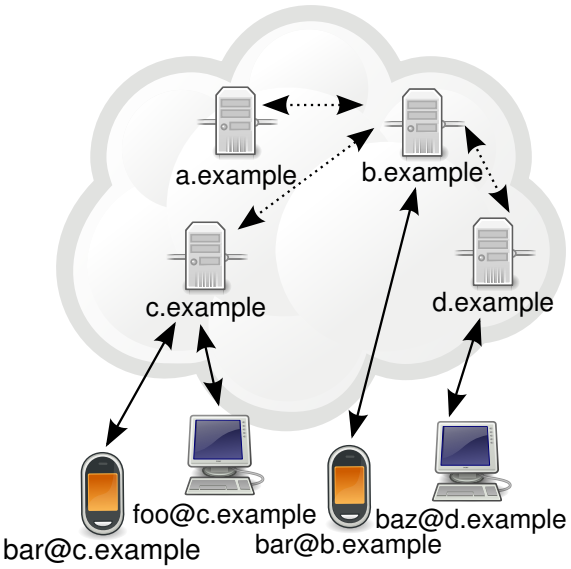


Figure 5.1: An example of a federated OSN.

The interface provides access to a rudimentary filesystem in which data objects are stored as files in directories. Each data object is accessible using a unique path. Such a path always starts with a full user identifier and a local path.

The user identifier contains the server's address an object will be stored on. Clients can access data objects by querying their server. If the host part of the user identifier is not the same as the provider's address, the server will fetch the data object from a remote host and forward it to the client.

From a server's point of view, data objects consist of metadata (author, path, groups) and the actual data payload.

Clients can establish a permanent connection and subscribe for updates on a given path. After this, the server will push notifications about any changes in this path.

By default, only the user himself (in this context also called *owner*) may write to his storage. He can grant access to other clients by adding their user identifier and a symmetric secret to a list on his server. After this, the owner can define a set of writable paths for groups of friends. It is also possible to allow write-access to a certain path for all users. The server will enforce this policy by checking author/location combinations of incoming data objects.

The servers of different providers use a federation protocol. Like the client protocol it uses HTTPS. In contrast to client-server communication, the requesting host uses an X.509 server certificate to authenticate in TLS. Thereby authenticity between

servers is ensured.

Servers exchange data objects written by their users. Servers are *authoritative* for objects whose path starts with one of their user identifier. If a user writes an object on a server it is not authoritative for, the server will *push* that object to its destination server.

If a server is authoritative for a user-written object, it pushes a notification about this new object to all servers hosting a user in the author's friend list. These remote servers decide whether they will pull that object or wait until one of its client actually requests it. If one of its clients has subscribed for the object's path, it will always pull the object. Otherwise heuristics are used and only small files below a certain threshold are pulled immediately. That way servers act as caches for objects of their user's friends.

5.1.2 OSN Features

Based on this federated architecture we have developed a distributed OSN. In the following section, we will list some of its features and how they are mapped to the underlying federation system.

5.1.2.1 Circles

We have adopted the idea of circles from Google Plus. Each circle is a group of friends which has access to different objects of the storage. If a data object is encrypted, a member of a circle can only decrypt it if that circle is mentioned in the object's *groups* attribute. This allows fine-grained access control.

5.1.2.2 Posts

Users can create posts to be shared in specified circles. Posts by $a@X$ are data objects stored in $/a@X/\_s/posts/$ using a generated post-id. Posts can be commented by other users. Given a post-id p, the comments are stored in $/a@X/\_s/posts/p/comments/$.

5.1.2.3 Chat

Besides sharing content in circles, there is also the possibility to exchange messages with only one friend. This is implemented in chats. Each chat message m is associated with a friend $b@Y$ and stored in $/a@X/\_s/chats/b@Y/m$.

5.1.3 OSN Security

We use a hybrid cryptosystem to enforce confidentiality. All content produced by users (posts, chats etc.) is client-to-client encrypted. For each entry, a random *entry key* is generated to encrypt it.

There is a *circle key* for each circle. This key is known to the circle's members and can be used to share content with them. For this purpose, the *entry key* is encrypted using the *circle key* and stored in the header of that entry. If content is shared in several circles, the *entry key* is encrypted once for each *circle keys*. There is a *key identifier* for each encrypted circle key to tell them apart.

Given such an identifier I, client $a@X$ can retrieve a circle key of author $b@Y$ by fetching it from /$b@Y$/_s/keys/circle_I. In this container, the circle key is encrypted for each of $b@Y$'s friends in that circle using their public key. These identifiers are consecutive integers starting from 0 and collide therefore for different users.

The expensive asymmetric decryption of circle keys can be relativized by caching decrypted circle keys. This is possible, because they change seldom. The circles whose keys have been used to encrypt this entry are stored in the data object's groups attribute. This allows clients to query only for data objects which they can decrypt.

Comments are handled differently: If Alice comments a post by Bob it is undesirable for a friend of Bob who is not a friend of Alice to be able to read it since depending on the context it might reveal information about Alice. Only those who are both a friend of Alice and were able to read the post in the first place should be able to read that comment. Therefore comments are encrypted using a hashed concatenation of both entry key and one of Alice's circle keys.

As soon as a friend is removed from a circle it is crucial to prevent him from reading any further content shared in this circle. Since he already possesses the associated circle key, we have to generate a new one with a new key identifier and distribute it by writing a new encrypted circle key for each remaining friend in that circle.

Besides circle-shared entries there is also content which is shared with a single friend. Chat is an example for such an access model. In that case, the entry is not encrypted using the circle key but a friend-specific symmetric key. This key can be accessed by the recipient of the message $b@Y$ at /$a@X$/_s/keys/$b@Y$/f2f. As circle keys, this friend-to-friend key is encrypted with $b@Y$'s public key.

So far key distribution has been discussed for reading clients. However, users are accustomed to using different devices for their OSN activities. To participate in this

OSN, a client has to be in the possession of all keys required for writing as well. Therefore clients store a copy of each key on the server. This copy is encrypted using a symmetrical user defined password. From a user's perspective, the application asks for this password at login as it is known from existing OSNs.

5.2 Social Graph Obfuscation

To hide all information that might hurt the user's privacy or unveil social interactions from other people we describe in this section how social graph obfuscation is achieved in our OSN architecture.

To obfuscate relationships and interactions between users within the OSN we hide bidirectional links by using *single direction aliases* (SDA) that are unique for each direction of user-to-user relationships instead of user identifiers. Aliases are random strings of sufficient length that are generated by servers to hide user identities from other servers and other users within the OSN and must be unique within a server.

The server stores a set of aliases for each user $a@X$. This set contains for each friend an entry with the alias $a@X$ is known by and the server of the friend {(*alias@own*, *server_of_friend*),...}. This way an authoritative server can resolve an incoming object addressed to an alias to the real username.

It is important to note that the alias sets do not contain any information about aliases of friends. The aliases of friends are only known by their own servers and by their friends.

Figure 5.2 shows an example of the local knowledge of two users $a@X$ and $b@Y$ and their respective servers X and Y. The example shows that only the two users know who the real usernames behind *both* aliases are. Even during communication between $a@X$ and $b@Y$, the servers X and Y are not able to determine any additional information about their users' friends, except that a user is communicating with *someone* at another server (noted as *?@server*).

Since the aliases are unique for every friend, friends are unable to see which friends a friend has and if there are common friends, unless this information is intentionally disclosed by a user to his friends.

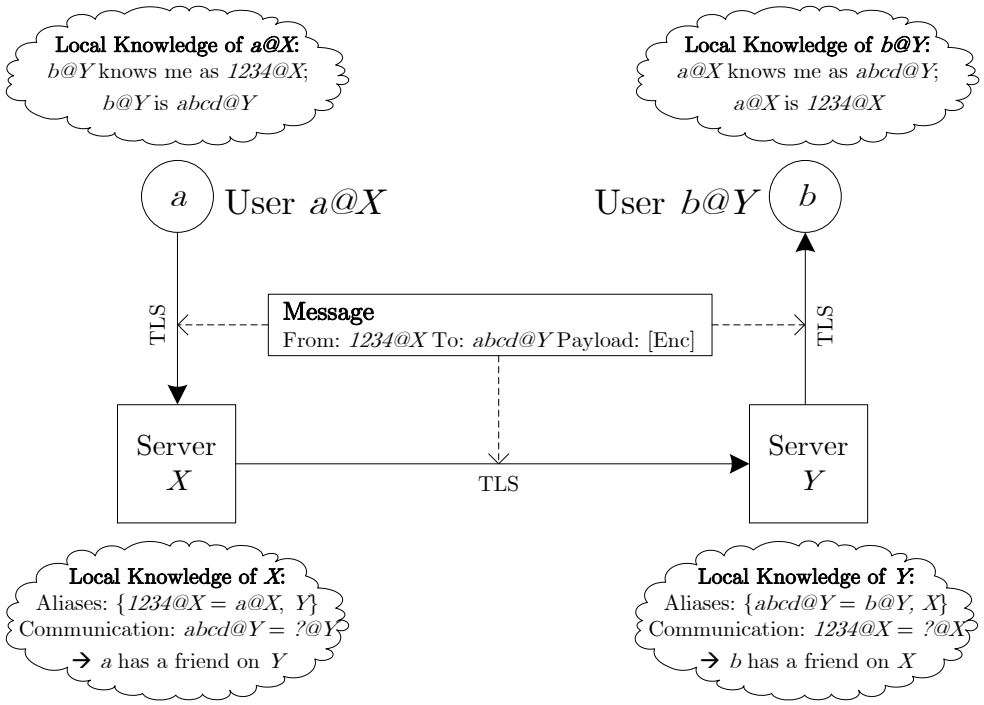


Figure 5.2: Disclosed information during communication.

5.2.1 Assigning Aliases and Becoming Friends

The main problem when creating new friendships between users is to not disclose personal information during the procedure of becoming friends until both users have proven their identities. Thus to provide security and authenticity, a handshake has to take place which involves exchanging each other's public key. It is essential that no man-in-the-middle attack can be performed, e.g. if $a@X$ wants to establish friendship with $b@Y$, both server X and server Y could intercept messages and provide forged public keys to both clients. This problem of initial key exchange is well known and several solutions have been proposed (e.g. Diffie-Hellman key exchange).

To achieve authenticity and security between two users trying to establish a new friendship we propose two friendship establishment procedures. First, an *out of band* authentication using already established trusted connections between users (e.g. email, Skype, ...) and second, a modified version of the socialist millionaires' protocol [1].

Out of band: As shown in figure 5.3, before creating aliases the users $a@X$ and $b@Y$ have to exchange their usernames and public keys, using a third party communication channel (e.g. email). Afterwards the user $a@X$ initializes the alias

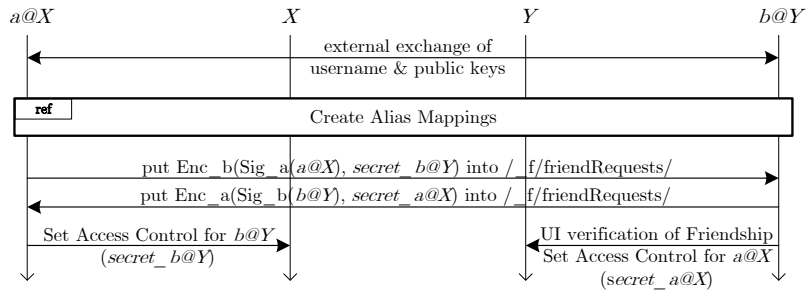


Figure 5.3: Out of band friendship creation.

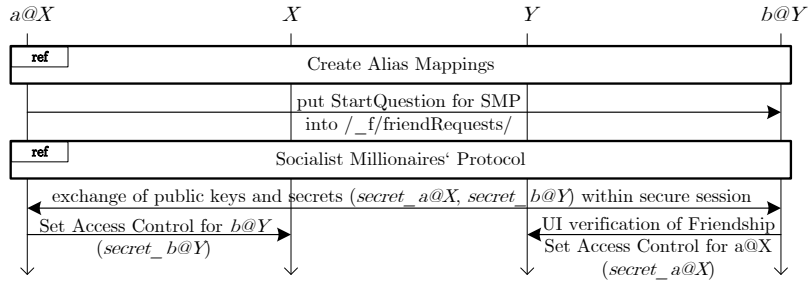


Figure 5.4: Modified Socialist Millionaires' Protocol friendship creation.

creation leading to $a@X$ knowing an alias to contact $b@Y$ and vice versa. Using these aliases $a@X$ puts a friend-request object containing its signed real username and a secret for calculating the message authentication code (MAC) encrypted into a specific directory $/b@Y/\_f/friendRequests/$ of user $b@Y$. The user $b@Y$ creates and stores the friend-request object the other way around. Now both users can verify the identity of the incoming friend-request and grant their new friend permission to write data objects.

Modified Socialist Millionaires' Protocol: For our second friendship establishment procedure we chose the socialist millionaires' protocol (SMP) for mutual authentication as shown in [1]. SMP is a zero knowledge protocol which allows two parties to compare a secret without revealing anything about that secret except whether their secrets are identical. We propose to use the SMP in such a way that the user $a@X$ who sends a friendship request to another user $b@Y$ only has to enter a question and a matching answer which is only known to him and his friend (see Fig. 5.4). If the protocol succeeds, both users end up with the correct public key of another. Using this authenticated public key $a@X$ and $b@Y$ can check the identity of each other and exchange the secrets for writing access verification.

Assigning aliases: When creating *single direction aliases* a user's identity needs only to be known to his own server and to his friends after the friendship was estab-

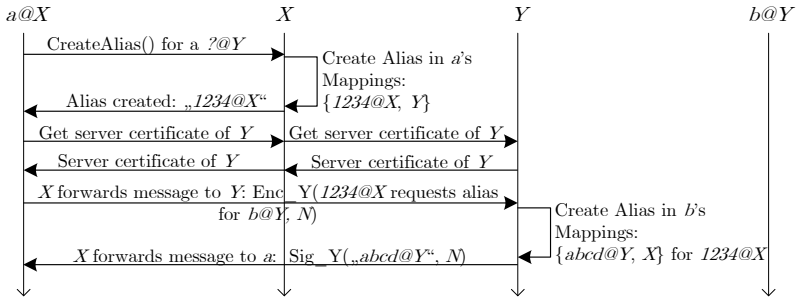


Figure 5.5: Creation of aliases.

lished. The identity or personal information do not need to be disclosed to other authorities, in particular not to the friends' servers. Figure 5.5 shows the alias creation in detail for a user $a@X$ requesting an alias for addressing a user $b@Y$. The client of $a@X$ requests a new unused alias from its own server X. Within this request $a@X$ states that this alias is determined for *someone* at server Y ($?@Y$) and thus server X does not know who will use this alias to contact $a@X$. However, server X does know that $a@X$ can be contacted under the specific alias from someone at server Y. First $a@X$ requests the server certificate of server Y to encrypt the communication between $a@X$ and Y. By relying on X.509 $a@X$ ensures that the certificate is valid and has not been tampered with. $a@X$ requests a new alias for user $b@Y$ on server Y. To hide its identity from server Y, $a@X$ uses its own alias as sender that is intended for communication with $b@Y$. Furthermore, it contains a cryptographic nonce N to detect replay attacks. Server Y now creates the alias for $b@Y$ and notes that the alias is used by $?@X$. Then server Y stores the requesting alias of $a@X$ to notify $b@Y$ how the user may contact $a@X$, even though neither the server Y nor $b@Y$ know which user is related to this alias. However, $b@Y$ will learn who the real user behind the alias is during the friendship creation before accepting the friendship request. In the end, server Y signs the created alias for $b@Y$ and N and sends them back to server X. After $a@X$ has validated the signature and checked N he knows an alias to contact $b@Y$, as well as the alias that $b@Y$ will use to contact him. The user $b@Y$ knows how to contact $a@X$ and which alias $a@X$ will use to contact him (after finishing one of the friendship creation protocols mentioned earlier). The server X only knows that $a@X$ can be contacted using a specific alias by $?@Y$ who uses another specific alias (and vice versa for server Y). Thus, the friendship relation between the real users cannot be recreated by one server alone.

5.2.2 Replication and Anonymous Retrieval

To increase availability of data objects we use caching on non-authoritative servers. This means that there can be several copies of a data object in the federation as a whole, since servers cache requested data objects that may be reused. If a user $a@X$ wants to read the object $/b@Y/\_s/file1$ and server Y is currently unreachable, server X first checks if it has a cached version. In this case the object will be retrievable without noticing the downtime of server Y. If this is not the case, $b@Y$'s friends could still have a cached version. However we do not want do disclose any information about friendship relation and thus cannot directly tell $a@X$ which friends of $b@Y$ may have cached versions of the requested object.

A key feature of our obfuscated OSN is, that replicated or cached user-data at friend-servers can be retrieved without disclosing any friend relation, even to friends. Our approach is a primary copy replication scheme realized as follows: Clients must be able to determine 1) on which servers they can access replicated data and 2) under which alias. Thus every user stores his alias set (that is maintained by its server) encrypted in his storage under the path /_s/aliasset. This file has to be updated by the client periodically. Every friend fetches this set and thus gets the information of aliases for $b@Y$ at other servers without knowing which friends of $b@Y$ these aliases represent.

Whenever the server Y of a friend $b@Y$ is not responding a user $a@X$ can contact a server listed in the alias-set and request cached objects for a specific alias. The server holding the replicated data neither knows which user is requesting the objects, nor whose replicates are requested. The request itself does not have to be restricted since only friends of the author will be able to decrypt the data.

5.2.3 Implications for Access Control

Hiding the identity of users from other servers has implications for access control. Servers can't use the public keys of remote users to verify data objects. Instead a symmetric key is used to generate a MAC. Servers have a list which maps aliases to such a key. Incoming objects can be verified by servers without requiring knowledge about their authors' usernames.

However the server is not able to verify that within the encrypted content the valid real username is provided. This can only be checked by the client by decrypting the content and checking if it is signed correctly. If posts happen to be spam or contain

an invalid signature the storage's owner is able to see the alias that has abused its writing permissions and revoke the permissions for this user.

5.2.4 Implications for Encryption

If an owner decrypts a comment posted at his storage, he will know the username belonging to the author's alias. Given this username and group information, he can choose the correct key.

If a client reads a comment on a remote storage, it has only the alias from the storage's owner point of view. It therefore cannot choose the correct key because the related mapping is not known to him. However, the group and author's host name can be used to reduce the set of possible authors. From this reduced set, tentative decryption is performed with each key. Once a correct key has been found, clients can store the deduced alias to username mapping to speed up future decryption. We chose this method of decryption to protect obfuscation. If key identifiers were unique, an attacker would be able to crawl hosts for encrypted circle keys. Once a key with an identifier I has been found in $a@X$'s storage, an attacker could deduce that all posts encrypted with I have been written by $a@X$. This is prevented by choosing colliding key identifiers.

5.3 Security Assessment

In this section we will discuss the provided security of our approach. We considered active and passive attackers in different scenarios and evaluated both obfuscation and confidentiality.

5.3.1 Local Area Network

In this scenario an attacker Mallory resides in the same local area network (LAN) as $a@X$. Using eavesdropping Mallory cannot break confidentiality since $a@X$ establishes an encrypted connection to her server. Even if Mallory modifies, forges or drops packets confidentiality still holds due to the integrity provided by TLS. An attack during connection establishment will also fail due to server authentication.

Since Mallory cannot break confidentiality, breaking obfuscation using transmitted payload data is not possible. However, Mallory can observe communication patterns, which might reveal parts of the social graph. Considering only sender and receiver on

a network layer level, there is no information leak since *a@X* communicates with her server exclusively. Her server might forward data to her friends, but since the attack takes place in the LAN Mallory also cannot observe how data from *a@X* is forwarded by her server.

If *a@X* and *b@Y*, one of her friends, are in the same LAN, Mallory could deduce that they are friends. Although both only exchange encrypted messages with their servers, the timing can be considered. If *a@X* sends a chat message to *b@Y*, *X* will forward this data object to *Y* and *Y* will push it to *b@Y*. Although Mallory cannot get hold of the content, correlation between these roughly equally sized messages allows Mallory to assume a friendship relation between *a@X* and *b@Y* with a certain probability. Repetition of such patterns, e.g. in a chat session, can increase that probability. In this attack, Mallory can only observe network addresses of *a@X* or *b@X*. She still does not know neither any alias of them nor their usernames.

Such attacks can be circumvented by sending bogus traffic and delaying message forwarding. However, such solutions decrease the efficiency. Especially with a mobile device, sending bogus traffic can require too much resources. Therefore, balancing the probability of such an attack with the costs of its mitigation, we accept this as a possible weakness.

5.3.2 Server

We will now assume that the attacker is a malicious server. As before, confidentiality remains unbroken. Only clients are in possession of the required keys. Social graph obfuscation can be reversed in a limited amount. Since servers have to keep a mapping of aliases to usernames, the malicious server can reconstruct all local friendship relations. However, this attack is limited to local friends only. Servers could also cooperate to break obfuscation between them. As before in the local case, this does not affect obfuscation of users on other servers, even if they are friends with a user on a malicious server.

Like Mallory in the previous scenario, servers can perform correlation attacks. If *a@X* and *b@X* are users of a malicious server, *X* could determine if they have a common friend *c@Y*. Although the alias for *c@Y* is different for both *a@X* and *b@X*, *X* can observe that *a@X* and *b@X* request identical objects from *?@Y*.

Besides passive attacks, the server could also perform data manipulations. Modifying existing data objects will cause clients to notice this since data objects are

stored in signed containers. The keys for these containers are only known to clients and therefore servers cannot forge signatures. The same is true for creating new containers.

The only possible data manipulation attack which is not detectable by clients are data object deletes. Those could be mitigated by clients generating cryptographic proofs of nonexistence as in [74]. However, this puts additional load on clients (every data object creation/deletion has to update these proofs) and only provides little benefit: Deletions could only be detected by clients but not prevented.

Clients store their keys in an encrypted container on servers. A server could perform a brute-force or dictionary attack on this to acquire all keys. This is mitigated by clients requiring users to choose strong passwords.

5.3.3 Fake Profile

A common problem in social networks are fake profiles, i.e. impersonation attacks. To be a threat to both confidentiality or obfuscation, the user had to be added as a friend in the first place. In this process, the user's identity will be verified (see section 5.2.1). For a faked profile, this verification will fail and therefore render impersonation attacks futile.

5.3.4 Malicious Friend

If a user $a@X$ has a malicious friend $m@X$ because a former accepted friend becomes evil, there are other attacks to consider.

Since $a@X$ never publishes her friend list to anybody even her friends cannot remove obfuscation using it. However, if $a@X$, $b@Y$ and $m@X$ are friends to each other, there is a high probability that messages are exchanged. If $b@Y$ comments one of $a@X$'s posts using a circle $m@X$ is part of, $m@X$ will know that they are friends. This could be mitigated by either making comments only visible for the original author of a post or by making posts anonymous. We believe however that this would decrease the benefits of an OSN since its functions would reduce to some kind of private messaging. Furthermore, this attack only succeeds if $m@X$ is a friend of $b@Y$. Otherwise $m@X$ would not be able to decrypt $b@Y$'s comment to $a@X$'s post.

Confidentiality, in its meaning of preventing unauthorized entities from acquiring secrets, is still given. This is because both $a@X$ and $b@Y$ gave $m@X$ permission to access their data and therefore $m@X$ is no longer an unauthorized entity. Other users

of the OSN do not suffer from consequences of $a@X$'s and $b@Y$'s decision to become friends with $m@X$.

If $m@X$ and $b@Y$ are not friends, $m@X$ will not be able to deobfuscate $b@Y$'s identity, even if $b@Y$ comments on $a@X$'s posts. Since key identifiers collide for different users, $m@X$ cannot correlate these with users.

5.4 Cryptographic Performance

To verify the feasibility on mobile devices, we measured the performance of cryptographic operations on an average consumer mobile phone in 2013 (HTC Desire S, Android 4.1.2). We chose RSA-2048, SHA-1 and RC4 as cryptographic primitives. We assumed an average user having 3 circles and 100 friends and calculated the mean value out of 100 test runs.

Before a user can participate in the OSN, he has to generate an asymmetric key pair. This one-time step required 3155.19 ms (±2088 ms). Creating a posts consists of 3 key encryptions, encrypting the post, generating an RSA signature and calculating a MAC. This took 28.47 ms (±1 ms) for a post of 150 bytes length.

Reading a post consists of choosing the correct key, RC4 decryption and RSA signature verification. In worst case, all 100 friends are in the same circle on the same host and the client has no known alias-to-user mappings. In this case, 100 keys have to be tested for decryption which took 10.44 ms (±0 ms) in total.

If a user is removed from a circle, we have to generate a new circle key for all remaining users in this circle. Again we consider the worst case with 99 RSA encryption operations and generation of one RSA signature. This finished after 56.12 ms (±1 ms).

5.5 Conclusion

We proposed a federated OSN in which all content is end-to-end encrypted between user devices. Servers act as storages for opaque data objects without having access to the keys. User-generated content cannot be read or forged by the server providers at all, meeting our privacy requirement **R1**. To hide the social graph from the servers we use an obfuscation technique which maps a username to a different alias for each directed edge in the graph. This way server providers can only identify friend relationships on their own servers, or need to collude to disclose the social graph beyond server

boundaries. The obfuscation approach is compatible with replication, improving the availability of data (**R2**) in case of temporary server downtimes. The application client always communicates directly with its own storage server, saving the overhead of many short-lived connections to third-parties. Despite using cryptographic operations extensively, performance measurements suggest that the application runs efficiently on mobile devices, complying with our efficiency requirement **R3**.

Chapter 6

Identifying TV Channels & On-Demand Videos

So far, we have discussed privacy threats by attackers on a network level and examined threats on the host itself in online social networks. However, even when the connection to the web server is authentic and content is end-to-end encrypted, privacy threats persist. Since users are not exclusively visiting their OSN provider's website, a broader analysis is required.

While surfing the web, users connect to a potentially high number of hosts controlled by various entities. Users might value the information provided by these websites even if they do not entrust them with personal information. For example, a user searching for product reviews might look at opinions from so far unknown web pages but would hesitate to share sensitive information with these web pages.

With the web as a platform, websites have become complex applications posing a potential privacy threat by disseminating data without the user's involvement. After a website has been downloaded, its scripts will be executed with an exposed Web API which can access local data sources and perform additional HTTP requests. Web browsers therefore employ a multitude of mitigation strategies to limit the capabilities of code downloaded from potentially untrustworthy servers. Most notably these are strict separation from the host system (realized by an interpreter, often complemented by a sandbox) and separation of data from different websites (same-origin policy). While both mechanisms prevent a broad range of attacks they do not limit what kind of data is sent to which server and how local data is being processed. Therefore, any information exposed to a website will potentially be disseminated. Websites can read ambient light sensors, gyroscopes and accelerometers without confirmation or even

awareness of the user. As these are low-level sensors their impact on privacy in terms of high-level statements is not intuitive and requires an analysis per use case.

In this chapter, we show that it is possible to detect which television (TV) show or movie a person is watching by utilizing the ambient light sensor of smartphones, tablets and smartwatches. We demonstrate that this is possible from apps as well as from web pages currently visited by the user. Our approach works by analyzing the ambient light that can stem from reflections of the wall if a mobile device is not pointed at the TV screen. The light level is collected with the ambient light sensor.

If the user is aware of this data collection and consents to it, this information can be used in various ways. For example, apps on the mobile device could offer users additional information about the current TV show since the app knows what the user is currently looking at. In addition, it can help with situational awareness to understand whether the user is watching an action movie, a TV cooking show or just a commercial. In the case of a movie, the user probably does not want to be disturbed, especially in the final minutes of the show. Hence, a pervasive computing system could mute the phone and refrain from playing sounds for every incoming chat message.

Determining the currently running TV show without user consent violates the user's privacy. We analyze which frequency and sensor resolution are sufficient for our approach. A privacy-aware mobile device could use this information to limit the entropy of the ambient light sensor: If the user gave permission, better data could be provided.

The contributions of this chapter are (1) a zero-permission approach for determining video's being played in the user's vicinity and (2) a novel edge-based algorithm to efficiently recognize on-demand videos.

The chapter is organized as follows: Section 6.1 provides physical background to light and its perception. Section 6.2 defines the system model in which the user is operating. Our multi-staged approach is described in Section 6.3. Section 6.4 provides details on our prototype implementation. We evaluate our approach extensively in Section 6.5. In Section 6.6 we introduce a confidence metric to increase the overall accuracy of our approach. Section 6.7 discusses real world challenges associated with our approach followed by a discussion of privacy implications. Related work is discussed in Section 6.9. Section 6.10 concludes this chapter by providing recommendations to mitigate the privacy issues found.

6.1 Background

The human eye can only detect a certain range of the electromagnetic radiation, which we interpret as color with different intensities. Photometric units consider this imbalance of sensibilities and valuate the visible color spectrum through a weighted function to accommodate the human eye. Ambient light sensors use the photometric unit Lux, which is implicitly based on the standard 1931 CIE photopic luminosity weighted function $V(\Lambda)$ [26]. The luminosity function is analogous to the Y tristimulus value from the CIE XYZ color space which is the standard reference for other color spaces such as sRGB.

Lux is the SI-unit for characterizing ambient light as perceived by humans and is used by light sensors in mobile devices, which our method is based on. Light sensors approximate Lux values by applying an empirical function to physical measurements of photodiodes. The accuracy is limited as a trade-off with power consumption and manufacturing costs, and the error is non-linear [104]. Hence, a robust method must cope with inaccuracies between different sensor implementations.

6.2 System Model

We assume a user with a mobile device watches a video on a dedicated screen, e.g., television or computer screen. The video is either an on-demand video stream or a TV broadcast respectively TV live stream. The user is indoors in dim light and the ambient light sensor in the device records diffuse light emitted by the screen. The sensor is not obstructed, for example the mobile device is held by the user or lying on a table face up.

Figure 6.1 shows our setup with different orientation scenarios that we consider in this chapter: *facing user* (S_1), *resting on table* (S_2) and *facing screen* (S_3), $\alpha = 45°$. Distance to screen (d) and distance to the white back wall (d') are also variables in our setup. d' is relevant for scenario S_1 because there is no direct sight to the screen and the light sensor relies on diffuse light reflected from the wall. We do not consider ceiling height as a variable of our system but assume standard residential values.

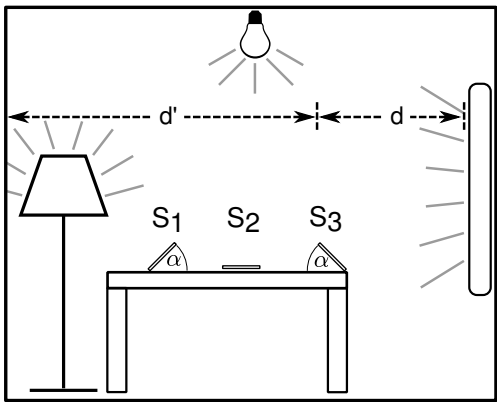


Figure 6.1: Experimental setup. Distance and orientation to screen vary.

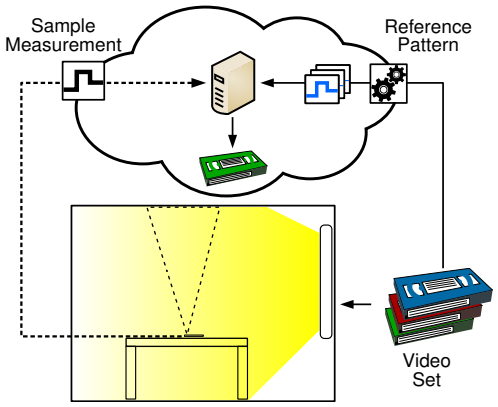


Figure 6.2: Data flow in overall system.

6.3 Approach

Our system is shown in Figure 6.2. The objective is to determine the video being played in the user's proximity from a set of known videos, e.g., currently running television shows or on-demand video content. The ambient light sensor readings from the user's mobile device are transmitted to an external server. The server has reference patterns ready, which have either been calculated from the RGB color information of the known video frames or measured in a reference setting. Reference patterns stem either from on-demand videos and have been precomputed once or represent a tv channel. In the latter case the server has to update these references continuously.

The server can identify the video played by comparing the Lux values retrieved from the mobile device with its reference patterns. Though these patterns do not necessarily match the mobile device's measured values exactly, we introduce a measure that works reliably despite varying light environments, different color reproduction of screens and different sensor calibrations. The reference pattern which is most similar to the sample measurement is considered to represent the video currently played back by the user.

6.3.1 Observing Similarities

The brightness values collected from a sensor measurement or calculated from video frames are discrete functions of time. Figure 6.3 shows three ambient light sensor measurements. The measurements cover disparate ranges of illuminance, which is the

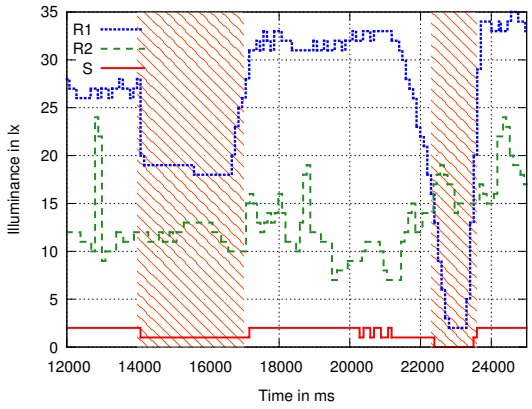


Figure 6.3: Brightness comparison of two reference patterns (R1, R2) and one sample measurement (S).

result of different light environments. R1 and R2 have been collected under ideal conditions with the phone facing the screen ($d = 1m$, S_3), while S has been collected in a more realistic setup with the phone resting on the table ($d = 2m$, S_2).

To illustrate the similarity of measurements, major changes of illuminance of S are indicated in the hatched area in Figure 6.3. As we can see, S and R1 behave alike; whenever S shows a variance, R1 does so too – although on a higher magnitude. This is not the case for S and R2. Instead there are some hints indicating different videos caused these measurements: neither causes any of R2's significant peaks a rise of S nor has S's drop at 22 300 ms any influence on R2. Based on this, one can conclude—correctly—that S and R1 represent the same video.

6.3.2 Correlation Analysis

Given the above observations, we will now formalize illuminance correlations. We define a measurement $A \subset \mathbb{N} \times \mathbb{N}$ as a set of tuples containing timestamps and Lux values. For two measurements A, B we define t_i as the duplicate-free ordered sequence of points in time where a value exists for either A or B, i.e.

$$t_i := \begin{cases} \inf\{x_1 | (x_1, x_2) \in (A \cup B)\} & \text{if } i = 1 \\ \inf\{x_1 | (x_1, x_2) \in (A \cup B) \wedge x_1 > t_{i-1}\} & \text{if } i > 1 \end{cases}$$

The samples do not have to share data points at the same points in time and as the example in Figure 6.3 indicates, this is rather infrequent in reality. However, to compare both measurements we have to interpolate values between data points. We

refrained from using linear interpolation because this led to poor identification rates in preliminary evaluations. The reason for this is the ambient light sensor resolution. Consider the interval between 14 000 ms and 17 100 ms in Figure 6.3: S would rise with linear interpolation, while the illuminance remains relatively constant for R1. Correlating a rising with a constant interval would lead to a weaker correlation coefficient than comparing two equally formed functions. Instead we interpolate values between data points by using a step function, i.e. we assume that between data points values remain constant. Formally, we define $f_X(u)$ as the illuminance of measurement X at time u as

$$f_X(u) := y_2 | (y_1, y_2) \in X \wedge \Big(y_1 = \inf\{x_1 | (x_1, x_2) \in X \wedge x_1 \geq u\} \Big).$$

Using these definitions, we apply Pearson's weighted correlation coefficient to determine the similarity of measurements. The fundamental idea is to correlate $f_A(t_i)$ with $f_B(t_i)$ for every element in t_i weighted with the time in between. The weighted average $\mathrm{m}(X,t)$ of a measurement X and the weighted covariance $\mathrm{cov}(A,B,t)$ are defined by

$$\begin{aligned} \mathrm{m}(X,t) &:= \frac{1}{\sum_i (t_{i+1} - t_i)} \sum_i (t_{i+1} - t_i) f_X(t_i) \\ \mathrm{cov}(A,B,t) &:= \frac{1}{\sum_i (t_{i+1} - t_i)} \sum_i \Big((t_{i+1} - t_i)(f_A(t_i) - \mathrm{m}(A,t))(f_B(t_i) - \mathrm{m}(B,t)) \Big). \end{aligned}$$

Finally, the weighted correlation coefficient $\mathrm{corr}(A,B,t)$ is defined by

$$\mathrm{corr}(A,B,t) := \frac{\mathrm{cov}(A,B,t)}{\sqrt{\mathrm{cov}(A,A,t) \cdot \mathrm{cov}(B,B,t)}}.$$

6.3.3 Determining Sample Offsets

Pearson's weighted correlation requires a *sample offset* O_S for the sample measurement within the reference pattern (see Figure 6.4). This sample offset states at which temporal point in the reference pattern the sample measurement is presumed to be located.

We present different methods to determine this offset for television channels and video on demand. The server has to chose the appropriate method according to the type of reference pattern it is comparing the sample measurement against.

6.3.3.1 Live Correlations

To determine O_S for television channels we propose a *live correlation* approach. The mobile device transmits ambient light sensor readings to a server which correlates these with its reference pattern set of TV channels. A potential source of interference during correlation at the server is a time offset of the mobile sensor measurement induced by latency between client and server, content propagation delay or imprecise clocks. We assume that this time offset lies within an application-dependent range R_{BF} of several seconds. In this case the potential sample offsets O_S can be determined by performing a brute force search, since the range of potential offsets is small. This is realized by correlating the sample measurement and the reference pattern in an interval of S_{step} milliseconds steps within the range R_{BF}. The correct sample offset can then be determined by choosing the O_S with the maximum correlation coefficient.

6.3.3.2 Deferred Correlations

If the user watches a video on demand stream, the mobile device can record ambient light sensor readings and send them to a server for a *deferred correlation*. In this case a brute force search is not feasible since a correlation would be necessary for every possible sample offset O_S within all reference patterns at the server. We therefore perform an optimized search to retrieve a list of potential sample offsets O_S within the reference patterns to reduce the effort necessary for correlation. This is realized by analyzing *features* of the sample measurement, as well as features of the reference patterns.

We consider a feature $F_n = (I \in \{H, L\}, \Delta t)$ to be a steep change in the illuminance of the measurement (H for a rising and L for a falling flank) in combination with the time difference Δt to the previous feature F_{n-1}. The idea for the approach is to extract distinctive features from the sample measurement S as well as from all reference patterns R_n. Note that the server can extract the features once for all reference patterns and does not need to recompute them for every correlation. We use illuminance flanks as features since they are more robust to work with than e.g. illuminance peaks. We dismissed using maxima and minima as these are not represented well by sensor readings since discretization obscures the exact point in time when the extreme value occurs. We do not consider the height of the illuminance changes since it highly depends on the ambient light sensor and light environment.

A key element of the sample offset determination is an over-approximation of

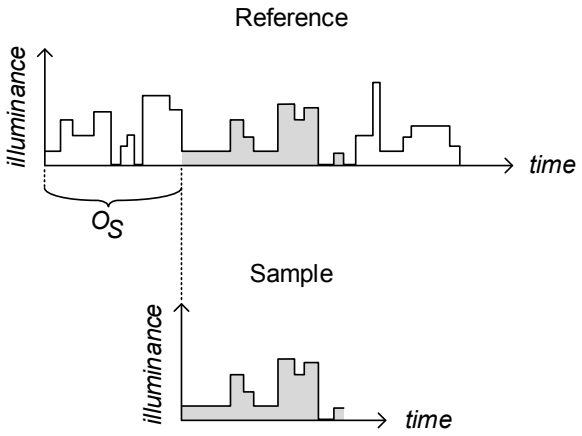


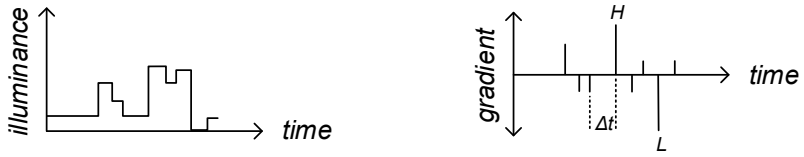
Figure 6.4: The sample offset O_S of a sample measurement within the reference pattern.

potential O_S values. This is necessary, since an under-approximation could yield a result set that does not contain the correct sample offset. Thus, the correlation would not be able to recognize the video, even though a brute force offset search would have yielded a correct correlation.

Before extracting the features we perform an edge detection for all measurements and reference patterns using a discrete derivative to approximate a derivation graph M' for the measurement graph M (see Figure 6.5). For gradient approximation at a certain data point x of M, we consider all points within a *derivation window* range D_W around x. Features are extremes in the derivation graph M' in combination with the time between the extremes and can thus be directly extracted. As result we retrieve a list of extracted features F_1 to F_n for the sample measurement S as well as for the reference patterns R_n.

Once the features of S and R_1 to R_n are extracted, we determine the potential offsets O_S of the sample within each reference. We expect the sample measurement to contain more features than the reference pattern (see Figure 6.6). This is because steep changes in illuminance in the reference pattern are always reflected in sample measurements — assuming the sensor is of sufficient quality — but the reverse is not necessarily true: Due to discretization a steep illuminance change in a sample measurement may also be caused by an inaccurate sensor.

For every sample feature F_n we search for features in the reference pattern that



(a) Measurement M. (b) Derivative M'.

Figure 6.5: Distinctive features in the measurement M and the corresponding approximated derivation graph M'.

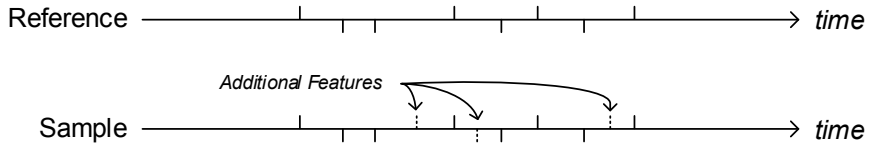


Figure 6.6: More features in the sample measurement.

are similar in terms of Δt, and whether it was a rising or a falling flank. Thus, we have to assume that for at least one feature F_n in the sample measurement there is at least one feature with about the same Δt and I (i.e. H or L) in the corresponding correct reference pattern. The reference features at the server are prepared for range queries enabling the algorithm to perform the lookup of features similar to the sample features in $\mathcal{O}(log\ m)$, where m is the number of features in the reference patterns. This has to be performed for each of the features found in the sample measurement.

Features that meet this condition are here referred to as *anchors*. For every anchor found we take the neighboring sample features left and right of the anchor and check if they align with a feature in the reference pattern. We consider two features as aligned, if their distances to the anchor are similar, i.e. the difference between the features is less than a *fuzziness threshold* T_F. Since the sample measurement might contain more features due to jitter it might happen that the feature of the sample measurement does not align with a feature in the reference pattern. Thus, the algorithm checks also the next T_L features, even if a sample feature could not be aligned. T_L is the *look-ahead threshold*, i.e. if $T_L + 1$ features did not align, then aligning is aborted. The search can yield several anchors, since several features of the sample might fit to features of a reference pattern. To differentiate how well the anchor fits to the

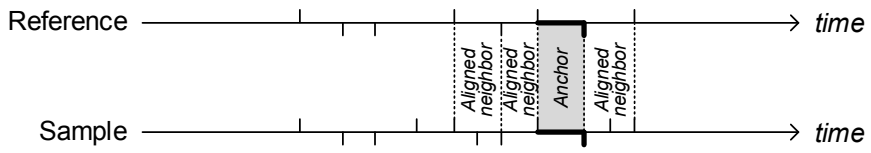


Figure 6.7: Anchors and alignment checks.

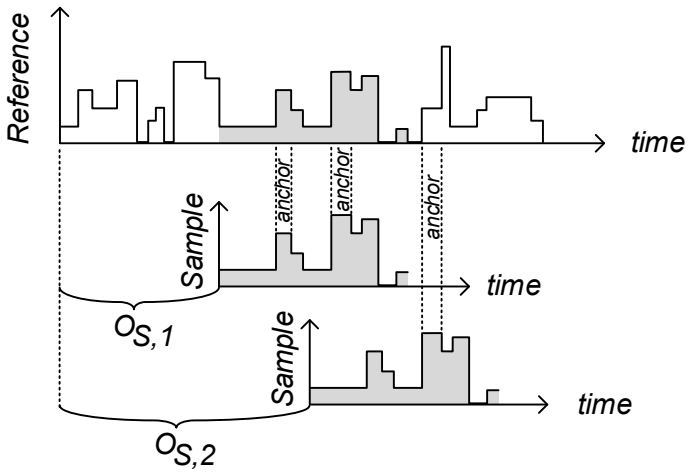


Figure 6.8: Deducing potential offsets from anchors.

reference pattern, a ranking of anchors is performed. The more sample features are aligned with the reference pattern and the smaller the difference between the aligned features, the better the anchor is ranked. The alignment of neighboring anchors is shown in Figure 6.7.

To determine the potential sample offsets O_S of the overall sample in a reference pattern (see Figure 6.4), the algorithm analyzes whether anchors found with high ranks yield the same (or a similar) offset for the sample measurement. This is realized by clustering similar offset results. Figure 6.8 shows an example where two potential offsets $O_{S,1}$ and $O_{S,2}$ were found. $O_{S,1}$ results from two anchors that yielded a similar offset and were thus combined. $O_{S,2}$ results from a third anchor that yields another potential offset. Which of them is the correct sample offset O_S will be determined by performing the correlation for both of them.

We evaluate the performance of the feature-based search algorithm in terms of

speedup and recognition accuracy in comparison to the brute force search in Section 6.5.10.

6.4 Implementation

The implementation of our approach consists of the illuminance measurement on the mobile device and a server-side analysis. We have implemented the measurement part as a website for Android devices and as native apps for Android Wear and Windows Phone 8.1.

6.4.1 Client-side Measurement

The website reads the light sensor via the *Ambient Light Events* API, which is work in progress by the W3C [72]. As of today, Firefox is the only browser that already implements the draft specification. Our website accesses the JavaScript API by registering for *devicelight* events. The event is fired upon registration and whenever the light level changes. Our implementation sends the collected Lux values together with a timestamp to our web server for correlation analysis. We tested the website successfully with Firefox for Android 40.0.3 on Nexus 5, Nexus 7 and Samsung GT I9023.

Firefox is not available for Windows Phone, thus we implemented a native Windows Phone app with the event-based *Windows.Devices.Sensors.LightSensor* API. The implementation is analogous to the website and works successfully on a Lumia 520.

We could not test our approach on iOS devices, because the ambient light sensor is currently not exposed to iOS applications (cf. Section 6.8).

Smartwatches are of special interest for our approach. Since they are typically worn on the wrist they are more likely to be able to measure ambient lighting conditions than a smartphone or a tablet, which are often kept in a pocket or bags. We therefore implemented our approach for Android Wear using the *SensorManager* API. In contrast to the previous implementations smartwatches do not necessarily have direct Internet access. While there are smartwatches with built-in Wi-Fi capabilities, not relying on Wi-Fi broadens the applicability of our implementation. Instead we transfer the measurements via Bluetooth to the paired hand-held device, which forwards them to our web server. We successfully tested this implementation on a Moto 360 Sport.

6.4.2 Server-side Reference Patterns

Our approach requires video reference patterns for comparison with our measured light values. In this section we present an efficient approach to acquire the reference patterns analytically without resorting to manual measurements.

Given an sRGB video frame, we can convert the sRGB color spectrum to the CIE XYZ color space (see Section 6.1) by linear transformation. Y in this case represents the illuminance in the XYZ color space. We convert an sRGB value $\vec{x}_{RGB}$ to Y according to [91] as follows:

$$h(\vec{x}_{RGB}) = \vec{x}_{RGB} \cdot \begin{pmatrix} 0.2126 \\ 0.7152 \\ 0.0722 \end{pmatrix} = Y$$

We define a video frame I to be a matrix comprised by $M \times N$ pixels, each with an sRGB color value. We then calculate the average luminance for an RGB encoded video frame using h:

$$f(I^{M \times N}) = \frac{1}{M \cdot N} \sum_{x=0}^{M} \sum_{y=0}^{N} h(I_{x,y})$$

6.5 Evaluation

We evaluated our approach in various scenarios. Unless otherwise noted, the samples were recorded with a Nexus 5 on Android 5.1.1 bearing an APDS-9930 ambient light sensor. Reference patterns were calculated as described in Section 6.4.2 with a 75 ms sampling interval.

6.5.1 TV Channel Recognition

To evaluate our method we selected 20 real world TV clips representing 20 different TV channels from 7 different television genres (advertisement, concerts, news, series, sports, talk shows, traditional animation). Each video was clipped to 300 seconds and measured with the device screen facing the back wall ($d = 2m, S_1$).

For every sample measurement we calculated the correlation coefficient with every reference pattern. For example, Figure 6.9 shows correlations between one ad-

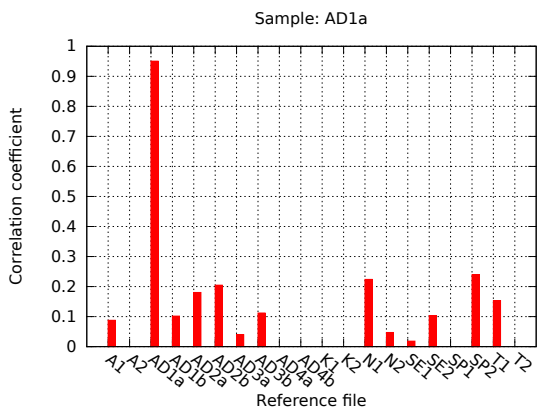


Figure 6.9: Correlation for one sample measurements to various reference patterns.

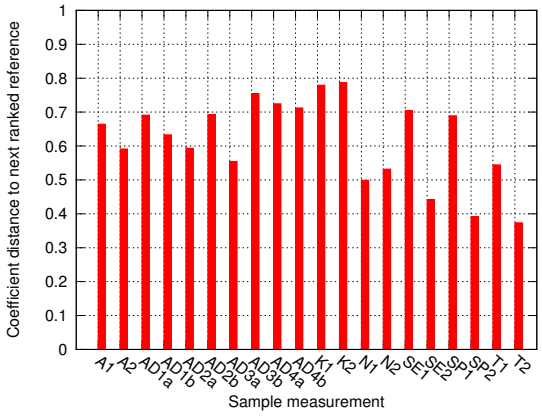


Figure 6.10: Distance to next ranked reference pattern for every sample measurement.

vertisement (**AD1a**) and every reference pattern. The sample correlated highly with the corresponding reference pattern and can be clearly told apart from other candidates. Even videos of the same genre—advertisements (**AD**) in this case—have a significantly lower correlation coefficient. This was also observed for the other sample measurements. We conclude that our method is not susceptible to mixing up videos of the same genres.

The distance between the corresponding reference pattern (0.95) and the next ranked reference pattern (0.26) in Figure 6.9 can be used as an indicator of how likely a video has been recognizer correctly. The higher this value, the lower the probability of a mismatch. To give an overview for the whole video set, we display this difference for every sample measurement in Figure 6.10. The lowest distance (**T2**) originates from a political talk show featuring primarily frontal shots of the discussants. This lack of tracking shots or cuts complicates recognition, showing the limits of our method. Yet we were able to correctly identify every video in this set.

Using the correlation coefficients and distance to second highest ranking video we will introduce a method for determining confidence in recognized videos in Section 6.6.

6.5.2 YouTube Recognition

Video on demand services provide a much larger set of potential videos than regular TV. To evaluate video matching abilities on this scale we composed a set of videos

by crawling YouTube.

6.5.2.1 Popular Videos

We downloaded 1526 unique videos, which were categorized as most popular YouTube videos in 81 regions. The record length is 60 seconds in this analysis, we thus omitted videos shorter than 60 seconds, which yielded 1180 video clips. For each video, we calculated a reference pattern and recorded a sample measurement $(d = 2\mathrm{m}, S_1)$.

65% of the video clips could be identified correctly. Compared to the previous scenario the recognition ratio is rather low. Spot-checking the set revealed this is due to seldom changes in illuminance caused by certain videos types characteristic for YouTube: *a*) Freeze frame videos, i.e. audio only *b*) Freeze frames with occasional text fade-in *c*) Single person speaking to a fixed camera. Since most productions are conducted by non-professionals, cuts are rarely used in the remaining cases.

6.5.2.2 Professional Productions

To evaluate professional video on demand content, we downloaded 200 videos from channels run by public broadcasting organizations, from which 149 remained after filtering for a length of at least 60 seconds. Reference patterns and sample measurements were conducted with the same parameters and in the same setup as before. This time 93% of the videos could be identified correctly.

One could argue that this higher ratio occurs due to choosing a smaller set of videos, since identifying one video out of a small set is easier than identifying one video out of a large set. However, reducing the previous YouTube video set to 149 randomly chosen clips yielded a recognition ratio of just 69% and therefore contradicts this hypothesis. This emphasizes the suitability of our method for professional real world productions as they appear on charged video on demand services.

Table 6.1 shows the distribution of data points in quartiles for each video set. Although the median $Q(0.5)$ does not differ significantly for professional and popular videos, 25% of popular productions have just 7 or less data points. This corresponds to recognition ratios, which is above 50% for all video sets but below 75% for popular YouTube videos. As a comparison, with 29 data points the TV videos set has the largest first quartile which is reflected by its 100% recognition ratio. We conclude that a sufficient amount of illuminance changes is crucial for a successful identification.

	$Q(0.25)$	$Q(0.5)$	$Q(0.75)$	$Q(1)$
YouTube Professional	23	32	43	115
YouTube Popular	7	27	46	155
TV	29	43	49	77

Table 6.1: Distribution of sensor readings per minute of analyzed video sets.

6.5.3 Environment

We now examine how different environmental settings such as distance and orientation towards screen and ambient light affect video recognition.

6.5.3.1 Range and Orientation

Using the video set and its reference measurement from Section 6.5.1 we performed 9 series of measurements by combining each scenario described in Section 6.2 with distances from $d = 1$m to $d = 3$m. Although lower illuminance was measured in less favorable conditions (cf. Figure 6.3) all videos could be recognized. However, we discovered that distance to screen does not necessarily have a negative impact on recorded illuminance. In scenario S_1 our measurement did yield better results at $d = 3$m than $d = 2$m. To understand this effect, we recorded illuminance of a white screen depending on scenario and distance to screen. In Figure 6.11, for S_2 and S_3 illuminance roughly decreases as distance increases. For S_1, this effect is reversed: the higher the distance, the more illuminance has been recorded. Though this happens on a low scale, it has an impact considering that some measurements only consist of 0 and 1 lx. An example for such a measurement is **T1** in Section 6.5.1. Since such measurements could be recognized correctly the measured values are not noise but caused by actual video frames. We conclude that this effect is caused by the white wall situated opposite of the screen. As d' decreases, more reflected light is measured.

6.5.3.2 Light Environment

So far we evaluated our method in an almost dark room at night. Although this is a plausible scenario, being able to recognize videos in a half-light environment increases the practical application of our approach. We therefore evaluated how our method performs in a lit room. The basis for our comparison is the video set and its reference measurements from Section 6.5.1. All samples were performed with $d = 2m$ and in orientation scenario S_2. For each measurement series we increased the ambient light

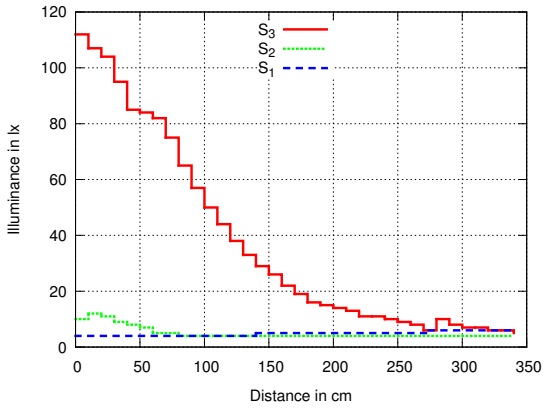


Figure 6.11: Recorded illuminance of a white screen depending on distance d.

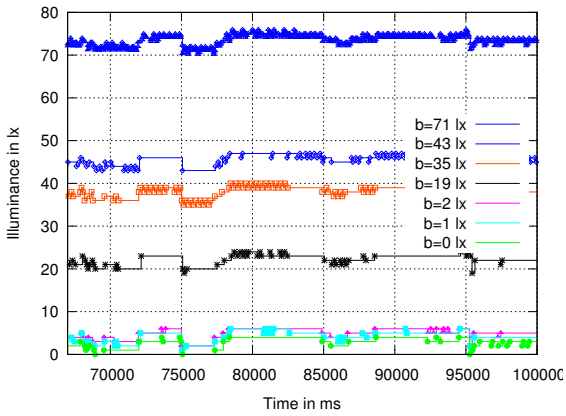


Figure 6.12: The same scene measured with varying pre-existing lighting conditions.

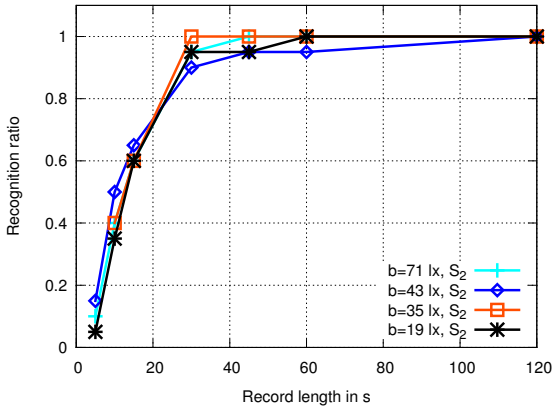


Figure 6.13: Recognition ratio depending on light condition and sample time.

baseline (i.e. with the screen turned off) from $b = 0$ lx to $b = 71$ lx by turning on an additional lamp. The maximum lighting $b = 71$ lx surpasses common living room conditions, which are at 50 lx according to [92].

Figure 6.12 shows the illuminance recorded on one sample video in different light environments. Although environmental ambient light causes sensor readings to be shifted up, relatively they remain nearly identical.

This is also reflected in the recognition ratios: apart from one video, which was mismatched at $b = 1$ lx and $b = 19$ lx, every video was recognized correctly. We therefore reduced the sampling time to check if there is any effect at all. The results can be seen in Figure 6.13. Even in a lit room at night, there is no significant impact on the recognition ratio. We conclude that our method is also applicable if moderate

ambient light is present.

This conclusion is supported by the fundamental properties of light. Physically seen, light and therefore also its illuminance is additive. Even with a very high baseline caused by, e.g., direct sunlight, the diffuse light emitted by the screen adds to the total. However, our method is limited by sensor accuracy; while most sensors detect reliably a variation of 3 lx in dim light, they fail to detect the same variation at a baseline of 100 000 lx.

6.5.4 Record Length

We now analyze the impact of the length of the measured sample on the recognition ratio. The basis for our comparison is the video set of Section 6.5.1 with $d = 2m$ and $d = 3m$ in all orientation scenarios (sample measurements). As stated in Section 6.5.3.1, all clips could be recognized under these conditions. We reduced the length down to a minimum of 5 seconds and observed the change in this ratio.

As shown in Figure 6.14, the minimum time required to achieve a perfect match with $d = 2m$ for every video is 60 seconds for S_2 and 120 seconds for S_1. For S_3, 100% recognition ratio is achieved after 15 seconds of record length.

Although the sensor orientation differs in S_2 and S_1, the recognition ratios are comparable for small record lengths. We conclude that in this case diffuse light reflected from the walls of the room is the primary source of information; only a small fraction of direct light actually reaches the sensor. This is different for S_3, where the mobile device is pitched towards the screen. The sensor records direct light from the screen, which results in a higher resolution and therefore a higher recognition ratio. This suggests that a lack of record length can be compensated by increasing the sensor resolution.

6.5.5 Hand-held Devices

Ambient light sensors approximate illuminance using empirical functions (cf. Section 6.1). To rule out a sensor or device bias, we conducted measurements with a device set consisting of Nexus 5, Samsung GT I9023, Nexus 7 (2013) and Lumia 520. To illustrate differences in perceived sensor readings, Figure 6.15 shows illuminance recorded by various devices in scenario $d = 2m, S_3$.

As we can see, the devices' sensor readings vary both in magnitude and frequency. The frequency is the result of sensor quality and sampling frequency of the operat-

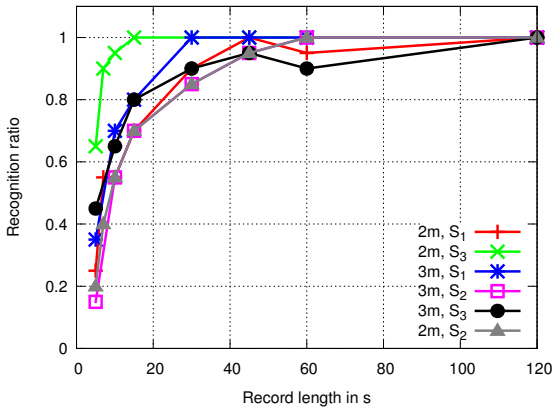


Figure 6.14: Recognition ratio depending on setup and sample time.

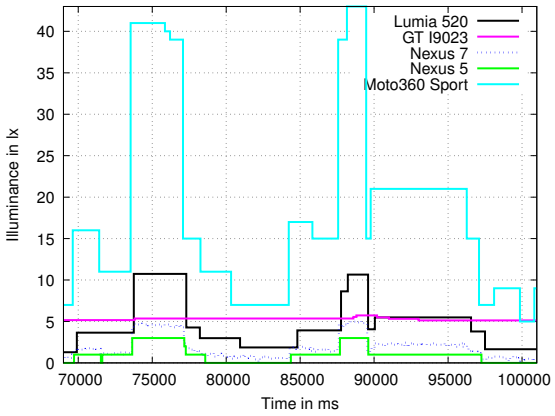


Figure 6.15: Recorded illuminance of the same scene by various devices.

ing system. The deviation of magnitude is caused by sensor-specific calibrations and accuracy as mentioned in Section 6.1. Since our method correlates changes of illuminance, the absolute sensor reading is of lesser importance. However, there is a lower bound: If the sensor resolution is too low to detect changes in illuminance, there will be too few values for the correlation analysis and our method will fail.

An example for this effect is visible in the interval from 70 000 ms to 80 000 ms in Figure 6.15. All devices except GT I9023 show a peak in illuminance followed by a local minimum. Since the effect is not limited to this particular measurement excerpt, it also has an impact on the recognition ratio: Given the video set from Section 6.5.1 the readings from Lumia 520, Nexus 5 and Nexus 7 yielded a 100% recognition ratio. Measurements from GT I9023 caused a ratio of 60%. This may be the result of technological advancements of ambient light sensors. The moderately performing GT I9023 was released in 2011 while the other smartphones were released in 2013.

6.5.6 Smartwatches

A smartwatch provides additional light sensor readings in case the smartphone's ambient light sensor is obstructed. A challenge of utilizing smartwatches in contrast to smartphones and tablets is their power constraint. To increase their battery life, smartwatches rely heavily on energy-saving measures, which put components into a low-power state. The ambient light sensing interval is reduced considerably compared to smartphones. For instance, the Moto 360 Sport smartwatch provides a new illu-

minance value every 10 to 20 seconds, which is not enough for recognizing videos of a few minutes length.

To get sensor readings more frequently, our Android Wear implementation acquires a wake lock to temporarily disable power saving measures. We evaluated whether this allows for an adequate video recognition by conducting sample measurements of the YouTube Professional video set (149 videos, each 60 seconds long) with the Moto 360 Sport smartwatch. We chose $d = 2m, S_3$ as recording scenario, unlike S_1 for smartphones in Section 6.5.2.2, because we assume a smartwatch is more likely to be facing towards the screen.

Our approach yielded a recognition ratio of 98%, which is even higher than our previous evaluation (93%) using a Nexus 5 in Section 6.5.2.2. This is either caused by an improved sensor or by the more favorable orientation. To tell these possibilities apart we measured the video set in $d = 2m, S_3$ with the Nexus 5. Again, a recognition ratio of 93% could be observed. Therefore the higher recognition ratio is caused by the smartwatches sensor and not the difference in orientation towards screen.

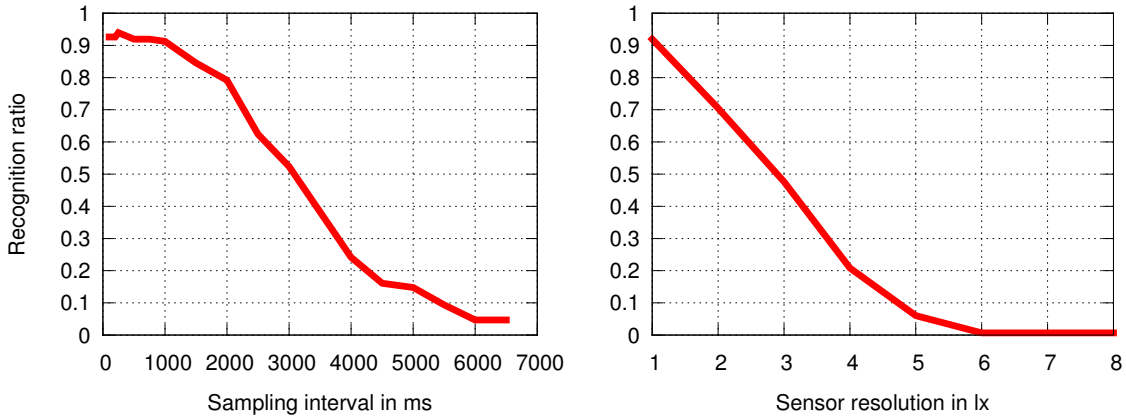
These results emphasize the suitability of smartwatches for our approach. However, a continuous illuminance measurement on the smartwatch is impractical due to battery constraints. Thus, we propose to use smartwatches as components actively queried by the user. For example, if the user wants to acquire additional information to the show he is currently watching, he could launch an app on his smartwatch which records illuminance on demand. That way the battery consumption is kept at a minimum.

6.5.7 Sensor Limits

As we identified in the previous sections, the sensor resolution is crucial for our approach. To analyze this dependency in a systematic manner, we truncated the sensor readings collected in Section 6.5.2.2 and observed the impact on the recognition ratio. The device used in that measurement provided a sampling rate of about 100 ms and had an accuracy of 1 lx.

6.5.7.1 Sampling Rate

First, we truncated the sampling rate, i.e. the minimum time between two sensor readings. Figure 6.16a shows that reducing the sampling rate has a non-linear negative impact on the recognition ratio. Although recognition decreases sharply for sampling



(a) Sensor sampling rate.

(b) Sensor resolution.

Figure 6.16: Effect of sensor quality on recognition ratio.

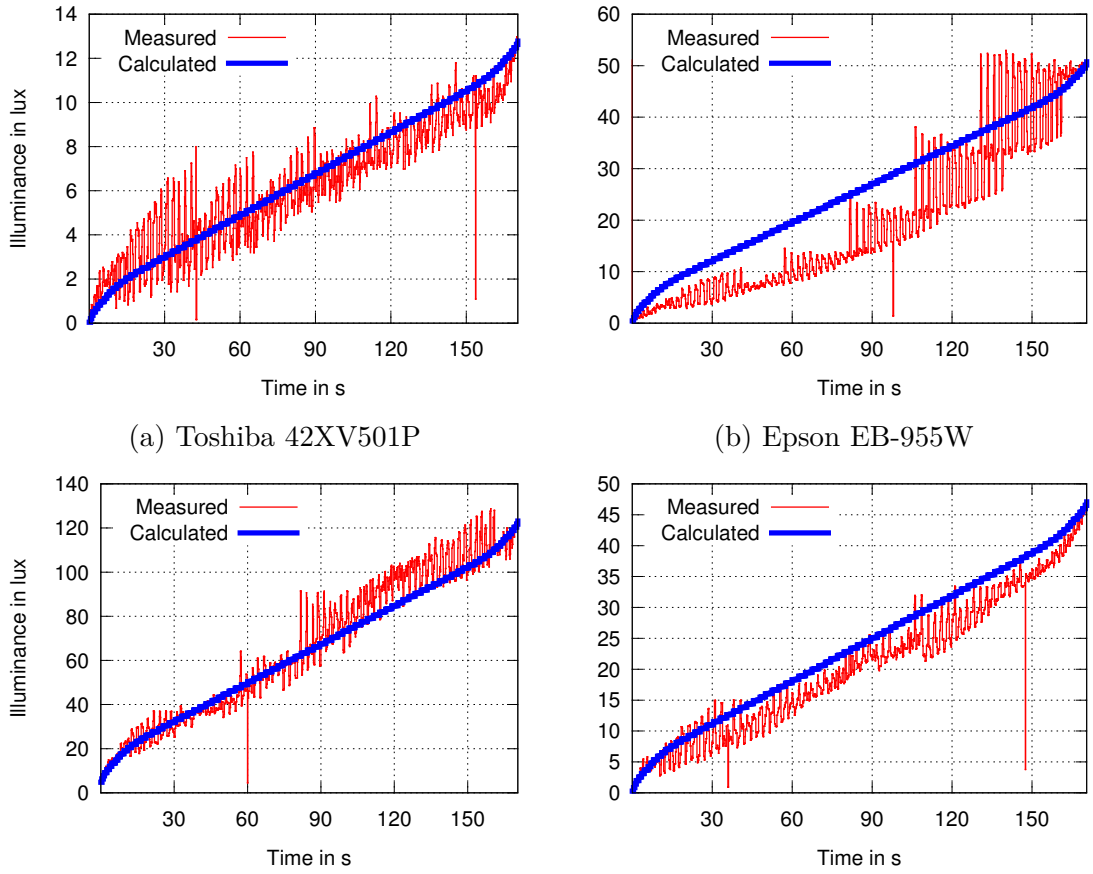
intervals $> 2000ms$, we do not consider this to be problematic since it has little real world impact: All ambient light sensors examined in this section provide sampling rates of $\ll 1000$ ms. This yields recognition ratios of at least 88%.

6.5.7.2 Sensor Resolution

We emulated a low sensor resolution by rounding measured illuminance down to multiples of the emulated granularity. The impact on recognition ratio can be seen in Figure 6.16b. As before, this has a negative impact on recognition ratio, though the effect is stronger: the recognition ratio falls below 43% when truncating the sensor resolution to ≥ 3 lx. The reason for this is that in most cases measured ambient light changes only vary within a range of 3 lx, as you can see for example in Figure 6.12. In these cases a sensor resolution of ≥ 3 lx would hardly yield any detected ambient light changes.

6.5.8 Reference Illuminance

One way for obtaining illuminance values is to analyze videos (cf. Section 6.4.2). To evaluate the suitability of this approach, we compared analytically derived illuminance with actual measured illuminance. For this, we evenly sampled the RGB color space and sorted the resulting colors by their calculated illuminance. We displayed these colors on three screens and one projector and recorded resulting illuminance on a Nexus 7 (2013). We chose this tablet since an accurate representation of illuminance is crucial for this approach and it bears the best sensor in our device set. The results



(a) Toshiba 42XV501P

(b) Epson EB-955W

(c) Samsung UE65F6470

(d) Toshiba 47WLG66

Figure 6.17: Analytic vs. measured illuminance of colors.

are presented in Figure 6.17. Since our method implicitly normalizes input sequences only relative changes are relevant. We therefore linearly transformed the calculated illuminance to fit the measured data.

As we can see, there are deviations from our calculated values depending on the TV screen. This occurs due to device-specific color rendering. To evaluate whether our method is capable to succeed despite color deviations we used the video set from Section 6.5.1 and recorded one minute long measurements. These measurements were compared with analytically derived values.

The recognition ratios were 80% for 6.17a and 6.17d, 95% for 6.17b and 100% for 6.17c. These ratios imply that recognition ratio does not correlate with color accuracy: the projector in 6.17b has the largest color deviation but ranks second in recognition ratio. Instead, illuminance is determining. The brightest display devices have the highest recognition ratios. This is reasonable because for a constant sensor

resolution a larger spectrum of illuminance can be represented by more sensor values leading to a more accurate representation.

We conclude that analytically derived illuminance provides a suitable estimation of actual measured illuminance.

6.5.9 Brute Force Offset Determination

The brute force search algorithm from Section 6.3.3.1 determines the sample offset by testing several offset values. The choice of the *step size* S_{step} is a trade-off between efficiency and effectiveness. To determine an appropriate S_{step}, we took 20 reference videos (see Section 6.5.1) and their corresponding samples and analyzed how big the difference between the correct sample offset and the correlated offset might be while still yielding valid correlation results. The analysis showed that the correlation yields a correlation coefficient that is less than 5% lower than the optimal correlation coefficient, as long as the offset difference is within a range of ±111 ms of the optimal offset. Hence, 222 ms is a feasible step size to be at least once within a range of 111 ms of the optimal offset. We therefore performed the brute force correlation by correlating various offsets within the brute force range $R_{BF} = 2000$ ms with a step size S_{step} of 222 ms.

6.5.10 Feature-based Offset Determination

In Section 6.3.3.2 we proposed a feature-based algorithm to determine potential sample offsets O_S of sample measurements within a reference video pattern. We now demonstrate that the algorithm improves the correlation efficiency while maintaining the recognition effectiveness. To evaluate the algorithm, we created a 2 400 second reference video by concatenating the 8 advertisement reference videos from Section 6.5.1. Furthermore, we created 80 samples with a length of 30 seconds each by splitting the 2 400 second. We configured the algorithm to use a derivation window D_W of 600 ms, a fuzziness threshold T_F of 195 ms and a look-ahead threshold of $T_L = 1$.

To analyze the reduction of correlations necessary to recognize a video, we performed the sample offset search for all of the 80 samples within the 2 400 second reference video. Each result list containing all potential sample offsets O_S was sorted by rank. The result lists contained 170 potential offsets in average (615 offsets in worst case) that need to be correlated. In comparison, searching the 2 400 seconds with the brute force algorithm exhaustively would require correlating 10 810 offsets

for a step size of 222 ms (see section 6.5.9). This is a reduction of about $98,4\%$ in average and $94,3\%$ in worst case.

Each result list contained the correct sample offset O_S, thus every video could be recognized correctly. The correct offset O_S was in average contained within the top 3% highest ranked sample offsets. This prompts the conclusion that it might not be necessary to correlate the sample and the reference video for every sample offset O_S in the result list to further speed up the correlation process. However, we cannot rely on this to be always true. For example in the worst case in our analysis the correct offset O_S was only in the top 77%. Thus, reducing the correlation to the top result list entries would reduce accuracy. This is, because in rare cases the correct sample offset O_S may receive a low rank and would be discarded. Thus the video would not be recognized.

The search for a single sample took 22.8 ms in average (with 9 ms being the fastest and 42.3 ms being the slowest search). To quantify the achieved speedup in comparison to the brute force search, we estimated the time E_B necessary for searching the sample offset exhaustively as well as the estimated time E_S for searching for the sample offset using the feature-based algorithm. We estimate the speedup factor F_S using the following formulas:

$$E_B = C_t \cdot \frac{D_R - D_S}{S_{\text{step}}}$$

$$E_S = S_t + C_t \cdot T_{\text{size}}$$

$$F_S = \frac{E_B}{E_S}$$

C_t is the time necessary to perform a single correlation for one offset. Our unoptimized proof of concept Python prototype requires 2 ms on a commodity laptop to perform this step for two 60 second videos from the set used in Section 6.5.2.2. Hence, we set C_t to 1 ms, since the sample videos are 30 seconds long. D_R and D_S are the duration of the reference video (i.e. 2 400 seconds) and of the sample video (i.e. 30 seconds). We use 222 ms as step size S_{step} for the brute force search. S_t is the average time necessary for searching for potential sample offsets O_S, i.e. 22.8 ms. Furthermore, T_{size} denotes the result list size, which was 170 potential sample offsets in average and 615 potential sample offsets in worst case for our measurements. This yields a speedup factor F_S of the feature-based offset search of $55.4\times$ in average and

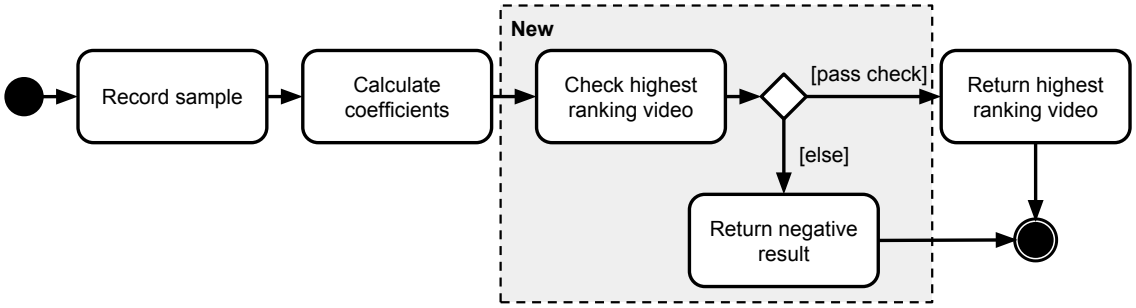


Figure 6.18: Recognition process with confidence estimation.

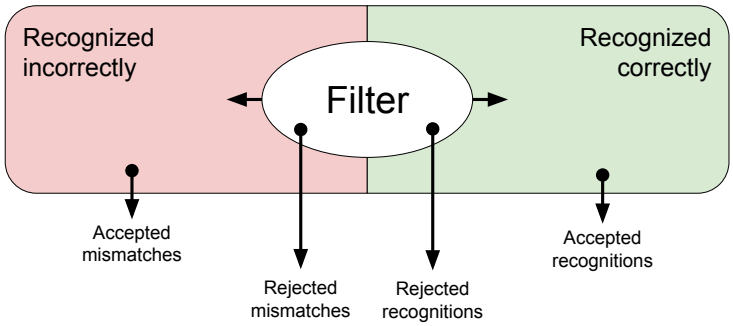


Figure 6.19: Visualization of recognition outcomes.

16.7× for our observed worst case.

6.6 Confidence Filtering

So far our approach considered the highest correlating video as the matching reference pattern. However, as we have seen in Section 6.5.2.1, videos with a low amount of illuminance changes are hard to recognize and the risk of mismatching is high. We therefore propose a filtering step, shown in Figure 6.18.

6.6.1 Approach

Fundamentally, instead of always nominating the highest ranking video as the video being played back we add the possibility of a negative result in case there is a lack of confidence. As a side effect, this negative result will also be provided if no video has been recorded, i.e. by sensor obstruction.

Using the correlation coefficients and the difference between two highest ranking correlation coefficients has been proposed in Section 6.5.1. While this works well for

the video set discussed in that section, it is of limited use if there are insufficient data points. We take this into consideration by proposing to filter all video candidates below a certain data point count (t_p), correlation coefficient (t_c) or coefficient difference (t_d).

6.6.2 Threshold Determination

To determine these thresholds empirically, we re-evaluated the three video sets discussed previously. More specifically, we distinguish between *accepted recognitions*, i.e. correctly recognized videos which were not filtered, *accepted mismatches*, i.e. incorrectly recognized videos which were not filtered, *rejected mismatches*, i.e. incorrectly recognized videos which were filtered, and *rejected recognitions*, i.e. correctly recognized videos which were filtered. Figure 6.19 shows this distinction in a Venn diagram of these categories. The filter quality depends purely on the chosen thresholds. As an optimization goal, we aim to minimize the sum of accepted mismatches and rejected recognitions. While maximizing the post-filter recognition ratio might seem like a more intuitive approach this would lead to a minimal amount of candidates passing the filter, jeopardizing practical usage.

6.6.3 Evaluation

In the YouTube Popular video set, all 8 optimal thresholds were within $t_p = 5$, $t_c \in [0.540, 0.548]$ and $t_d \in [0.045, 0.046]$. The minimal sum of accepted mismatches and rejected recognitions amounts to 126. Depending on the actual thresholds, between 440 and 444 out of 476 actual mismatched videos could be identified as *rejected mismatches*.

Before we applied this filter, the video set had a recognition ratio of 65% and we identified insufficient data as the main cause for this (c.f. Section 6.5.2.1). Our new approach increases this significantly. Of the videos passing the filter, a recognition ratio of 95% is achieved.

The effect on the YouTube Professional video set is moderate since the initial recognition ratio was already high. Optimal results (accepted mismatches and rejected recognitions summed up to 7) were achieved with $t_p \in \{6, 10\}$, $t_c = 0.430$ and $t_d = 0.009$. The recognition ratio increased from 93% to 96% post-filter for $t_p = 6$ and to 97% for $t_p = 10$.

No effect could be observed for the TV video set, where the recognition rate remained at 100%. Our optimization approach found an optimum of 0 filter mistakes

Video Set	Accepted		Rejected	
	Mismatches	Recognitions	Mismatches	Recognitions
YouTube Popular	38	614	438	90
YouTube Professional	4	128	8	9
TV	0	20	0	0

Table 6.2: Threshold results for $t_p = 5$, $t_c = 0.311$, $t_d = 0.045$

for $t_p = 18$, $t_c = 0.311$ and $t_d = 0.126$.

Each video set yielded a different set of optimal thresholds. To analyze the suitability of a common set of thresholds we created a new filter of the minimal thresholds ($t_p = 5$, $t_c = 0.311$, $t_d = 0.045$) and applied it to each video set. Table 6.2 shows the results. Compared to the individual optimal settings, these common thresholds show a minor decrease in quality: YouTube Popular's rejected mismatches are reduced by 0.5% and YouTube Professional accepted recognitions dropped by 4.5%. However, the filter does still have a positive impact compared to our initial approach without confidence consideration: YouTube Popular yielded a recognition ratio of 94% (previously: 65%) and YouTube Professional of 97% (previously: 93%). Again, the YouTube Popular video set shows a substantial improvement due to its low entropy content. This suggests that filtering results subject to an estimated recognition confidence improve the overall recognition ratio despite occasional misclassifications.

6.7 Feasibility

So far we have evaluated our method in various scenarios. In this section we will discuss real world challenges for our approach.

6.7.1 Automation

Since measuring reference patterns for all videos is a cumbersome task, the server can analyze each video entirely in software and map it to a brightness scale (cf. Section 6.4.2). By abolishing the need for physical measurements, this increases the mass-scale practicability of our method.

To evaluate the quality of calculated reference patterns we used the professional YouTube video set from Section 6.5.2 and created reference *measurements* in nearly ideal conditions ($d = 1\text{m}, S_3$). We applied our method on these reference and sample

measurements. This time a recognition ratio of 92% could be reached, which is slightly below the recognition ratio yielded from calculated reference patterns in Section 6.5.2. This suggests that calculated reference patterns are more suited for our approach than reference measurements.

6.7.2 Network Load

Since a video is typically played at 25 FPS a sensor sampling rate below 40 ms does not yield better results. Although illuminance *can* change for every frame this is seldom in real world settings (cf. Figure 6.3 and Figure 6.12). If we assume this as an upper bound and represent a sensor reading with two bytes, our method will require at most 3,000 bytes for one minute of video footage.

Compared to streaming camera footage to a server, our approach requires a significant lower amount of data. In [120] current state of the art video encoding required a minimum of about 0.05 Mbps for a 480p video of average subjective quality. Put into perspective with our approach this requires about 131 times the amount of data to be transmitted. We therefore conclude that our approach is feasible with respect to network load.

6.7.3 Server-side Load

Given a sensor measurement, the server has to a) maintain its reference set and b) compute correlation with this set.

The first step requires the server to sample all TV channels at a specific rate and calculate illuminance using the method shown in Section 6.4.2. As shown in Figure 6.16a, it is sufficient to sample every 250 ms to achieve a high recognition ratio. Also, a video stream with low resolution suffices for illuminance determination easing CPU requirements. Our prototype implemented in a browser environment requires 10%-20% CPU usage on consumer grade hardware for real-time illuminance conversion. For on demand videos this conversion can be performed lazy whenever new videos are added to the catalog. We therefore deem it feasible to analyze a large amount of TV channels and video on demand catalogs using dedicated servers.

The actual server-side load for correlation depends on whether the server performs a live correlation or a deferred correlation. For live correlations the load depends on the expected content propagation delay range and thus the brute force range R_{BF} (see Section 6.3.3.1). In our measurement, computing a range of 4 seconds with brute

force took about 36 ms for one sensor measurement and one reference pattern. This needs to be done for every potential reference pattern. Once the content propagation delay has been determined for a specific user, this information can be reused. In this case, the offset search only needs to compensate for jitter and thus the brute force range R_{BF} can be reduced further.

For deferred correlations the load depends on the length of reference patterns (see Section 6.3.3.2). In our measurement it took 362.8 ms to find a sample measurement within a 2 400 second reference video. This needs to be performed for every potential reference pattern. While this takes longer than identifying a TV channel, we do still consider this to be feasible.

6.8 Privacy Considerations

Our approach utilizes the ambient light sensor whose data is less privacy-invasive than, e.g. camera or microphone recordings. Nevertheless, learning about the user's context violates their privacy when collecting data without consent or when using the data in ways unexpected by the user. In this section, we discuss countermeasures and survey the permissions required for reading the ambient light sensor on various platforms.

6.8.1 Truncation

The purpose of the ambient light sensor is to dim the screen to ensure visibility in bright daylight while conserving battery and avoiding eye fatigue at night. Adjusting the screen brightness is handled by the operating system. The sensor Lux values are exposed to applications to give them the opportunity of adjusting the appearance of the user interface, e.g., switch to a darker theme. The operating system could limit the potential for context inference without consent by truncating the sensor readings.

We have shown in Section 6.5.7 that our approach is still feasible with a sensor sampling interval of 1 seconds. Truncating the sensor sampling interval to 5 seconds makes our approach infeasible, but also makes the sensor readings useless for mobile devices, whose light conditions change quickly when taking the device out of the pocket or opening the protective cover. Another dimension for truncation is the sensor resolution. Our approach becomes infeasible when the sensor readings are rounded to levels of 5 lx, yet this resolution is more than adequate for adjusting the screen

brightness of user interfaces. In fact, rounding to three or five different illuminance levels should suffice for most applications. The W3C Media Queries [100] for example provide a three-level (*dim, normal, washed*) ambient light reading to applications. The additional advantage of a qualitative classification is that devices can use different illuminance thresholds to account for technological characteristics—e-ink displays are for example better readable in sunlight than LCDs.

6.8.2 Permissions

Running on Firefox for Android, our web implementation does not need to request user permission when accessing the ambient light API. The light event is fired on the active tab only and not on background tabs, iframes or when the screen is turned off. This behavior has been specified in the API draft [72], which also recommends to consider an indication to the user when the sensor is active and to allow turning the sensor off. Enabling the user to notice and control the sensor readings is a worthy idea, but not trivial to achieve given the magnitude of active sensors besides the light sensor on today's mobile devices and the limited space on the screen for visual indicators. Requesting user permission before sensor access is not intended in the API draft, unlike, e.g., in the geolocation web API [94].

Sensor data can be collected in background when implemented as native app. Android exposes the ambient light sensor as Lux value since Android 1.5 [111]. Background sensing is possible even when the screen is locked or turned off. There is no permission required and no user indication of an active sensor. The same is true for Android Wear: No permissions are necessary besides network access and keeping the device awake. There is also no permission required to run applications on the smartwatch. Whenever an app bundled with a smartwatch component is installed on the paired hand-held device, this component is pushed to the watch without further actions from the user.

A similar case is Windows Phone 8.1, which allows apps to access the sensed Lux value without permission or user indication. Background sensing can be implemented via the *DeviceUseTrigger* API.

iOS does not provide the ambient light sensor readings to applications. There is an unsupported, private API in iOS, but third-party apps are denied access to private APIs in the operating system.

6.9 Related Work

Our approach allows to infer context about the high-level activity of the user (watching TV) from a low-level sensor reading (ambient light sensor). The fundamental difference to image-based video fingerprinting [90, 70, 75] or audio fingerprinting techniques [53, 23] is that we do not need to use the camera or microphone, which has advantages in terms of power consumption, CPU and network usage and usability.

Utilization of the ambient light sensor for context-aware computing has had little attention in the literature compared to other sensors. Ravi and Iftode [98] suggested to use the light sensor for fingerprinting room lighting conditions for the purpose of indoor localization. Li et al. [76] suggested to use visible light communication for indoor localization: a light sensor attached to a smartphone receives location beacons, which are broadcasted by modulating white light-emitting diodes (LED) in the room. Visible light communication allows for accurate sub-meter localization but the LED modulation requires a more sophisticated light sensor with high-frequency sampling.

Several researchers suggested to extract ambient color and illuminance from camera images for the purpose of indoor localization [2, 24, 97, 4]. The illuminance feature could be gathered with the ambient light sensor in today's mobile devices. Color sensing is available on a few devices such as Samsung Galaxy S3 with CM36651 sensor, though most devices measure the illuminance only.

Spreitzer [117] demonstrated a potential side channel attack by exploiting the ambient light sensor: after a training phase, a malicious app could use the ambient light reading to infer information about probable keystrokes, e.g., to guess personal identification numbers. This emphasizes our demand for sensor truncation unless the user consents to ambient light sensing.

Some of the above approaches employ machine learning techniques while we compare raw signals. In order to utilize machine learning effectively, a TV channel would have to be identifiable by a certain learnable feature. However, our evaluation in Section 6.5.1 does not show evidence for a genre-specific or channel-specific correlation. Therefore a machine learning approach would have to be trained continuously with the current patterns from all channels which is similar to our approach. A more fitting use case for machine learning could be categorizing an unknown video in genres based on their illuminance changes, which is out of scope of this work.

Besides the weighted Pearson method used in our approach there are various ways to match signals. A common method is cross-correlation which correlates two

time shifted series $x[n]$ and $y[n]$ as a function f of displacement m with $f_{x,y}(m) = \Sigma_{i=0} x[i]y[i+m]$ [82]. For application in our use case a preprocessing step would have to map ALS time series to a one dimensional sequence by sampling values at a constant rate. Afterwards a normalized cross-correlation could be used to find a maximum across all video candidates.

In the field of bioinformatics the problem of aligning nucleotide sequences has sparked several dynamic programming solutions. Due to mutations a perfect match of two sequences is unlikely which poses the challenge of finding an alignment with a minimum of deviations. The most notably solutions are the Needleman–Wunsch algorithm [87] for finding a global optimal alignment and the Smith–Waterman algorithm [116] with a focus on locally optimized alignments. Both approaches utilize a matrix where rows correspond to one sequence and columns to the other. It is filled with a score representing the likelihood of a correct alignment. The metrics for similarity in these algorithms incorporate biological domain knowledge, i.e. the likelihood of certain mutations. To be used in our approach, this scoring penalty would have to be replaced with an illuminance metric. Additionally ALS readings would have to be sampled as in the cross-correlation case.

A different approach for signal comparison consists of considering the frequency domain by Fourier transforming the ALS time series. In the frequency domain, convolutions of time series are equal to cheaper multiplications. This can be used since cross-correlating time series can be expressed as a convolution with one of the series complex conjugated. [17] This method is usually motivated by its speed advantages. [64] For application in our approach, the low entropy of ambient light readings and short sampling times could prove to be challenging and would require an evaluation. However as our focus lies on showcasing the feasibility of video recognition, performance improvements are outside of the scope of this work.

6.10 Conclusion

We presented an approach for recognizing a video playing in the user's proximity by analyzing the ambient light sensor readings of mobile devices. Our method correlates the characteristic video flickering with reference illuminance values from a set of known videos. The video sample can also be identified correctly if it consists only of a fraction of the reference video. These reference patterns can either be obtained by a reference measurement or by calculating them from a video stream. We have tested our

approach successfully on several mobile devices in a typical living room scenario. Our evaluation shows that the recognition works well for television channels (100%) and professional YouTube content (93%), yet moderately for amateur YouTube content (65%). The discrepancy is caused by the amount of video cuts or sudden light changes, which are more frequent in professional video productions.

By performing a confidence assessment of the recognized video, the amount of mismatched videos could be reduced significantly: only 5% of the amateur YouTube video recognitions (previously 35%) were incorrect.

Although our method works best at night, room lights do not have a detrimental effect on the recognition ratio as long as the mobile device is not pointed directly into the light source. We argue that existing ambient light in general can be compensated by a higher sensor resolution. Vital parameters for successful video recognition are the measured record length, sensor resolution and sampling rate. A recognition ratio of about 90% is achieved with 7 seconds samples when having a direct sight from sensor to screen and with 30 seconds samples when pointing the sensor away from the screen (e.g. holding the mobile device). Sensors with higher resolution achieve better recognition ratios with a given record length, where a resolution of 1 lx recognizes 90% of professional videos. The sampling interval is suitable on all tested devices and can be even reduced to 1 second to save battery power without severely degrading the recognition ratio.

To reduce the amount of correlations necessary to find the offset of a sample within a reference pattern we proposed a new optimized method. Compared to using a brute force search 98.4% less correlations need to be performed yielding a speedup of a factor of 55. This shows suitability of our approach for video on demand recognition.

The Lux values collected by the ambient light sensor are exposed to applications on Android, Android Wear, Windows Phone and Firefox for mobile devices. Permission for accessing the sensor values is not required, which bears the privacy risk of leaking the user's context without prior consent. Our recommendation to system implementers is to truncate the sensor Lux values to predefined light levels, which suffice to adjust the user interface subject to the ambient light environment. Applications should be required to ask for permission when exact Lux values are needed, e.g., to support the user with contextual information derived from our video recognition approach.

For future work, required record length could be decreased and the recognition ratio further increased with improved ambient light sensors. High-resolution RGBW

sensors could allow sensing at daylight or recognizing slight changes of illuminance, e.g., in amateur videos.

Chapter 7

Mobile Devices as Digital Sextants for Zero-Permission Geolocation

The previous chapter showcased a privacy violation by determining the video content being played in the user's vicinity. An even more severe privacy violation would consist in locating the user using unrestricted sensor readings.

Geolocation determines the location of a user's device in the world. Positioning or geolocating services are a standard feature offered by smartphones and other mobile devices. They achieve a high accuracy by combining satellite navigation (e.g., GPS, Galileo or GLONASS), cell tower and Wi-Fi triangulation.

The location of a mobile device is synonymous to the location of its user and therefore subject to privacy concerns. Thus, operating systems like Android or iOS, or platforms like web browsers ask for user permission before giving location access to applications.

Unfortunately, adversaries can utilize side channels to determine the user's location without consent. A prominent example is IP address geolocation, which allows country-level or city-level geolocation [124]. Hiding the IP address protects from this type of geolocation, for example with a VPN, proxy server or Tor. Prior work has shown that freely accessible inertial sensors provide enough information to infer the trajectory of moving mobile devices such as acceleration of metro lines [59] or cars in city streets [55]. Even the phone's power meter can be used to infer the location based on cellular radio power consumption [83]. This enables the geolocation of the user even when hiding the IP address with a VPN. These approaches have in common that they require a systematic charting of a given area subject to specific features before geolocation becomes possible.

In this chapter, we propose a zero-permission geolocation method based on mobile sensors, which does not require prior charting nor user permission. We determine the altitude of the sun by creating a digital sextant based on the ambient light sensor and *accelerometer*. Combined with a compass, this allows for geolocation anywhere on Earth where the sun is visible without prior training or cartography. However, *magnetometers* in smartphones used as compasses are too inaccurate for this purpose, especially because we cannot expect the user to calibrate them when they are unaware of being geolocated. Our method compensates for magnetometer deficiencies by inferring the sun's movement with time-delayed measurements, from which we can derive the user's location on Earth.

The sensors used by our method are accessible by Android apps or even websites on Firefox for Android without requesting user permission. Major challenges are inaccurate sensor readings by uncalibrated sensors on consumer devices. Yet, our web-based implementation achieves an accuracy below 500 km in 50% of our measurements. This shows that malicious websites can perform a country-level geolocation even when the user employs a VPN to hide their location.

The contributions of this chapter include:

1. A novel approach to locate mobile devices without any infrastructure support.
2. A systematic evaluation of its accuracy, depending on daytime, latitude and sensor error.
3. An analysis of countermeasures based on reduced sensor resolution.

This chapter is organized as follows: Section 7.1 motivates an attack use case. Section 7.2 provides astronomical background that the attack relies on. Section 7.3 describes assumptions about the threat model and Section 7.4 explains the geolocation method. Section 7.5 describes the implementation and how to handle practical interferences. Section 7.6 evaluates the location accuracy and causes of measurement errors. Section 7.7 discusses countermeasures and their effectiveness. Section 7.8 compares our work with related approaches.

7.1 Use Case

Our digital sextant achieves an accuracy suitable for country-level geolocation. Although this is a coarse result, it can be used as part of a multi-level approach to

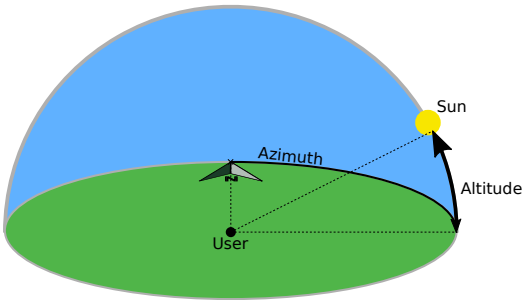


Figure 7.1: The horizontal coordinate system.

bootstrap a more accurate geolocation method. There is a number of approaches for location tracking, which require either the starting point or at least the approximate area of where the user resides [77, 36, 48, 49, 133, 55]. Based on an approximate position, they allow to infer vehicular movement on a street map or similar approaches. Without any prior areal indication at all, there are too many potential matches and the resources required to process global map data renders the geolocation attempt infeasible. Our method thus yields the approximate area of the user, which can be then narrowed down with a computationally-expensive method to a specific location.

7.2 Background

Our method is based on knowledge about planetary movements and celestial navigation combined with sensors available in smartphones.

7.2.1 Celestial Navigation

Celestial navigation relies on knowledge about perceived positions of celestial bodies depending on observation time and place on Earth. To describe celestial positions from a local observer's perspective, astronomers use the horizontal coordinate system (Fig. 7.1). In it, every celestial object can be defined using two angles: *altitude* describes elevation from the observer's local horizon and *azimuth* the clockwise angle between north and that point on the horizon below the celestial body.

Combined with a precise location information of the observer (latitude/longitude) and time of the observation, this uniquely defines a celestial position and allows for identification of that celestial body. In the opposite case, if horizontal coordinates, time and celestial body are given, the observer's location is uniquely defined, which

we utilize in our method.

7.2.2 Planetary Movements

While the general model of Earth orbiting the sun is well-known, various factors have to be taken into consideration to precisely predict its celestial position. Among these are ecliptic (tilt between rotational axis and orbital axis), exact orbital period (leap years/seconds), gravitational influences, day of year and time of day. Even then, there is no universal formula to describe celestial bodies precisely. Instead astronomers rely on fundamental ephemeris (i.e., tables of positions of celestial objects and their movements) which can be used to predict future positions. These calculations can be performed by broadly available astronomic programming libraries such as PyAstronomy [30] with high accuracy.

7.2.3 Smartphone Sensors

Due to its brightness the sun is especially suited for detection using an ambient light sensor (ALS). Usually, ALS are mounted on the mobile device's front near the camera and prefaced by a lens to sample ambient light from a wide angle. Silicon photodiodes used in ALS are sensitive to a broad spectrum of light. To approximate human perception of illuminance and restrict that sensor sensitivity to visible light, filtering techniques are used to ultimately yield illuminance in the photometry unit Lux. Nearly every smartphone and tablet is equipped with an ALS to adjust screen brightness, allowing for a sensor-based trade-off between screen readability and energy consumption. Besides its user interface usage, ALS readings are also exposed to applications on Android and in the web API without any dedicated permission requirements. Both APIs yield illuminance as IEEE 754 floating points.

7.3 Threat Model

As illustrated in Fig. 7.2, a user holds their smartphone or another smart device while being exposed to direct sunlight. The user takes reasonable measures to hide their location such as using a VPN and thereby rendering IP address-based geolocation useless. Unconscious movements by the user cause the smart device to be facing in various directions.

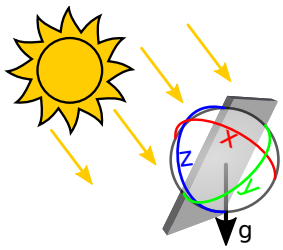


Figure 7.2: The mobile device records ambient light and positional sensors.

An attacker aims to geolocate the user without their consent. The attacker can execute code without special permissions on the user's device, in particular without access to the geolocation API. The malicious code reads the ambient light sensor, accelerometer and magnetometer of the user's device. This assumption is met by installing a seemingly harmless app or simply visiting an HTML5 website, since the sensors of interest to our method are exposed via web APIs [123, 68].

We furthermore assume the attacker runs another measurement on the user's device after one to six hours and observes a different position of the sun. This could be achieved by running the app in background or because the user visits the website a second time.

7.4 Method

Our method to locate a mobile device consists of 1. measuring directional ambient light at two points in time, 2. processing these measurements to find out the sun's altitude and azimuth, 3. calculating location candidates and 4. finally aggregating them to a position. We will discuss each step in detail in the following sections.

7.4.1 Measurements

We start by continuously reading accelerometer, magnetometer and ambient light sensors with corresponding time stamps. Since Android apps or websites do not require additional permissions for this, this data collection can be conducted without the user's permission or even awareness.

Since our approach does not control the user's movement, we merely assume the ambient light sensor eventually points towards the sun. We refer to this set of collected sensor readings as one *measurement*. To overcome inaccurate magnetometer values,

we use at least two measurements from different points in time, which will be merged together in Section 7.4.3.

7.4.2 Preprocessing

Given a measurement, we want to locate the sun and calculate its altitude and azimuth.

For this, all sensor readings are transformed to a horizontal coordinate system. The accelerometer presents its values as a three dimensional vector (x, y, z) where z represents the front/back forces acting on the device. Assuming gravitational force is the main component, we can use the normal vector of the xy-plane to calculate the device's facing direction, which gives us the altitude in the horizontal coordinate system (cf. Fig. 7.1). When interpreted as a compass, magnetometer readings can be used to determine the azimuth. As we cannot expect a calibrated magnetometer, the resulting azimuth is shifted with an unknown offset error. We combine these positional sensors with ALS readings to generate a directed lighting map. An example of such a transformed measurement is shown in Fig. 7.3. The x-axis represents the (shifted) azimuth as derived from magnetometer readings while the y-axis shows altitude calculated from accelerometers. The color corresponds to recorded luminance. Fig. 7.3 therefore shows the path where the mobile device faced the sky with corresponding color-encoded brightness.

While this gives us luminance values of the user's surroundings, the sun's position is not directly obvious. We need to fill the areas not passed by the path as the sun's center might be there. We therefore interpolate a global luminance model using double-powered inverse distance weighting by applying a convolution matrix $(k_{x,y})$ of order N with

$$k_{x,y} = \begin{cases} \left(1 - \left(\frac{\sqrt{x^2+y^2}}{\frac{N}{2}}\right)^2\right)^2 & \text{if } \sqrt{x^2+y^2} < \frac{N}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

We chose this matrix as it has the following properties: *a)* elements in the center of the matrix $\left(\frac{N}{2}, \frac{N}{2}\right)$ are weighted with 1, *b)* as the distance to the center increases, elements are weighted from 1 to 0 with an eased function and *c)* the corners of the matrix outside of a radius of $\frac{N}{2}$ from the center are weighted with 0.

We then apply this convolution matrix to our directed lighting map (Fig. 7.3). For

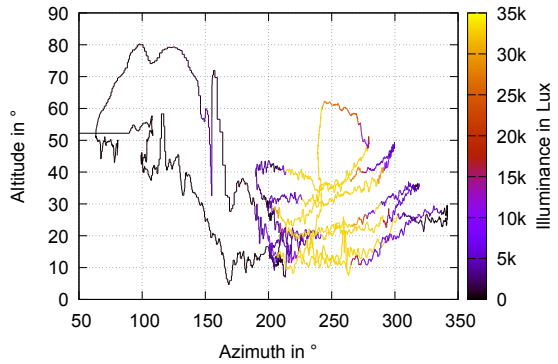


Figure 7.3: Sensor values transformed to horizontal coordinate system with color encoded illuminance.

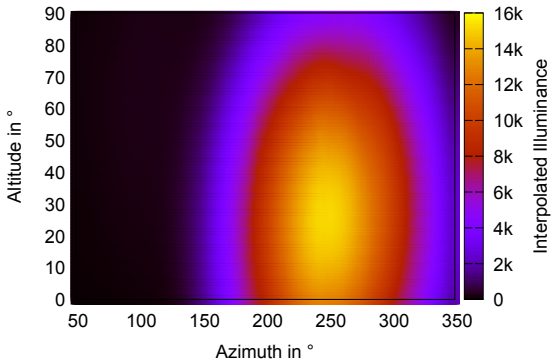


Figure 7.4: Interpolated luminance.

every altitude/azimuth pair, the matrix $(k_{x,y})$ is centered on that pair and the sum of its weighted neighborhood is the value of that coordinate pair in a new transformed lighting map. Fig. 7.4 shows an example of this convolution applied to Fig. 7.3. The color in that figure corresponds to interpolated luminance in that direction.

Finally, we estimate the sun's altitude and azimuth by finding the global maximum. We refer to a pair of estimated altitude and azimuth at point in time t_i as an *observation* denoted by $\text{Obs}(t_i)_{\text{alt}}$ and $\text{Obs}(t_i)_{\text{azi}}$.

7.4.3 Location Candidates

One observation defines the sun's position non-ambiguously in the horizontal coordinate system and therefore the user's location. However, preliminary tests showed an error of up to $\pm 40°$ in azimuth due to inaccurate magnetometer readings. This is in line with prior research which reported a compass error of $10 - 30°$ on mobile devices [14]. We therefore keep the altitude that we derived from the accelerometer, and use observed azimuth only as a cue. Actual azimuth is determined with a second measurement. Considering only the altitude of the sun, one observation defines a set of locations where such an apparent altitude can be observed at a given time. Geometrically this set forms a circle on the Earth ellipsoid.

Determining perceived altitude and azimuth of a celestial object from a local observer at a certain point of time is a standard task of astronomy programming libraries but requires knowledge of the observer's position which is unknown in our approach. However, we do know the sun's horizontal position and can therefore numerically approximate possible observer locations. The center of that circle of possible observer

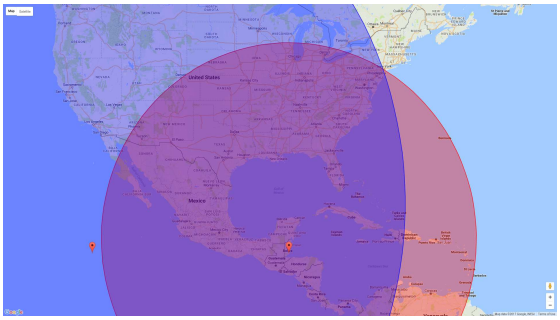


Figure 7.5: Circles defined by two altitude measurements in New York on 2017-08-06 at 14:00/16:00. Simulated.

locations is the *subsolar point*, where the sun is at 90° altitude.

Given two observations at different points in time, the observed altitude differs and two distinct circles are defined (Fig. 7.5). Assuming the user is still near their location from the first measurement, we perform a circle-circle intersection to reduce the number of location candidates to two[1]. If this assumption is not met the user's position difference adds a linear error to our approach. Compared with the empirical accuracy of our approach (cf. Sec. 7.6) this error is negligible in most cases.

Fig. 7.6 shows the approach for two observations $\text{Obs}(t_1)$ and $\text{Obs}(t_2)$. Intersecting them yields I_1 and I_2 as intersection points and therefore two location candidates. To select the correct location out of these two, we use the magnetometer azimuth readings $\text{Obs}(t_1)_{\text{azi}}$ and $\text{Obs}(t_2)_{\text{azi}}$. Although we argue that the magnetometer is too inaccurate to represent the azimuth, it still points in the general direction and suffices to make the correct selection out of opposing candidates. We therefore calculate the expected azimuth $azi(t_i, I_j)$ of both intersection points for both points in time using an astronomy library. The expected azimuth values are then compared to the ones yielded by the magnetometer readings and the intersection with least divergence is chosen. In Fig. 7.6, the smaller azimuth deviation at I_1 indicates that this is the correct candidate, because the measured azimuth is closer to the expected azimuth than at I_2. We call the selected candidate the *intersection result*.

[1]Mathematically, also zero, one or an infinite number of intersection points are possible. This could happen due to an erroneous measurement or the user moving several hundreds kilometers. In these theoretical cases—which never occurred during our evaluation—we would discard the measurements and perform new ones.

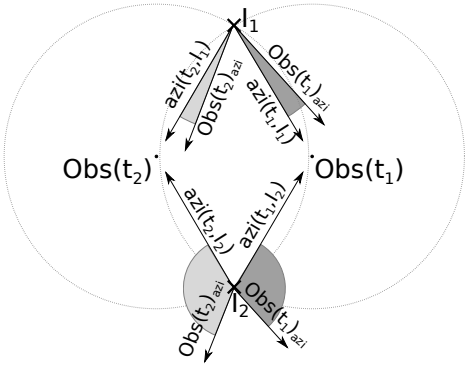


Figure 7.6: Intersection of location circles.

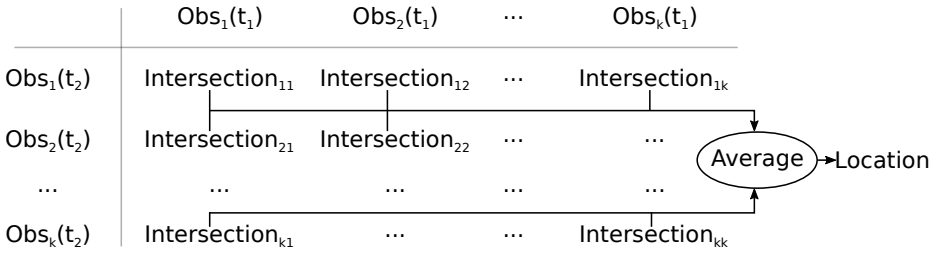


Figure 7.7: Increasing accuracy by using redundant observations.

7.4.4 Location Aggregation

In the previous section we showed how we compensate azimuth inaccuracies and calculate device locations. If redundant measurements are available, we can utilize them to mitigate altitude measurement errors and thus improve the accuracy of geolocation. Redundant measurements occur when the user keeps using their mobile device even after we have computed one observation Obs(t_i).

Fig. 7.7 shows how this redundancy integrates in the whole approach. Instead of two measurements, we perform k measurements for both points in time, which allows us to perform k^2 circle intersections and thus yields k^2 intersection results. We then select the median latitude and the median longitude of all intersections as the final result. We refer to the whole process as one *location determination*.

7.5 Implementation

We use a web-based prototype that collects sensor data with the Ambient Light Sensor API [68] and DeviceOrientation API [123]. The implementation also records the device location as reported by the GPS-based Geolocation API [95] for evaluation purposes

to determine the accuracy of the digital sextant. The processing of the collected data is implemented in Python and uses PyAstronomy [30] as astronomy library.

Dealing with interferences. Reflections may interfere with the result subject to the reflecting surface. In one measurement series we found a light source at an altitude $< 0°$, which was due to a reflecting window board. While this is trivial to detect and filter, reflections from perpendicular surfaces (for example windows) are more challenging. In this case altitude is not altered, but azimuth will be misestimated and can lead to an inaccurate location if the angle is large enough. As a plausibility check we have implemented a duplicate peak detection: if more than one significant light source above a threshold (the sun and its reflection) is found while searching the global maximum (cf. Section 7.4.2), the measurement will be marked as indecisive and rejected.

The sun threshold may vary between mobile devices due to different sensors used. If the device model is known (e.g., based on the user agent), the threshold can be preset subject to the sensor datasheet or to empirical data. If the model is unknown, the threshold can be set to 95% after collecting sensor data sufficiently long, assuming that the sun has been recorded eventually.

As our method requires exposure to direct sunlight, being indoor or under a clouded sky prevents geolocation. However, these cases can be identified trivially as the sun is orders of magnitude brighter on the Lux scale than other light sources. By prefiltering measurements for sunlight exposure we can avoid mislocating the user in theses cases.

7.6 Evaluation

The evaluation addresses the following research questions:

- Is the method applicable in everyday smartphone usage scenarios? (Section 7.6.1)
- What impact does the device have on the result? (Section 7.6.2)
- What is the impact of the waiting time between two observations? (Section 7.6.3)
- What is the impact of the user's location on Earth on the accuracy? (Section 7.6.4)
- What is the impact of the time of day and time of year on the accuracy? (Section 7.6.5)

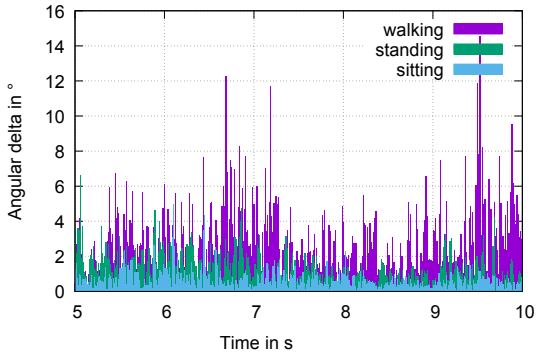


Figure 7.8: Impact of user behavior on accelerometer readings.

- Do redundant observations help to improve the result? (Section 7.6.6)

We evaluate our method with a Google Nexus 7 (2013) tablet and a Samsung Galaxy S7 (2016) smartphone running Firefox for Android 48.0.

7.6.1 Practical Applicability

An essential requirement of our method is to face the sun, which leads to the question, whether this is realistic during everyday smartphone activities. To provide an indication for the applicability we performed the following tests with a Samsung Galaxy S7:

1. Stand with the sun in the back (*standing*)
2. Walk 25 m towards the sun, turn around and walk 25 m away from the sun (*walking*)
3. Sit with the sun in the back (*sitting*).

In each setting the test person looked at the smartphone while touching, scrolling and reading the screen, which causes the tilting phone to eventually face the sun. Measurements were 40 seconds long and took place on an unclouded day at 11:00, 13:00 and 15:00 in Duisburg, Germany. Each altitude/azimuth observation has been repeated $k = 5$ times, which amounts to a total of 45 measurements.

Movement profiles. Fig. 7.8 shows the angular difference between consecutive accelerometer vectors (data points) for an excerpt of each test. We see that during walking the smartphone moves more than during standing, and during standing more than during sitting, leading to different *movement profiles*. The sensor readings are

Table 7.1: Average angular velocity ω of various data sets.

	Table	Hand	Sit	Stand	Walk
ω in $\frac{^\circ}{s}$	10.41	27.44	115.70	157.04	229.80

Table 7.2: Accuracy in each test.

	t_2	Error (km)	Spread (median/km)
Walk	13:00	1196.53	636.79
	15:00	1083.73	424.73
Stand	13:00	808.98	960.48
	15:00	777.08	231.13
Sit	13:00	380.24	300.79
	15:00	146.43	131.44

not only influenced by user movement, but also by sensor jitter. This is demonstrated with two other tests: 1) the test person holds the smartphone in their hand without deliberate movement (*hand*), 2) the smartphone rests flat on a table (*table*). Table 7.1 shows the average angular velocities of each test: the phone resting on the table without any observable movement measures an angular velocity of $\omega = 10.41\frac{^\circ}{s}$, showing that sensor jitter has indeed an influence.

Location Determination. We apply our approach to the 11:00 observations of each test, paired with their corresponding observation at t_2 two or four hours later. Table 7.2 shows the location accuracy, i.e., the error of the determined location compared with the actual location. We can see a clear influence of movement profiles on accuracy: as (unconscious) motions of the user are reduced, the distance between determined and actual location reduces.

We conclude that while motion influences our approach, there are plausible scenarios that achieve an accuracy usable for country-level geolocation, in spite of sensor noise. A remaining open question is how often these scenarios occur with everyday smartphone usage of unaware users. This depends on the users' habits and we leave it for future work to collect sensor data of multiple persons during everyday activities

Table 7.3: Accuracy comparison of Nexus 7 and Galaxy S7.

Nexus 7		Galaxy S7	
Error (km)	Spread (median/km)	Error (km)	Spread (median/km)
154.1	592.9	388.2	223.0
174.7	426.1	393.9	274.8
326.5	208.1	469.9	213.9
360.9	243.6	487.0	134.9
456.2	239.5	688.3	177.4
463.4	258.5	753.3	250.3
464.3	329.0	763.7	201.7
527.7	390.6	976.5	169.1
1,052.4	353.5	1,023.9	339.7
1,993.8	487.3	1,575.0	543.8

to quantify the occasions for our geolocation method.

Based on the observation that too much macro human movement deteriorates the accuracy, we can restrict the method to run in situations with low-movement profiles only. This can be achieved by analyzing angular velocity or by using an existing method [71] to determine the user's current activity.

7.6.2 Device Comparison

Now that we have an indication for the practical applicability of our approach, we systematically analyze various factors that influence the geolocation accuracy. In the following measurements, we hold the mobile device in one hand and tilt it in two dimensions while pointing at the sun (cf. Fig. 7.2).

We performed 10 location determinations with a Google Nexus 7 (2013) and a Samsung Galaxy S7 in Duisburg, Germany. First measurements were conducted at noon, the second ones two hours later at 14:00. Each altitude/azimuth observation has been repeated $k = 5$ times yielding a total of 100 measurements per device.

Our results are shown in Table 7.3. Column "Error" shows the error of 10 location determinations between our method and the true location as recorded by the Geolocation API. Accuracy ranges from 154.1 km to 1993.8 km with a median error of 459.8 km for the Nexus 7. For the S7, the accurancy ranges from 388.2 km to 1575.0 km. While the error range is smaller, the median error is 720.8 km and thus 56% higher

than for the Nexus 7.

Because each location determination consists of $k^2 = 25$ redundant intersection results, we can examine the spread of these intermediate results as well. Column "Spread" shows the median error to the correct position of all 25 intersection results for each location determination. Interestingly, a larger spread within each location determination does not negatively impact the final accuracy. For example, the best result of the Nexus 7 in the first row has the highest spread of all location determinations for this device. Systematically, accuracy and spread correlate weakly with a Pearson correlation coefficient of $r = 0.2367$ for the Nexus 7. For the S7, this value is significantly larger with $r = 0.7553$. We will discuss differences of these devices and possible explanations in Section 7.6.2.2. Since there is no general high correlation between accuracy and spread, we conclude that our approach is robust and is not easily influenced by random measurement errors. In other words, each redundant measurement contributes to the accuracy and does not distort the final result.

7.6.2.1 Effect of Altitude Error

We now analyze the impact of an altitude estimation error in order to determine the expected results with more (or less) accurate sensors.

Analysis. Altitude alone — without considering non-linear influences of circle intersection — yields a location error of $\frac{\delta}{360°} E_{circ}$ for an assumed altitude error of δ. This can be derived from the Earth cross section diagram in Fig. 7.9. Parallel sun rays reach Earth and are correctly observed at P_1 with an altitude of α_1. Angular distance to the subsolar point Z (i.e., angular radius of the circle) amounts to $\gamma_i = 90° - \alpha_i$ by corresponding angles. In case there is an altitude estimation error and $\alpha_2 = \alpha_1 + \delta$ is observed this will yield an erroneous location P_2. Solving these equations yields an angular distance between P_1 and P_2 of $\delta = \gamma_1 - \gamma_2$ which corresponds to a location offset by $\frac{\delta}{360°} E_{circ}$.

Simulation. To simulate the non-linear parts above this lower error bound we created two groups of artificial observations with a time difference of two hours. Each simulation initially assumes perfect altitude and azimuth observations as computed by PyAstronomy for this time and place. For each group we then generate simulated observations with altitude deviations of $\pm\delta$ and a probing width of 0.25°. Since we are considering the worst case impact, we perform our location determination approach on each pair out of both groups and then use the maximum location error as result.

The results are presented in Fig. 7.10 and Fig. 7.11. The worst case location error

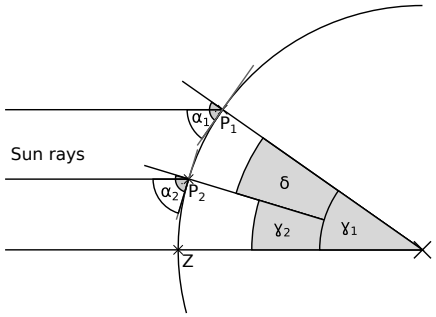


Figure 7.9: Location error depending on altitude.

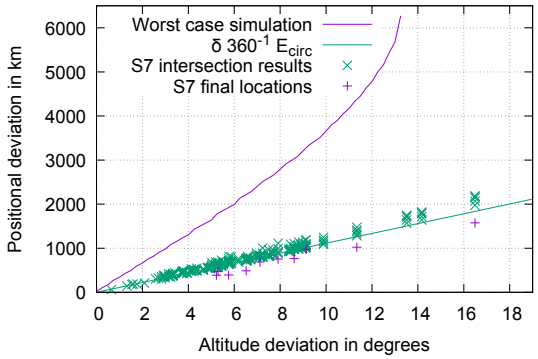


Figure 7.10: Simulated worst case observations compared to S7 based measurements.

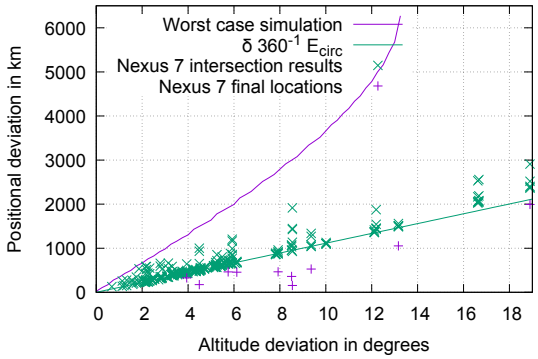


Figure 7.11: Simulated worst case observations compared to Nexus 7 based measurements.

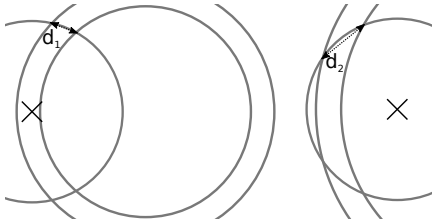


Figure 7.12: Effect of circles intersection angles on distance, $d_1 < d_2$.

is significantly larger than the linear lower bound estimation. We can also see the worst case gradient grows as altitude deviation increases, which means the maximum error is non-linear. This is due to one circle growing so large that it almost covers the other shrunken circle completely. The effect is illustrated in Fig. 7.12. The small intersection angle cause any additional radius difference/altitude deviation to yield an even higher location error. Measurements with altitude deviations greater than 13.5° are not guaranteed to yield intersecting circles causing the result of the function to be undefined thereafter.

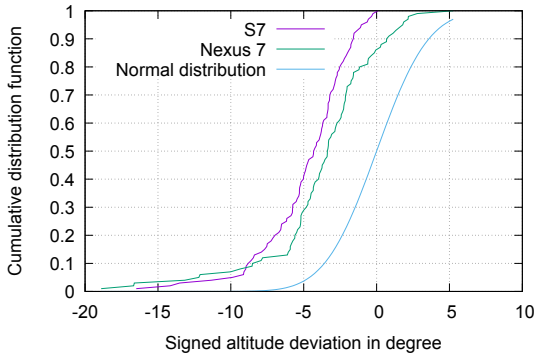


Figure 7.13: Distribution of signed altitude deviation.

Practical Evaluation. While the error could be very high in theory, we now examine whether this has happened during our practical evaluation. Since we know the correct altitudes in our experimental setup, we can calculate altitude deviations for the intersection results (i.e., intersections before calculating a median and yielding a final position) and final positions in these figures. For each data point we aggregate the altitudes of all involved observations as a maximum to get a worst case view. In terms of location error the measurements are close to the linear bound. This indicates that the empirical average case is close to the linear bound and that the theoretical worst case does not occur in practice. Interestingly, the final positions are below the linear threshold, suggesting that measurement errors cancel each other out and thus redundant measurements produce a more accurate location result.

7.6.2.2 Systematic or Random Error

We now investigate why the results of the two mobile devices scatter to a different extent and whether the sensor error is random or systematic.

A difference between Fig. 7.10 and Fig. 7.11 lies in the vertical spread of intersections. While they stick to the lower limit in the former they spread more towards the upper bound in the later. This means the same absolute altitude deviation has a different impact on both devices. To analyze this anomaly we took the sign of the altitude deviation into consideration.

Fig. 7.13 shows a cumulative error distribution function of the signed altitude error for both devices. If deviations were merely due to random errors we would assume a uniform distribution around 0. Instead there is strong bias towards negative deviations, i.e., measuring the sun at a lower position as expected. The effect is even

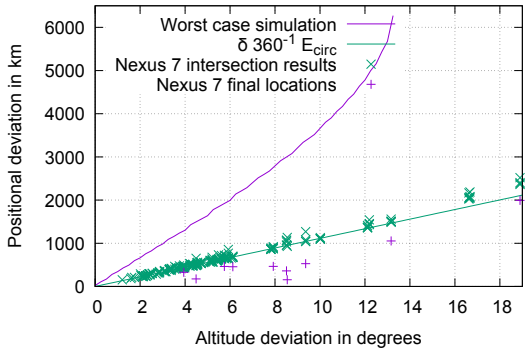


Figure 7.14: Simulated impact of altitude error on Nexus 7 without negative altitude deviated measurements.

stronger on the Galaxy S7, where we observed almost no positive deviations, suggesting a device-specific systematic error. This distinction also provides an explanation for increased vertical spread: A positive deviation intersected with a negative deviation yields a higher location error than an intersection between two equally signed deviations. To verify this finding we removed intersections with a positive deviation on Nexus 7 and plotted the result in Fig. 7.14. The vertical variance has been drastically reduced, which confirms a device-specific systematic error.

This device-specific difference in altitude deviation also provides a viable explanation for the different correlation coefficient outcomes from Section 7.6.2. Since location determinations performed with the Nexus 7 contain more outliers in the intersection results (see Fig. 7.11), the results will have a higher spread than with the Galaxy S7. We calculate a median out of these intersections to obtain the final position. Since medians are in general robust against influences by outliers, there is no strong correlation between the spread of a location determination and its positional deviation on the Nexus 7.

This has implications for our redundancy parameter k. If there is a systematic bias like for the Galaxy S7, then additional measurements will not increase the accuracy of our approach. On the other hand a random error will be compensated by an increase of k and successive averaging. Telling these errors apart outside of an experimental setup where no correct altitude is known is non-trivial. If a systematic error depended solely on the device model, an attacker could classify various models and act accordingly. However, analyzing this dependency is outside the scope of this chapter.

Table 7.4: Frequency of sensor readings.

	Nexus 7		Galaxy S7	
	$\bar{x}$	σ	$\bar{x}$	σ
Ambient light readings / Hz	3.67	1.30	5.61	0.04
Acceleration readings / Hz	192.60	2.37	99.09	0.41

7.6.2.3 Sensor Sampling Rate

Another difference between both mobile devices is how often they provide new sensor values. We now analyze whether this has an effect on our method.

Table 7.4 shows the sensor frequency per device across all measurements from Section 7.6.2. Concerning steadiness, the S7 performs better due to its significantly lower standard deviation for both the ambient light and acceleration sensor. This could be due to the faster processor and the general technological advancements during the 3 years between the release of both devices. Since our JavaScript implementation runs inside a web browser on top of a non-real-time operating system, there are several components involved being possible causes of this difference. The frequency of the Nexus 7 light values scatters widely with a standard deviation of 1.30 from its average of 3.67 Hz. However, this does not seem to affect our method as the Nexus 7 achieves a better accuracy (cf. Tab 7.3). The only sensor values more in favor of the Nexus 7 are acceleration events per second, which are nearly twice as frequent on the Nexus 7 than on the S7. Although this correlates with a higher location accuracy, it does not provide a plausible explanation. Our approach uses both accelerometer *and* ambient light readings to determine the altitude of the sun. Even if the accelerometer frequency is doubled and if the device orientation is more accurate, the luminance in that direction still lacks behind—especially when the ambient light sensor has a low sampling frequency.

7.6.3 Time Between Measurements

We now evaluate whether the length of the time span between two measurements has an influence on the location result. We performed hourly measurements from 10:00 to 18:00 with the Nexus 7 in Duisburg, Germany. Each of these 9 time slots was measured on 5 different sunny days. As before we performed each altitude observation

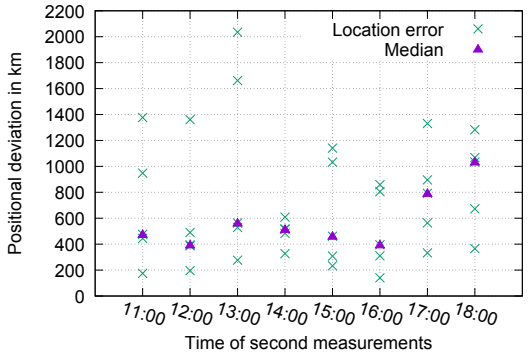


Figure 7.15: Impact of time difference between measurements. First measurement at 10:00.

$k = 5$ times, yielding a total of 225 measurements.

Fig. 7.15 shows the location determination between the 10:00 measurements and every other time slot from the same day. Similar to the measurement from Section 7.6.2, the median location error is between 400 and 500 km. While there is a high dispersion within each time slot, there is no definite trend through out the day up to 16:00. After that, the median error of the measurements at 17:00 and 18:00 is twice as large as before.

As these results fit into the spreading range of previous measurements at 11:00, 12:00 and 13:00 this could be due to a measurement error or have a systematic cause. Either way, we conclude that a time frame of 1 to 6 hours between two measurements is suitable to determine the circle intersection. Within this time span there will be no detrimental effect on the location accuracy.

Location Changes. There is no requirement to carry out the two measurements at the exact same location. If the user has moved in the meantime between two measurements, the positional error caused will be proportional to the distance moved. The effect corresponds to an altitude error analyzed in Section 7.6.2.1. For example, if the second measurement takes place 111 km away from the first, this may introduce an altitude error of up to 1°. In worst case, this corresponds to an error of 300 km (cf. Fig. 7.10). However, subject to the direction of the user movement, the altitude may be distorted less or not at all. Plus, the introduced positional error does not necessarily add up but may cancel out an existing measurement error, which means the 300 km worst case is unlikely in practice. We can conclude, the error introduced by a location change of, e.g., < 30 km will be negligible in comparison to other error effects.

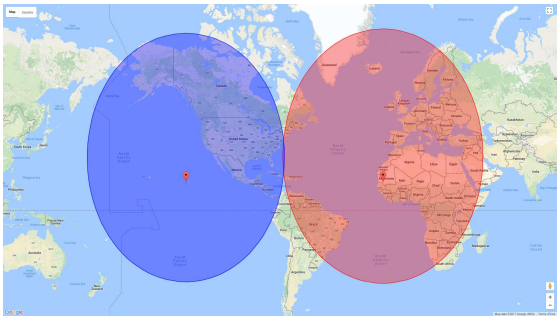


Figure 7.16: Circles defined by altitude measurements in New York on 2017-08-06 with 8 hours interval (simulated).

7.6.4 Location Impact

So far, the evaluation took place at a fixed location. We now analyze by simulation whether the device location on Earth has an impact on the accuracy of our method.

Assuming an altitude error of $\delta \leq 1°$, we simulate the worst case location error on a fixed longitude $-73.996°$ with latitude ranging from $-70°$ to $70°$. Our results in Fig. 7.17 show a definite symmetric similarity around latitude $= 17°$. Minimal worst case deviations of 163 km are achieved at latitudes $2°$ and $33°$ with the error increasing in both directions. The reason for a growing error are the angles of the intersecting circles. At latitudes $2°$ and $33°$, the circles intersect at nearly $90°$ minimizing non-linear effects discussed in Section 7.6.2.1. If we move further away from the axis of symmetry, the radiuses of circles and therefore intersection areas and angles rise, thus increasing the error almost linearly.

On the other hand, if we move closely to the axis of symmetry, the effect shown in Fig. 7.16 occurs: the intersection area decreases and intersection angles become more acute, thus increasing the error superlinearly. Between a latitude of $12°$ and $22°$ there is even no guaranteed intersection within the defined altitude error margin, which means that our method does not yield a location. This is understandable if we reason about the cause of this axis of symmetry. On the day of our simulation the subsolar point (i.e., where the sun has an altitude of $90°$ throughout the day) intersects longitude $-73.996°$ at latitude $16.435°$. The centers of both circles lie on this subsolar path which causes the intersection area to become smaller the closer the observer is to that path. As soon as the minimal central diameter is smaller than $\frac{2°}{360°} E_{circ}$ the intersection area disappears at an altitude error of $\leq 1°$.

Longitude has no fundamental impact on this finding. Besides a shift of the axis

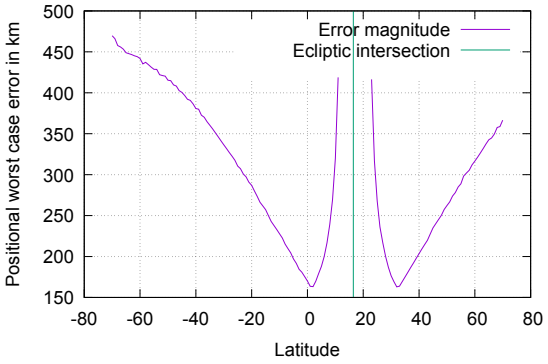


Figure 7.17: Simulated impact of latitude on accuracy (2017-08-06, altitude deviation $\leq 1°$, longitude $= -73.996°$, jittering due to numerical approach).

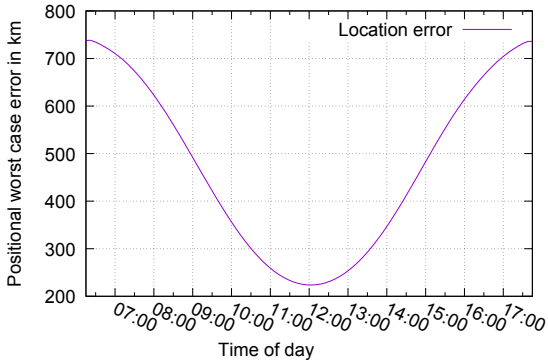


Figure 7.18: Simulated effect of daytime on accuracy (2017-08-06, altitude deviation $\leq 1°$, New York).

of symmetry caused by axial tilt, a similar figure is obtainable for any other longitude with an adjusted time.

This means, a measurement error has a larger detrimental impact on the location result near the equator than in other parts of Earth. In worst case, our approach fails to yield a location. However, assuming typical measurement errors, we can at least deduce from the sun's altitude that the user is near the equator between two non-intersecting circles.

7.6.5 Time Impact

Similar to the location on Earth, the time of day and time of year could influence the accuracy as well. We determine the impact by simulation, assuming an altitude error of $\delta \leq 1°$ and plotting the worst case location error.

Time of Day. Fig. 7.18 shows the simulated effect of the time of day. The first measurement takes places at the time shown on the x-axis and the second measurement takes place two hours later. Again we see the non-linear effect of sun altitude on accuracy: when the sun reaches solar noon, a minimum worst case error of 206 km is reached. Before and after this the error increases symmetrically. This is due to more acute intersection angles described in Section 7.6.4.

Time of Year. The sun altitude is not only affected throughout the day but also throughout the year. Fig. 7.19 shows the simulated worst case location error of two measurements at 14:00 and 16:00 with any altitude error $\delta \leq 1°$ for each day in 2017. In general, accuracy is better in summer than in winter due to the higher sun altitude

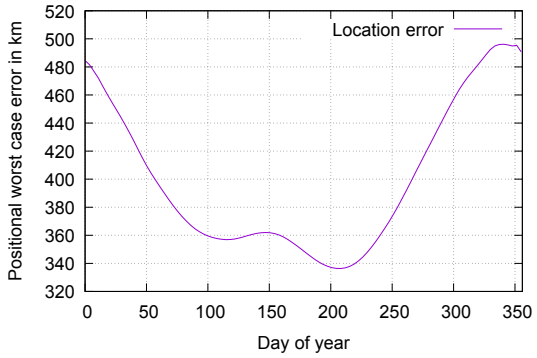


Figure 7.19: Effect of day of year on accuracy (2017, altitude deviation $\leq 1°$, New York).

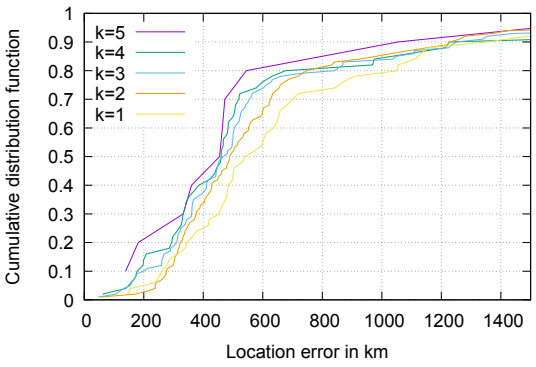


Figure 7.20: Effect of redundant observations on Nexus 7.

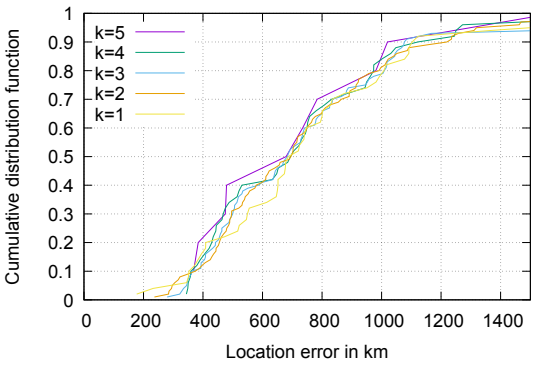


Figure 7.21: Effect of redundant observations on Galaxy S7.

angle. However, the impact of seasons is much smaller than the time of day. The bump in the graph is caused by a decreasing accuracy towards the subsolar point, which is the same effect as discussed in Section 7.6.4: intersecting circles become smaller causing more acute angles. The size of the bump is subject to the latitude.

7.6.6 Effect of Redundant Observations

Our approach aggregates redundant observations in Section 7.4.4 to increase accuracy. This section evaluates the effectiveness of this step.

We use the Nexus 7 and Galaxy S7 measurements from Section 7.6.2, which were recorded with $k = 5$ and simulate decreased redundancies $k' = 4$ down to $k' = 1$. To avoid a selection bias we simulate all $\binom{k}{k'}$ combinations for every k'. Our results are shown as cumulative distribution functions in Fig. 7.20 and Fig. 7.21.

The figures differ in their distributions due to device differences discussed in Sec-

tion 7.6.2.2. These device differences also affect to what degree accuracy increases with an increase of k'. As the S7 has a larger systematic error than the Nexus 7, its results improve less from redundant observations than the Nexus 7. For example, 50% of the Nexus 7 location errors are below 540 km for $k' = 1$ and below 454 km for $k' = 5$, which is an increase of accuracy of 15.9%. On the other hand, the location error of the S7 improves just by 2.3% for the 50% mark between $k' = 1$ and $k' = 5$. Still both devices have in common that $k' = 5$ yields the best results in the vast majority of cases.

As this increase of accuracy is even more the case on the Nexus 7 we conclude that our approach of redundant observations is suitable to mitigate random errors during measurements.

7.7 Countermeasures

As the geolocation method bears the risk of violating the user's desire for privacy, we now investigate potential countermeasures. An obvious remedy is to disable sensor access completely for apps and for websites. For example, the Orfox Browser [122], a mobile Tor Browser, restricts the use of any sensors and thus limits the possibilities to leak information to websites. While this prevents a whole class of sensor-based attacks, it also limits the potential of the web as an application platform.

Another remedy is to artificially reduce the sensor resolution to provide a compromise between privacy concerns and legitimate use cases. We analyze this possibility by truncating the sensor data of the Nexus 7 from Section 7.6.2, both regarding ALS and accelerometer. The truncation consists of rounding sensor readings to next multiples of a variable sensor truncation factor ϵ. This simulates a sensor with a reduced resolution.

7.7.1 Ambient Light Sensor

Reducing the ALS resolution has been shown to prevent information leakage in other use cases [105]. Interestingly, our approach did not yield significantly worse results while iteratively increasing the truncation factor ϵ.

In extreme case, we round the ambient light sensor readings to binary values. Table 7.5 shows the results of such a binary truncation. Compared with regular results from Table 7.3 the results have shifted: while the average location error increased

Table 7.5: Location accuracy with ALS truncated to binary scale.

Nexus 7	
Distance (km)	Spread (median/km)
104.7	378.8
230.6	375.7
318.7	176.0
361.2	269.3
382.0	235.6
449.8	366.1
460.8	347.4
487.1	370.0
979.4	628.7
2,736.2	1,112.5

slightly by 9%, some results have become more accurate.

Although this result appears surprising, it is reasonable due to how our approach works. The sun is significantly brighter than everything else recorded like diffuse reflections or artificial lighting. This will cause all values to become 0 except those measurements directly pointed towards the sun. Our inverse distance weighting will then yield the center of these points as the altitude. As potential interferences do not pass the binarization filter, some results become more accurate while others become worse due to a loss of information.

From these results we conclude that reducing ALS resolution definitely does not provide an obstacle for our approach.

7.7.2 Accelerometer

Similarly, we simulate the impact of truncating the 3D accelerometer on location accuracy with truncation factors of $\epsilon \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ $\frac{m}{s^2}$. To put these values into perspective with Earth's gravitational acceleration of $9.81 \frac{m}{s^2}$, the coarsest sensor resolution in our simulation should only be able to differentiate rotations multiple of 90°.

Fig. 7.22 visualizes the simulation output. While up to $\epsilon = 4$ the median location error does not become significantly worse, the number of successful location determinations drops as ϵ increases. This is due to observational circles not intersecting when

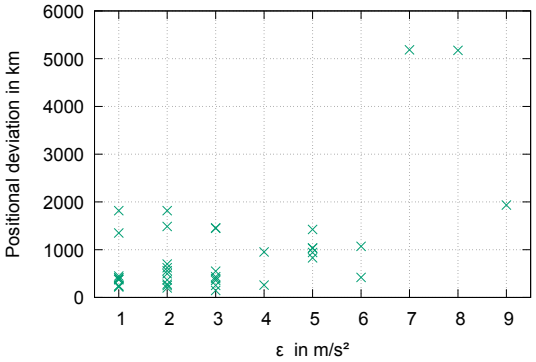


Figure 7.22: Impact of truncated accelerometer resolution.

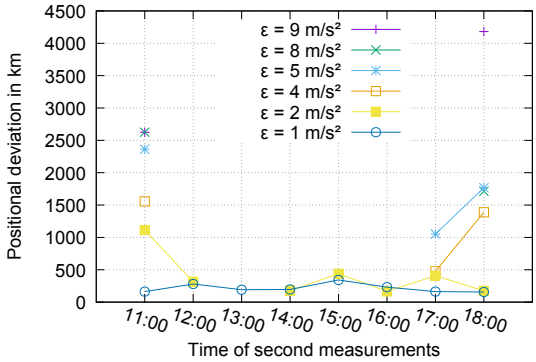


Figure 7.23: Impact of truncated accelerometer during the day.

the altitude error is too large or due to contradicting global maxima caused by the threshold function.

Time of Day. The above data bears a systematic bias as it provides little variance in altitude: all measurements took place at 12:00 or 14:00. However, the effect of truncating the accelerometer depends on the altitude of the sun. To investigate other possible outcomes, we re-run this simulation on the best measurements per time of day from Section 7.6.3.

The results are shown in Fig. 7.23. While there is no significant impact for $\epsilon \leq 2$ except for one outlier at 11:00, using $\epsilon \geq 4$ causes our approach to fail when the second measurement takes place between 12:00 to 16:00. This coincides with our previous findings from Fig. 7.22, where the effectiveness of our approach degrades with a higher truncation factor.

The absence of successful location determinations between 12:00 and 16:00 is explainable by considering the course of the sun. An altitude of 0° at the beginning and end of the day can be represented correctly even by a truncated sensor.

This shows that a truncation of the accelerometer impairs the geolocation approach, but may still leak coarse information with a large error to an attacker.

7.8 Related work

The user's location is generally regarded as a privacy-sensitive information with a large body of research dedicated to ways of utilizing various available sources to infer it.

Locating a device by looking up its IP address in a database is a heuristic im-

plemented by various commercial products. The observation behind this approach is that Internet service providers distribute IP addresses in geographical proximity. While this approach is straightforward once a database has been created, it fails when a privacy-aware user employs techniques to obfuscate their IP address (e.g., VPN, proxy server, Tor).

Michalevsky et al. [83] propose to repeatedly read the phone's power meter, which is available without special permissions in the Android API. Cellular radio power consumption depends on location due to obstructions and cellular tower placement allowing trajectory reconstructions using a dedicated coverage database. While providing more accurate results than our approach, it bears additional requirements: both an app has to be installed on the device and the user has to move in a coverage charted area.

Using camera images is another way of locating a user. Guan et al. [52] show that it is feasible to determine positions in a city by reading camera images and inertial sensors. Ma et al. [78] take the sun into consideration to navigate and locate the user on a map. Both approaches show remarkable accuracies but require camera permissions. A privacy-aware user is unlikely to grant this to a dodgy website or app.

In some scenarios inertial sensors alone provide enough information to locate users, if users move along given paths. Hua et al. [59] showed metro lines have distinctive accelerations patterns allowing to track users. ACComplice [55] uses solely the accelerometer to identify car trajectories in trained data. Narain et al. [86] additionally use magnetometer and gyroscope readings and present an approach to locate a car using publicly available cartographic data without any training. While this increases practical applicability, feasibility on a global scale remains open. Compared to our approach this also requires cartographic material which might not be available for the subject's location.

Wi-Fi BSSID-based approaches are a standard way of locating smart devices. Zhou et al. [134] showed that an Android application without location permission—and therefore not able to utilize the operating system's BSSID queries—can still read the BSSID of the connected access point and perform its own BSSID lookup on application level. Another side channel discovered by Zhou et al. consists of the speaker API. Every application can query whether any other application is currently playing sounds. Originally designed to provide apps with means of coordination, this allows to measure playback duration and deduce announcements made by a GPS navigation app.

Ambient light is considered in [4] and [76] for location determination. SurroundSense [4] looks at a combination of ambient sound, light and color to acquire fine-grained location fingerprints to distinguish shops in a mall. Epsilon [76] uses visible light beacons that are broadcasting their position. High frequency pulse width modulations on LED bulbs makes this flickering indistinguishable from dimming to the human eye. While both approaches promote non-malicious use cases, especially Epsilon is suited to violate the user's privacy since it could be implemented using a zero-permission app or website. However, it would require to deploy an infrastructure of beacons to track users.

7.9 Conclusions

In this chapter we presented a novel approach to locate mobile devices using sun-based measurements. The approach utilizes mobile device sensors that are accessible on platforms like Android without asking the user for permission. Unlike related work, a prior training or cartography of the user's area is not necessary, as we are relying on the well-known movement of celestial bodies.

In our experimental evaluation we achieved a median accuracy of better than 500 km, which is sufficient for country-level geolocation. The location accuracy will improve with more accurate sensors in mobile devices. Our analysis has shown that both random and systematic sensor errors influence the result, where the random error portion can be minimized by utilizing redundant measurements.

For future work we would like to improve our approach to cope with indirect sunlight. So far we rely on direct exposure to calculate altitude and azimuth. With an advanced sky model and sensor calibration it could be possible to estimate altitude based on a path not intersecting or tangent to the sun.

In line with previous work in this field, our zero-permission geolocation approach once again shows that it is unforeseeable what high-level information might be concluded from seemingly harmless sensor values. One way to cope with this threat in general is to truncate sensor readings by default, which helps to preserve privacy in several cases while still providing a value to legitimate applications. This shift of mindset is for example adapted by the web community: the August 2017 editor's draft of the W3C Ambient Light Sensor specification [66] urges browser vendors to consider mitigation strategies like reduced sensor sampling frequency and accuracy.

In our case, truncating the ambient light sensor has almost no effect on location

accuracy and thus does not help. Truncation of the accelerometer worsens the accuracy and number of location determinations when rounded to multiples of 4 or more. Such an impairment likely affects legitimate use cases, thus questioning the adequacy of such a tradeoff.

A mitigation strategy might be to ask the user for permission before allowing sensor access at all—not for individual sensors as this affects the usability, but for a group of sensors that are less privacy-invading than camera or microphone access while still revealing some contextual information about the user, including accelerometer, barometer, magnetometer and ambient light sensor. An important element would be to disclose to the user what information is being collected and what it is used for. A technical enforcement combined with a documented privacy policy allows users to make an informed choice whether they approve the disclosure of contextual information.

Chapter 8

Conclusion and Outlook

In this thesis, we discussed classical and novel privacy threats in the mobile web based on selected examples.

Architectures

We have shown CAs to be susceptible to network-level attackers. While some CAs responded positively to our disclosure and vulnerabilities were closed, the main problem persists. Due to the trust model of the web PKI, an attacker can choose the weakest CA for an attack. Even if the domain owner employs signed `CAA` records to prevent switching to such a CA, there is no mitigation provided if the attacked CA does not use DNSSEC. Requiring all CAs to implement DNSSEC by codifying this in the Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates would be an important step to mitigate such attacks. In October 2018 the *Validation Subcommittee* of the Server Certificate Working Group has been established by the CA/Browser Forum[1]. While this indicates some awareness regarding the need for improvement of validation practices there has been no public proposal which would address the vulnerabilities found in this work.

We have presented architectural approaches which promise privacy-improved online social networks but come with their own trade-offs: hosting own infrastructure, being online while one's friends are, entrusting unknown entities with data or having worse performance than large commercial OSNs. Moreover, the challenge of hiding meta data is still an active field of research. The degree of privacy preservation varies

[1] `https://cabforum.org/2018/10/03/ballot-sc-9-establish-the-validation-subcommittee-of-the-scwg/`, Accessed 2019-03-02

between approaches as different privacy goals are formulated with varying attacker models. A significant challenge lies in traffic correlation attacks, i.e. an attacker concluding that outgoing messages by A and shortly after inbound messages to B represent communication between these entities. Obvious solutions such as bogus traffic generation or delayed message relaying have the drawback of inefficiency or directly impairing user experience.

However, these academic considerations should not conceal that there is a consensus among the scientific community favoring end-to-end encryption for privacy preservation. During the time of origin of this thesis this idea has become popular and is showing in the fact that several privacy-focusing instant messengers (e.g. Threema[2], Signal[3] or Telegram[4]) are publicly available and used by millions of users. With WhatsApp and Facebook Messenger employing the Signal protocol for private messaging, end-to-end encryption has become mainstream with being available to more than 1 billion users. [25]

As of 2018, distributed online social networks have not reached a dominant position. Diaspora is the second largest federation with an estimated number of accounts between 300,000[5] and 600,000[6]. It has been surpassed by Mastodon with 1,8 million[7] accounts. Unlike Diaspora, Mastodon is based on the W3C recommendation ActivityPub [129] which could foster alternative implementations. The ongoing scandal about Cambridge Analytica's misuse of Facebook data in the 2016 US presidential election campaign and the Brexit referendum have caused dissatisfaction with Facebook and centralized OSN platforms in general. While federated OSNs could profit from this, technological properties are not a guarantor for user attraction. Their main motivation for OSN usage is "keeping in touch" [63] with acquaintances which requires a critical mass of participants. Whether these approaches will attract enough users to fulfill this purpose remains open and highly depends on perceived privacy of centralized OSNs.

[2] `https://threema.ch/en`. Accessed 2018-12-16.
[3] `https://signal.org/`. Accessed 2018-12-16.
[4] `https://telegram.org/`. Accessed 2018-12-16.
[5] `https://diasp.eu/stats`, Accessed 2018-12-16.
[6] `https://fediverse.party/en/diaspora/`, Accessed 2018-12-16.
[7] `https://mnm.social/`, Accessed 2018-12-16.

Sensors

In this thesis, we have demonstrated how unrestricted access to low-level sensor readings can be used for privacy violations. This concern is currently being considered by the W3C in their approach towards a Generic Sensor API [112] which unifies previous sensor APIs used in this thesis. Although this does not directly mitigate the specific vulnerabilities found in this work, the generic specification features a comprehensive list of privacy considerations and mitigation strategies which partially overlap with our suggestions (e.g. reducing sampling frequency or accuracy).

Still, the suggestions in this specification framework are generic by nature and have to be adapted for each sensor. As of December 2018, none of the major browsers are released with an enabled Generic Sensor based implementation of the Ambient Light API [65]. Whether the enumeration of privacy considerations in the generic specification will actually lead to privacy improvements remains to be seen. In the meantime, privacy-aware users should consider utilizing a feature-reduced web browser with a focus on privacy.

Bibliography

[1] C. Alexander and I. Goldberg. „Improved user authentication in off-the-record messaging“. In: *Proceedings of the 2007 ACM workshop on Privacy in electronic society*. DOI: 10.1145/1314333.1314340.

[2] H. Aoki, B. Schiele, and A. Pentlan. „Realtime personal positioning system for a wearable computer“. In: *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*. IEEE. 1999, pp. 37–43.

[3] A. Arnbak, H. Asghari, M. Van Eeten, and N. Van Eijk. „Security Collapse in the HTTPS Market“. In: *Commun. ACM* 57.10 (Sept. 2014), pp. 47–55. ISSN: 0001-0782. URL: http://doi.acm.org/10.1145/2660574.

[4] M. Azizyan, I. Constandache, and R. Roy Choudhury. „SurroundSense: Mobile Phone Localization via Ambience Fingerprinting“. In: *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*. MobiCom '09. Beijing, China: ACM, 2009, pp. 261–272. ISBN: 978-1-60558-702-8. URL: http://doi.acm.org/10.1145/1614320.1614350.

[5] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. „Persona: an online social network with user-defined privacy“. In: *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. ISBN: 978-1-60558-594-9. DOI: 10.1145/1592568.1592585.

[6] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten. *Automatic Certificate Management Environment (ACME)*. Internet-Draft draft-ietf-acme-acme-09. http://www.ietf.org/internet-drafts/draft-ietf-acme-acme-09.txt. IETF Secretariat, Dec. 2017.

[7] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten. *Automatic Certificate Management Environment (ACME)*. Internet-Draft draft-ietf-acme-acme-12. http://www.ietf.org/internet-drafts/draft-ietf-acme-acme-12.txt. IETF Secretariat, Apr. 2018.

[8] D. Basin, C. Cremers, T. H. J. Kim, A. Perrig, R. Sasse, and P. Szalachowski. „Design, Analysis, and Implementation of ARPKI: An Attack-Resilient Public-Key Infrastructure“. In: *IEEE Transactions on Dependable and Secure Computing* 15.3 (May 2018), pp. 393–408. ISSN: 1545-5971. DOI: 10.1109/TDSC.2016.2601610.

[9] D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski. „ARPKI: Attack Resilient Public-Key Infrastructure“. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS '14. Scottsdale, Arizona, USA: ACM, 2014, pp. 382–393. ISBN: 978-1-4503-2957-6. URL: http://doi.acm.org/10.1145/2660267.2660298.

[10] K. Bhargavan, A. Delignat-Lavaud, and N. Kobeissi. „Formal Modeling and Verification for Domain Validation and ACME“. In: *Financial Cryptography and Data Security*. Ed. by A. Kiayias. Cham: Springer International Publishing, 2017, pp. 561–578. ISBN: 978-3-319-70972-7.

[11] R. Bhaumik and A. Kostiainen. *Proximity Sensor*. W3C Working Draft. W3C, July 2016. URL: http://www.w3.org/TR/2016/WD-proximity-20160719/.

[12] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, and H. Zhang. „The growth of Diaspora - A decentralized online social network in the wild“. In: *Computer Communications Workshops, 2012 IEEE Conference on*. DOI: 10.1109/INFCOMW.2012.6193476.

[13] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal. „Bamboozling Certificate Authorities with BGP“. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 833–849. ISBN: 978-1-931971-46-1.

[14] J. R. Blum, D. G. Greencorn, and J. R. Cooperstock. „Smartphone sensor reliability for augmented reality applications“. In: *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer. 2012, pp. 127–138.

[15] R. Bodmeier, D. Scheck, and K. Lieber. „Mobile Eats the Retail World“. In: *Handel mit Mehrwert*. Springer, 2019, pp. 135–152.

[16] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna. „Cloud strife: mitigating the security risks of domain-validated certificates“. In: *Proceedings of NDSS'18*. Internet Society, Feb. 2018. DOI: 10.14722/ndss.2018.23327.

[17] R. Bracewell. *Fourier Analysis and Imaging*. Springer US, 2004. ISBN: 978-0-306-48187-1.

[18] J. F. Brunelle, M. Kelly, M. C. Weigle, and M. L. Nelson. „The impact of JavaScript on archivability“. In: *International Journal on Digital Libraries* 17.2 (June 2016), pp. 95–117. ISSN: 1432-1300. URL: https://doi.org/10.1007/s00799-015-0140-8.

[19] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. „PeerSoN: P2P Social Networking - Early Experiences and Insights“. In: *Proceedings of the Second ACM Workshop on Social Network Systems Social Network Systems 2009, co-located with Eurosys 2009*.

[20] B. Budington. *Symantec Issues Rogue EV Certificate for Google.com*. Accessed: 2018-06-27. Sept. 2015. URL: https://www.eff.org/deeplinks/2015/09/symantec-issues-rogue-ev-certificate-googlecom.

[21] CA/B Forum. *Baseline Requirements Documents*. Accessed: 2018-06-27. URL: https://cabforum.org/baseline-requirements-documents/.

[22] M. Caceres and M. Lamouri. *The Screen Orientation API*. W3C Working Draft. W3C, Oct. 2018. URL: https://www.w3.org/TR/2018/WD-screen-orientation-20181012/.

[23] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. „A review of audio fingerprinting". In: *Journal of VLSI signal processing systems for signal, image and video technology* 41.3 (2005), pp. 271–284.

[24] B. Clarkson, K. Mase, and A. Pentland. „Recognizing user context via wearable sensors". In: *Wearable Computers, The Fourth International Symposium on*. Oct. 2000, pp. 69–75. DOI: 10.1109/ISWC.2000.888467.

[25] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. „A formal security analysis of the signal messaging protocol". In: *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*. IEEE. 2017, pp. 451–466.

[26] Commission Internationale de L'Eclairage. *The Basis of Physical Photometry, 2nd ed. (CIE 018.2-1983)*. Jan. 1983, p. 49. ISBN: 978-9-290-34018-8.

[27] 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, May 2008.

[28] M. Cui, Z. Cao, and G. Xiong. „How Is the Forged Certificates in the Wild: Practice on Large-Scale SSL Usage Measurement and Analysis". In: *Computational Science – ICCS 2018*. Ed. by Y. Shi, H. Fu, Y. Tian, V. V. Krzhizhanovskaya, M. H. Lees, J. Dongarra, and P. M. A. Sloot. Cham: Springer International Publishing, 2018, pp. 654–667. ISBN: 978-3-319-93713-7.

[29] L. Cutillo, R. Molva, and M. Onen. „Safebook: A distributed privacy preserving Online Social Network". In: *World of Wireless, Mobile and Multimedia Networks, 2011 IEEE International Symposium on a*. DOI: 10.1109/WoWMoM.2011.5986118.

[30] S. Czesla. *PyAstronomy*. https://github.com/sczesla/PyAstronomy. Accessed on 2018-05-16.

[31] I. Dacosta, M. Ahamad, and P. Traynor. „Trust No One Else: Detecting MITM Attacks against SSL/TLS without Third-Parties". In: *Computer Security – ESORICS 2012*. Ed. by S. Foresti, M. Yung, and F. Martinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 199–216. ISBN: 978-3-642-33167-1.

[32] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and W. Lee. „Increased DNS Forgery Resistance Through 0x20-bit Encoding: Security via Leet Queries“. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*. CCS '08. Alexandria, Virginia, USA: ACM, 2008, pp. 211–222. ISBN: 978-1-59593-810-7. URL: http://doi.acm.org/10.1145/1455770.1455798.

[33] T. Dalenius. „Towards a methodology for statistical disclosure control“. In: *Statistik Tidskrift* 15.429-444 (1977), pp. 2–1.

[34] d. m. boyd danah m. and N. B. Ellison. „Social Network Sites: Definition, History, and Scholarship“. In: *Journal of Computer-Mediated Communication* 13.1 (Oct. 2007), pp. 210–230. ISSN: 1083-6101. URL: https://dx.doi.org/10.1111/j.1083-6101.2007.00393.x.

[35] A. Delignat-Lavaud, M. Abadi, A. Birrell, I. Mironov, T. Wobber, and Y. Xie. „Web PKI: Closing the Gap between Guidelines and Practices“. In: *Proceedings of NDSS'14*. Internet Society, Feb. 2014. DOI: 10.14722/ndss.2014.23305.

[36] R. Dewri, P. Annadata, W. Eltarjaman, and R. Thurimella. „Inferring Trip Destinations from Driving Habits Data“. In: *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*. WPES '13. Berlin, Germany: ACM, 2013, pp. 267–272. ISBN: 978-1-4503-2485-4. URL: http://doi.acm.org/10.1145/2517840.2517871.

[37] Diaspora Inc. *https://joindiaspora.com/*. 2012. URL: http://onesocialweb.org/.

[38] V. Dukhovni and W. Hardaker. *SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)*. RFC 7672. RFC Editor, Oct. 2015.

[39] M. Dürr, M. Werner, and M. Maier. „Re-Socializing Online Social Networks“. In: *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. Dec. 2010, pp. 786–791. DOI: 10.1109/GreenCom-CPSCom.2010.18.

[40] M. Dürr, M. Maier, and F. Dorfmeister. „Vegas - A Secure and Privacy-Preserving Peer-to-Peer Online Social Network“. In: *Social Computing, 2012 IEEE Fourth International Conference on*.

[41] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. „Analysis of the HTTPS Certificate Ecosystem“. In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. Barcelona, Spain: ACM, 2013, pp. 291–304. ISBN: 978-1-4503-1953-9. URL: http://doi.acm.org/10.1145/2504730.2504755.

[42] W. H. Dutton, N. B. Ellison, and D. M. Boyd. *Sociality Through Social Network Sites*. Mar. 2013. URL: http://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199589074.001.0001/oxfordhb-9780199589074-e-8.

[43] C. Dwork. „Differential Privacy“. In: *Automata, Languages and Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. ISBN: 978-3-540-35908-1.

[44] D. Eastlake 3rd and M. Andrews. *Domain Name System (DNS) Cookies*. RFC 7873. RFC Editor, May 2016.

[45] D. Eastlake. *Transport Layer Security (TLS) Extensions: Extension Definitions*. RFC 6066. http://www.rfc-editor.org/rfc/rfc6066.txt. RFC Editor, Jan. 2011.

[46] S. Friedl, A. Popov, A. Langley, and E. Stephan. *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*. RFC 7301. http://www.rfc-editor.org/rfc/rfc7301.txt. RFC Editor, July 2014.

[47] K. Fujiwara, A. Kato, and W. Kumari. *Aggressive Use of DNSSEC-Validated Cache*. RFC 8198. RFC Editor, July 2017.

[48] X. Gao, B. Firner, S. Sugrim, V. Kaiser-Pendergrast, Y. Yang, and J. Lindqvist. „Elastic Pathing: Your Speed is Enough to Track You“. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. Seattle, Washington: ACM, 2014, pp. 975–986. ISBN: 978-1-4503-2968-2. URL: http://doi.acm.org/10.1145/2632048.2632077.

[49] X. Gao, B. Firner, S. Sugrim, V. Kaiser-Pendergrast, Y. Yang, and J. Lindqvist. „Transforming Speed Sequences into Road Rays on the Map with Elastic Pathing“. In: *CoRR* abs/1710.06932 (2017). arXiv: 1710.06932. URL: http://arxiv.org/abs/1710.06932.

[50] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz. „LifeSocial.KOM: A secure and P2P-based solution for online social networks“. In: *Consumer Communications and Networking Conference, 2011 IEEE*. DOI: 10.1109/CCNC.2011.5766541.

[51] B. Greschbach, G. Kreitz, and S. Buchegger. „The devil is in the metadata - New privacy challenges in Decentralised Online Social Networks“. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. 2012, pp. 333–339. DOI: 10.1109/PerComW.2012.6197506.

[52] T. Guan, Y. He, J. Gao, J. Yang, and J. Yu. „On-Device Mobile Visual Location Recognition by Integrating Vision and Inertial Sensors“. In: *IEEE Transactions on Multimedia* 15.7 (Nov. 2013), pp. 1688–1699. ISSN: 1520-9210. DOI: 10.1109/TMM.2013.2265674.

[53] J. Haitsma and T. Kalker. „A Highly Robust Audio Fingerprinting System.“ In: *ISMIR*. Vol. 2002. 2002, pp. 107–115.

[54] P. Hallam-Baker and R. Stradling. *DNS Certification Authority Authorization (CAA) Resource Record*. RFC 6844. RFC Editor, Jan. 2013.

[55] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang. „ACComplice: Location inference using accelerometers on smartphones“. In: *Fourth International Conference on Communication Systems and Networks, COMSNETS 2012, Bangalore, India, January 3-7, 2012*. Ed. by K. K. Ramakrishnan, R. Shorey, and D. F. Towsley. IEEE, 2012, pp. 1–9. ISBN: 978-1-4673-0296-8. URL: https://doi.org/10.1109/COMSNETS.2012.6151305.

[56] 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, Aug. 2012.

[57] P. Hoffman and J. Schlyter. *Using Secure DNS to Associate Certificates with Domain Names for S/MIME*. RFC 8162. RFC Editor, May 2017.

[58] R. Holz, L. Braun, N. Kammenhuber, and G. 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*. IMC '11. Berlin, Germany: ACM, 2011, pp. 427–444. ISBN: 978-1-4503-1013-0. URL: http://doi.acm.org/10.1145/2068816.2068856.

[59] J. Hua, Z. Shen, and S. Zhong. „We Can Track You if You Take the Metro: Tracking Metro Riders Using Accelerometers on Smartphones“. In: *IEEE Transactions on Information Forensics and Security* 12.2 (Feb. 2017), pp. 286–297. ISSN: 1556-6013. DOI: 10.1109/TIFS.2016.2611489.

[60] L. S. Huang, A. Rice, E. Ellingsen, and C. Jackson. „Analyzing Forged SSL Certificates in the Wild“. In: *2014 IEEE Symposium on Security and Privacy*. May 2014, pp. 83–97. DOI: 10.1109/SP.2014.13.

[61] A. Hubert and R. van Mook. *Measures for Making DNS More Resilient against Forged Answers*. RFC 5452. RFC Editor, Jan. 2009.

[62] C. Jackson, D. R. Simon, D. S. Tan, and A. Barth. „An Evaluation of Extended Validation and Picture-in-Picture Phishing Attacks“. In: *Financial Cryptography and Data Security*. Ed. by S. Dietrich and R. Dhamija. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 281–293. ISBN: 978-3-540-77366-5.

[63] A. N. Joinson. „Looking at, Looking Up or Keeping Up with People?: Motives and Use of Facebook“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 1027–1036. ISBN: 978-1-60558-011-1. URL: http://doi.acm.org/10.1145/1357054.1357213.

[64] K. Kapinchev, A. Bradu, F. Barnes, and A. Podoleanu. „GPU implementation of cross-correlation for image generation in real time“. In: *2015 9th International Conference on Signal Processing and Communication Systems (ICSPCS)*. Dec. 2015, pp. 1–6. DOI: 10.1109/ICSPCS.2015.7391783.

[65] A. Kostiainen. *Ambient Light Sensor*. Candidate Recommendation. W3C, Mar. 2018. URL: https://www.w3.org/TR/2018/CR-ambient-light-20180320/.

[66] A. Kostiainen. *Ambient Light Sensor*. https://w3c.github.io/ambient-light/. Accessed on 2018-05-30.

[67] A. Kostiainen and M. Lamouri. *Battery Status API*. Candidate Recommendation. W3C, July 2016. URL: http://www.w3.org/TR/2016/CR-battery-status-20160707/.

[68] A. Kostiainen, T. Langel, and D. Turner. *Ambient Light Sensor*. W3C Working Draft. W3C, Aug. 2017. URL: https://www.w3.org/TR/2017/WD-ambient-light-20170814/.

[69] D. Kumar, Z. Wang, M. Hyder, J. Dickinson, G. Beck, D. Adrian, J. Mason, Z. Durumeric, J. A. Halderman, and M. Bailey. „Tracking Certificate Misissuance in the Wild". In: *2018 IEEE Symposium on Security and Privacy (SP)*. Vol. 00. 2018, pp. 288–301. URL: doi.ieeecomputersociety.org/10.1109/SP.2018.00015.

[70] D. Kundur and K. Karthik. „Video fingerprinting and encryption principles for digital rights management". In: *Proceedings of the IEEE* 92.6 (2004), pp. 918–932.

[71] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. „Activity Recognition Using Cell Phone Accelerometers". In: *SIGKDD Explor. Newsl.* 12.2 (Mar. 2011), pp. 74–82. ISSN: 1931-0145. URL: http://doi.acm.org/10.1145/1964897.1964918.

[72] T. Langel and A. Kostiainen. *Ambient Light Events*. W3C Working Draft. Mar. 2016. URL: https://www.w3.org/TR/2016/WD-ambient-light-20160329/.

[73] B. Laurie, A. Langley, and E. Kasper. *Certificate Transparency*. RFC 6962. RFC Editor, June 2013.

[74] B. Laurie, G. Sisson, R. Arends, and D. Blacka. *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*. RFC 5155 (Proposed Standard). IETF.

[75] S. Lee and C. D. Yoo. „Robust video fingerprinting for content-based video identification". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 18.7 (2008), pp. 983–988.

[76] L. Li, P. Hu, C. Peng, G. Shen, and F. Zhao. „Epsilon: A Visible Light Based Positioning System". In: *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, 2014, pp. 331–343. ISBN: 978-1-931971-09-6.

[77] Z. Li, Q. Pei, I. Markwood, Y. Liu, M. Pan, and H. Li. „Location Privacy Violation via GPS-agnostic Smart Phone Car Tracking". In: *IEEE Transactions on Vehicular Technology* (2018), pp. 1–1. ISSN: 0018-9545. DOI: 10.1109/TVT.2018.2800123.

[78] W. C. Ma, S. Wang, M. A. Brubaker, S. Fidler, and R. Urtasun. „Find your way by observing the sun and other semantic cues“. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 6292–6299. DOI: 10.1109/ICRA.2017.7989744.

[79] D. Margolis, M. Risher, B. Ramakrishnan, A. Brotman, and J. Jones. *SMTP MTA Strict Transport Security (MTA-STS)*. Internet-Draft draft-ietf-uta-mta-sts-13. http://www.ietf.org/internet-drafts/draft-ietf-uta-mta-sts-13.txt. IETF Secretariat, Dec. 2017.

[80] S. Matsumoto and R. M. Reischuk. „Certificates-as-an-Insurance: Incentivizing accountability in SSL/TLS“. In: *SENT'15*. Internet Society, 2015. DOI: 10.14722/sent.2015.23009.

[81] A. Menezes, J. Katz, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. Discrete Mathematics and Its Applications. CRC Press, 1996. ISBN: 9781439821916. URL: https://books.google.de/books?id=MhvcBQAAQBAJ.

[82] A. Mertins. *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*. Wiley, 1999. ISBN: 9780471986263.

[83] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly. „PowerSpy: Location Tracking Using Mobile Device Power Analysis“. In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015, pp. 785–800. ISBN: 978-1-931971-232.

[84] Mozilla Foundation. *Mozilla Root Store Policy*. Accessed: 2018-06-27. June 2017. URL: https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/policy/.

[85] K. E. Murray and R. Waller. „Social networking goes abroad“. In: *International Educator* 16.3 (2007), p. 56.

[86] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir. „Inferring User Routes and Locations Using Zero-Permission Mobile Sensors“. In: *2016 IEEE Symposium on Security and Privacy (SP)*. May 2016, pp. 397–413. DOI: 10.1109/SP.2016.31.

[87] S. B. Needleman and C. D. Wunsch. „A general method applicable to the search for similarities in the amino acid sequence of two proteins“. In: *Journal of Molecular Biology* 48.3 (Mar. 1970), pp. 443–453. URL: https://doi.org/10.1016/0022-2836(70)90057-4.

[88] M. Nieles, K. Dempsey, and V. Y. Pillitteri. *An introduction to information security*. Tech. rep. June 2017. URL: https://doi.org/10.6028/nist.sp.800-12r1.

[89] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia. „Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching“. In: *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*. CoNEXT '12. Nice, France: ACM, 2012, pp. 337–348. ISBN: 978-1-4503-1775-7. URL: http://doi.acm.org/10.1145/2413176.2413215.

[90] J. Oostveen, T. Kalker, and J. Haitsma. „Feature extraction and a database strategy for video fingerprinting“. In: *Recent Advances in Visual Information Systems*. Springer, 2002, pp. 117–128.

[91] *Parameter values for the HDTV standards for production and international programme exchange*. RECOMMENDATION ITU-R BT.709-6. International Telecommunication Union - Radiocommunication. Geneva, July 2015. URL: http://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-6-201506-I!!PDF-E.pdf.

[92] A. Pears. „Chapter 7: Appliance technologies and scope for emission reduction“. In: *Strategic Study of Household Energy and Greenhouse Issues*. Australian Greenhouse Office. 1998, p. 61.

[93] Pew Research Center: Internet, Science & Tech. *Mobile Fact Sheet*. Accessed: 2019-02-26. 2019. URL: http://www.pewinternet.org/fact-sheet/mobile/.

[94] A. Popescu. *Geolocation API Specification*. W3C Editor's Draft. July 2014. URL: http://dev.w3.org/geo/api/spec-source.html.

[95] A. Popescu. *Geolocation API Specification 2nd Edition*. W3C Recommendation. W3C, Nov. 2016. URL: https://www.w3.org/TR/2016/REC-geolocation-API-20161108/.

[96] F. Raji, A. Miri, M. Jazi, and B. Malek. „Online Social Network with flexible and dynamic privacy policies“. In: *Computer Science and Software Engineering, 2011 CSI International Symposium on*. DOI: 10.1109/CSICSSE.2011.5963982.

[97] N. Ravi, P. Shankar, A. Frankel, A. Elgammal, and L. Iftode. „Indoor Localization Using Camera Phones“. In: *Seventh IEEE Workshop on Mobile Computing Systems Applications (WMCSA'06 Supplement)*. Vol. Supplement. Aug. 2006, pp. 1–7. DOI: 10.1109/WMCSA.2006.4625206.

[98] N. Ravi and L. Iftode. „Fiatlux: Fingerprinting rooms using light intensity“. In: *Advances in Pervasive Computing: Adjunct Proceedings of the 5th International Conference on Pervasive Computing*. 2007.

[99] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. RFC Editor, Aug. 2018.

[100] F. Rivoal and T. Atkins Jr. *Media Queries Level 4*. W3C Editor's Draft. Apr. 2016. URL: https://drafts.csswg.org/mediaqueries-4/.

[101] G. Schaefer and M. Rossberg. *Security in Fixed and Wireless Networks*. New York: John Wiley & Sons, 2016. ISBN: 978-1-119-04074-3.

[102] Q. Scheitle, T. Chung, J. Hiller, O. Gasser, J. Naab, R. van Rijswijk-Deij, O. Hohlfeld, R. Holz, D. Choffnes, A. Mislove, and G. Carle. „A First Look at Certification Authority Authorization (CAA)“. In: *SIGCOMM Comput. Commun. Rev.* 48.2 (May 2018), pp. 10–23. ISSN: 0146-4833. URL: http://doi.acm.org/10.1145/3213232.3213235.

[103] B. Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. Wiley, 2015. ISBN: 1119096723.

[104] C. Schuss, T. Leikanger, B. Eichberger, and T. Rahkonen. „Efficient use of solar chargers with the help of ambient light sensors on smartphones“. In: *Open Innovations Association (FRUCT16), 2014 16th Conference of*. IEEE. 2014, pp. 79–85.

[105] L. Schwittmann, V. Matkovic, M. Wander, and T. Weis. „Video recognition using ambient light sensors“. In: *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Mar. 2016, pp. 1–9. DOI: 10.1109/PERCOM.2016.7456511.

[106] L. Schwittmann, M. Wander, and T. Weis. „Domain Impersonation is Feasible: A Study of CA Domain Validation Vulnerabilities“. In: *2019 IEEE European Symposium on Security and Privacy (EuroS & P)*. June 2019, pp. 544–559. DOI: 10.1109/EuroSP.2019.00046.

[107] L. Schwittmann, C. Boelmann, V. Matkovic, M. Wander, and T. Weis. „Identifying TV Channels & On-Demand Videos using Ambient Light Sensors“. In: *Pervasive and Mobile Computing* 38 (2017). Special Issue IEEE International Conference on Pervasive Computing and Communications (PerCom) 2016, pp. 363–380. ISSN: 1574-1192. URL: http://www.sciencedirect.com/science/article/pii/S1574119216302085.

[108] L. Schwittmann, C. Boelmann, M. Wander, and T. Weis. „SoNet – Privacy and Replication in Federated Online Social Networks“. In: *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*. July 2013, pp. 51–57. DOI: 10.1109/ICDCSW.2013.20.

[109] L. Schwittmann, M. Wander, C. Boelmann, and T. Weis. „Privacy Preservation in Decentralized Online Social Networks“. In: *Internet Computing, IEEE* 18.2 (Mar. 2014), pp. 16–23. ISSN: 1089-7801. DOI: 10.1109/MIC.2013.131.

[110] L. Schwittmann, M. Wander, and T. Weis. „Mobile Devices as Digital Sextants for Zero-Permission Geolocation“. In: *Proceedings of the 5th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*. INSTICC. SciTePress, 2019, pp. 55–66. ISBN: 978-989-758-359-9. DOI: 10.5220/0007254000550066.

[111] *Sensors Overview*. Google Inc. URL: http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[112] A. Shalamov, M. Pozdnyakov, and R. Waldron. *Generic Sensor API*. Candidate Recommendation. W3C, Mar. 2018. URL: https://www.w3.org/TR/2018/CR-generic-sensor-20180320/.

[113] Y. Sheffer, R. Holz, and P. Saint-Andre. *Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. RFC 7525. http://www.rfc-editor.org/rfc/rfc7525.txt. RFC Editor, May 2015.

[114] R. Shirey. *Internet Security Glossary, Version 2*. RFC 4949. http://www.rfc-editor.org/rfc/rfc4949.txt. RFC Editor, Aug. 2007.

[115] R. Shoemaker. *ACME TLS ALPN Challenge Extension*. Internet-Draft draft-ietf-acme-tls-alpn-01. http://www.ietf.org/internet-drafts/draft-ietf-acme-tls-alpn-01.txt. IETF Secretariat, May 2018.

[116] T. Smith and M. Waterman. „Identification of common molecular subsequences". In: *Journal of Molecular Biology* 147.1 (Mar. 1981), pp. 195–197. URL: https://doi.org/10.1016/0022-2836(81)90087-5.

[117] R. Spreitzer. „PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices". In: *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*. Scottsdale, Arizona, USA: ACM, 2014, pp. 51–62. ISBN: 978-1-4503-3155-5. DOI: 10.1145/2666620.2666622.

[118] W. Stallings. *Cryptography and Network Security: Principles and Practice, Global Edition*. Pearson Education Limited, 2016. ISBN: 1292158581.

[119] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. „Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning". In: *2016 IEEE Symposium on Security and Privacy (SP)*. May 2016, pp. 526–545. DOI: 10.1109/SP.2016.38.

[120] T. K. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J. R. Ohm, and G. J. Sullivan. „Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance". In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (Jan. 2016), pp. 76–90. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2015.2477916.

[121] I. Telegraph and T. C. Committee. *CCITT Recommendation X.800: Data Communication Networks: Open Systems Interconnection (OSI); Security, Structure and Applications : Security Architecture for Open Systems Interconnection for CCITT Applications*. International Telecommunication Union, 1991. URL: https://books.google.de/books?id=gEXWHwAACAAJ.

[122] The Tor Project. *Orfox: Tor Browser for Android*. https://guardianproject.info/apps/orfox/. Accessed on 2018-05-30.

[123] R. Tibbett, T. Volodine, S. Block, and A. Popescu. *DeviceOrientation Event Specification*. W3C Candidate Recommendation. W3C, Aug. 2016. URL: https://www.w3.org/TR/2016/CR-orientation-event-20160818/.

[124] S. Triukose, S. Ardon, A. Mahanti, and A. Seth. „Geolocating IP Addresses in Cellular Data Networks“. In: *Passive and Active Measurement: 13th International Conference, PAM 2012, Vienna, Austria, March 12-14th, 2012. Proceedings*. Ed. by N. Taft and F. Ricciato. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 158–167. ISBN: 978-3-642-28537-0. URL: http://dx.doi.org/10.1007/978-3-642-28537-0_16.

[125] UNESCO and University of Oxford (UK). *World trends in freedom of expression and media development - global report 2017/2018*. Paris: UNESCO Publishing, 2018. ISBN: 978-9-231-00242-7.

[126] G. Urdaneta, G. Pierre, and M. V. Steen. „A survey of DHT security techniques“. In: *ACM Comput. Surv.* 43.2 (Feb. 2011), 8:1–8:49. ISSN: 0360-0300. URL: http://doi.acm.org/10.1145/1883612.1883615.

[127] T. Vissers, T. Barron, T. Van Goethem, W. Joosen, and N. Nikiforakis. „The Wolf of Name Street: Hijacking Domains Through Their Nameservers“. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: ACM, 2017, pp. 957–970. ISBN: 978-1-4503-4946-8. URL: http://doi.acm.org/10.1145/3133956.3133988.

[128] M. Wander. „Measurement survey of server-side DNSSEC adoption“. In: *2017 Network Traffic Measurement and Analysis Conference (TMA)*. June 2017. DOI: 10.23919/TMA.2017.8002913.

[129] C. Webber and J. Tallon. *ActivityPub*. W3C Recommendation. W3C, Jan. 2018. URL: https://www.w3.org/TR/2018/REC-activitypub-20180123/.

[130] P. Wouters. *DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP*. RFC 7929. RFC Editor, Aug. 2016.

[131] W. Xu, X. Zhou, and L. Li. „Inferring privacy information via social relations“. In: *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*. 2008, pp. 525–530. DOI: 10.1109/ICDEW.2008.4498373.

[132] J. Yu, V. Cheval, and M. Ryan. „DTKI: A New Formalized PKI with Verifiable Trusted Parties“. In: *The Computer Journal* 59.11 (2016), pp. 1695–1713. URL: http://dx.doi.org/10.1093/comjnl/bxw039.

[133] L. Zhou, Q. Chen, Z. Luo, H. Zhu, and C. Chen. „Speed-Based Location Tracking in Usage-Based Automotive Insurance“. In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. June 2017, pp. 2252–2257. DOI: 10.1109/ICDCS.2017.278.

[134] X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt. „Identity, Location, Disease and More: Inferring Your Secrets from Android Public Resources“. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Berlin, Germany: ACM, 2013, pp. 1017–1028. ISBN: 978-1-4503-2477-9. URL: http://doi.acm.org/10.1145/2508859.2516661.

[135] W. T. Zhu and J. Lin. „Generating Correlated Digital Certificates: Framework and Applications“. In: *IEEE Transactions on Information Forensics and Security* 11.6 (June 2016), pp. 1117–1127. ISSN: 1556-6013. DOI: 10.1109/TIFS.2016.2516818.