Master Thesis

PortuLock: Privacy Preserving OpenPGP Keyserver and Certificate Authority for Organizations

Master Computer Science
Faculty for Electrical Engineering and Computer Science
Hochschule Ravensburg-Weingarten
University of Applied Sciences

Author: Erik Escher
Matriculation Number: 33443

Time Period: 2021-06-01 - 2021-11-30
Date: 2021-11-30
Primary Examiner: Prof. Dr. rer. nat. Tobias Eggendorfer
Secondary Examiner: Prof. Dr. rer. nat. Markus Pfeil

Statement of Originality

I hereby confirm to have created the accompanying document by myself, without contributions from any sources other than those cited in the text and acknowledgments. This document has not been provided in similar or identical form to any other examination authority. It has not been published either.

2021-11-23 in Friedrichshafen ______________________
Erik Escher

Abstract

This thesis describes the design and implementation of a privacy preserving keyserver and certificate authority for organizations called PortuLock that solves the problems associated with privacy considerations around key distribution as well as the verification of keys for internal and external communication partners.

It allows users to generate keys, have them certified and published as well as manage their lifecycle on their own by authenticating with the organization's existing single-sign-on infrastructure to verify their name and prove membership in the organization.

Keys remain in the sole control of the user, while PortuLock's web generator takes care of complex processes such as creating trust signatures over certificate authority keys, that are not exposed by all clients.

On the privacy front it gives users full control over what information can be distributed with their certificate to protect against denial of service, distribution of problematic materials and to prevent leaking the social graph of certifications. It also ensures that data can only be published with approval from the subjects to which it pertains.

When it comes to identity validation, PortuLock leverages the network of trusted certificates spanned by the OpenPGP-CA project and library while making it accessible and usable for organizations of all sizes due to the scalable and easy self-service approach to certification.

PortuLock extends this network by providing an aggregation feature that requests keys for email addresses directly from the target domain using established protocols and without leaking personally identifiable information such as email addresses and fingerprints to third parties as is common with centralized keyservers. This aggregation feature allows administrators to configure trust based on the origin of a key and in turn issue certifications on the fly, extending this trust to clients and lifting the burden of verification from the users.

In summary, PortuLock makes OpenPGP usable for organizations by providing a scalable trust system, that users can easily join while also protecting their privacy and the privacy of communication partners by requesting information about certificates directly from the respective domains.

Contents

1. Introduction

1.1. Motivation

Multiple vulnerabilities in the recent years have exposed security issues with the current state of email systems.

Mail servers are by their very nature exposed to attackers and occasionally fail spectacularly [Sch21; Jog21; Wik21a; MS21; Zuc21]. When they do, they expose all mails stored in all mail accounts and all sorts of secrets within [Wik21a; MS21]. They also allow the attacker to intercept any password reset mails, which can be used to access most online accounts of any user.

Mail spoofing is also a common problem and used in dangerous and often expensive spear phishing and "CEO fraud" [BKA16] attacks. While there are systems to combat spoofing [RFC7489], these are designed to fight spam and are often insufficient to stop targeted attacks (see subsection 2.1.2).

Globalization, Digitalization, Home Office and the COVID-19 pandemic have often led to the adoption of "Digital Signature" systems consisting of drawing onto a virtual document or even printing and scanning it. These systems provide none of the security guarantees that should be in place for signatures and seem odd and antiquated to someone with knowledge of cryptographic signatures. Centralized (sometimes cloud-based) systems [Ado21; air21; Cit21; Zoh21; Doc21d] can provide some of these guarantees but also form a single point of failure and often rely on email verification to authenticate users.

What these problems have in common is that they could be solved or alleviated by deploying asymmetric cryptography systems that offer cryptographic signature capabilities for documents and emails and the ability to encrypt mails (and files) end-to-end and at-rest.

There are a few reasons (discussed in the next two subsections) why these systems are not widely used. This thesis aims to address some of them by developing a keyserver that handles privacy preserving key distribution and identity verification/certification.

1.1.1. Privacy Considerations around Key Distribution

To encrypt messages and validate signatures, the sender or relying party needs to have access to the public key of the recipient or signatory. While certificates could be sent manually as needed, this causes a lot of friction and hampers adoption. For use cases where many people want to validate signatures or communicate with varying partners this quickly becomes unreasonable.

On the other hand when openly publishing keys, which contain personally identifying information, care must be taken to reduce the exposure of this information to a feasible minimum.

1.1.2. Verifying Keys of Internal and External Communication Partners

Verifying someone's identity and that a certain key belongs to them over untrusted channels is a difficult and tedious process that scales badly with the number of potential contacts and typically involves synchronous communication. This project aims to let organizations confirm the identities of their members and securely trust the assertions of other organizations to build a large network of validated certificates that can be used without manual verification.

Certificates for users that don't belong to a trusted organization can still be retrieved and verified at a lower validation level by looking them up using the Web Key Directory/Discovery (WKD) protocol or from public key servers that verify email address control before publishing keys.

As organizations already have the necessary information about their members and an automated means of authenticating them, this provides an effective way of certifying keys without having to rely on complicated and expensive external parties.

1.2. Goals

The goal of this project is to develop a keyserver that can solve many of the problems currently holding back adoption of OpenPGP in organizations.

It should make generating, distributing, managing and verifying keys much easier for users and administrators in organizations and integrate with existing tools and other organizations where possible.

1.3. Document Structure

This thesis begins by providing useful definitions and technical background information before describing existing solutions for the problems and deriving requirements for the keyserver developed as part of this thesis. Chapter 5 provides an overview of the solution and describes it from the perspective of a user, before chapter 6 goes into more detail describing the technical implementation as well as various trade-offs and decisions made in the implementation of this keyserver.

Chapter 7 evaluates how well the requirements were met and how it compares to the existing solutions.

Chapter 8 describes further improvements that could be made to the keyserver before chapter 9 proposes improvements in the broader domain of OpenPGP and certification of asymmetric keys.

Chapter 10 describes the contribution to the OpenPGP ecosystem. Finally, chapter 11 highlights related projects that aim to increase the adoption of OpenPGP and provides an outlook on the future.

Abbreviations and terms used in this document are linked to their respective glossary entries (Appendix E) in the digital version of this thesis.

2. Definitions and Technical Background

This chapter provides some relevant definitions and technical background used throughout this thesis. In addition to the topics covered below, knowledge of Digital Certificates, Digital Signatures, the X.509 ecosystem and Asymmetric Encryption is required.

2.1. Necessity of Digital Signatures and Encryption

Digital signatures and secure end-to-end encrypted communication provide a number of benefits, some of which from the realm of communication are described below.

2.1.1. Lack of Protection Against Mail Server Compromise

While email is increasingly protected using transport encryption, though often only opportunistically allowing detectable attacks, unencrypted mails still reside on mail servers. These servers have proven vulnerable to a number of catastrophic zero-day exploits, providing full access to mailbox data (and the rest of the server) [Sch21; Jog21; Wik21a; MS21; Zuc21].

While encryption won't prevent compromise of the mail server it can significantly reduce its effects by protecting the content of mails both in transit and at rest even in the case of a full mail server compromise.

2.1.2. Lack of Protection Against Mail Spoofing

Despite efforts [RFC6376; RFC7208; RFC7489] to reduce it, mail spoofing [Wik21g] is still fairly easy [Ded21]. These efforts are often designed to combat spam, but their effectiveness seems limited against resourceful attackers performing one-off attacks against individual targets. Adoption of Domain-based Message Authentication, Reporting and Conformance (DMARC) (a crucial tool to ensure email authenticity) is estimated between 20 and 50 percent depending on sector, region and organization size [Ver19; MX 21; DMA20; DMA21; Naj20; Tru17]. Even of the organizations that have adopted DMARC about half request that recipients only report authentication failures rather than marking the email in question as spam or even rejecting it [MX 21; Ver19; DMA20; DMA21].

Mail servers often don't verify that an authenticated sender is authorized to use specific email addresses either. This allows members of an organization (or attackers that have compromised a member's credentials) and in some cases even others using the same hosted mail service to spoof emails with valid DomainKeys Identified Mail (DKIM) signatures and/or Sender Policy Framework (SPF) senders. [Wik21g]

Mail spoofing is often used in combination with social engineering to conduct "Business Email Compromise" [Wik21d] or "CEO Fraud" [BKA16] attacks, that can cost organizations many millions of Euros [Wik21d].

A lot of these problems could be solved by properly deploying existing email authentication technologies, however digital signatures on email can provide another layer of security. In particular email signatures can ensure integrity and authenticity of messages even if one or both mail servers are compromised. Additionally, digital signatures provide non-repudiation, meaning that the sender cannot deny having sent the email later.

2.1.3. "Digital Signature" Systems

The desire (or need) to establish digital communication methods for signatures combined with the unavailability of cryptographic signature systems for all but a few organizations and individuals has led to a number of "Digital Signature" systems that offer none of the security guarantees (authenticity, integrity, non-repudiation) typically desired for such systems.

Some of these systems consist of embedding the user's paper signature in a document while others even require documents to be printed, signed and scanned again. Others ask the user to draw a signature on their computer or generate "handwriting" based on the users typed name.

Document storage and transmission range from email, fax or file shares to centralized (sometimes cloud-based) services confirming that a certain user clicked a button or uploaded a document [Ado21; air21; Cit21; Zoh21; Doc21d]. The former provides little to no protection against manipulation, while the latter introduces stringent requirements on the security of the central service to protect the integrity of all documents and metadata.

While these systems are not much worse than traditional paper based signatures, which are also difficult to verify, they also miss out on significant opportunities.

Replacing these systems with proper digital signatures would make forgeries much harder while also reducing the requirements placed on the systems recording the signatures. It would improve the user experience by providing a uniform way of signing any document or file and ensure availability as well as verifiability even if the signature system becomes unavailable.

2.1.4. Secure Communication Outside the Regular Infrastructure

Digital certificates even allow secure communication in cases where the organization's regular infrastructure is unavailable or otherwise unsuitable. In this case or if other communications systems need to be used, that are not integrated with the organizations sign on system, members, might have trouble setting up secure communication channels and verifying each other's identity.

Digital certificates proving the holder's identity and organization membership can significantly improve this process by allowing members to sign, verify and encrypt statements and messages over any communication channel as long as they have access to their personal key. The certificates can also be exchanged without central infrastructure and members can authenticate each other even without prior contact.

2.2. PGP/GPG/OpenPGP

OpenPGP [RFC4880] is a specification describing a system for asymmetric encryption and signatures of messages and other objects using public and private keys. It also describes ways to exchange and verify digital certificates.

The terms PGP, OpenPGP and GNU Privacy Guard (GPG) are commonly used interchangeably to refer to the specification and ecosystem. PGP is a trademark currently held by Broadcom Inc. [Gle93; Wik21e; Wik21c] and is used for a commercial standard and implementation. OpenPGP is an open specification compatible with PGP, while GNU Privacy Guard (GPG) [g1020] is the name of the most commonly used client implementation. In the rest of this document OpenPGP will be used to refer to the specification, while the term "client" will be referred to any software running on the client side, which can consist of a number of programs working together.

For cryptographic material and associated data referring to an entity, the terms "key" and "certificate" are used interchangeably and do not indicate whether the cryptographic material has been certified by a Certificate Authority (CA).

This project uses OpenPGP extensively so readers should be familiar with it and some of its advanced features. A summary of the most important aspects for this project is provided below with references linking to further information.

2.3. OpenPGP Key Composition[1]

OpenPGP keys are constructed as a list of packets, that are bound together using signatures.

1 Portions of this section on OpenPGP Key Composition are based on my previous work on an OpenPGP-based anonymity system for Git called "git-anon" [Esc21]. Some subsections were adapted or extended and some sentences are copied.

The most important and first packet of a key is its primary key material. This primary key material is hashed to form the key's fingerprint, which is often used to uniquely identify the key. The primary key material forms the identity of the key and cannot be changed. Keys can also be identified by their 32- or 64-bit KeyID, which is derived from the fingerprint, however this is not cryptographically secure and therefore discouraged.

Other important packets include UserIDs, subkeys and the signatures that bind them to the primary key as well as certifications from third parties.

Different representations of a key can exist containing subsets of the full key. In particular different versions containing different UserIDs and certifications can be created and distributed.

An example of such a key is provided for illustration in Appendix A.

2.3.1. UserIDs

UserIDs are UTF-8 strings typically consisting of the user's full name and email address and describe the identity of the key holder. An OpenPGP key can contain any number of these UserIDs, which is often used to include multiple email addresses in the same key.

2.3.2. Subkeys

In addition to the primary key pair an OpenPGP key can contain additional keys used for various purposes. This allows for various advanced use cases such as offline primary keys, signature and encryption subkeys with shorter lifetimes or signature keys generated and stored on smart cards.

2.3.3. Signatures

OpenPGP Signatures can contain a number of metadata subpackets indicating additional information about the signature such as the issuers fingerprint, their UserID or theoretically even their full key which can be used in an attempt to retrieve the signature key. While issuer fingerprints are quite common, UserIDs are more rare and full keys don't seem to be used in practice.

Signatures can be used to sign arbitrary data such as emails or files. Special signatures called self-signatures are used to cryptographically connect packets to a key. These are required for any components that only the keyholder should be able to add (such as subkeys or UserIDs) and prevent attackers from manipulating such elements.

2.3.4. Certifications

Special signatures called "Certifications" can also be used to confirm the binding between a third party's UserIDs and their key. This is a statement to anyone verifying the certification, that the issuer believes that the UserIDs accurately describes the keyholder. These certifications are typically included as packets in the key that is certified and distributed along with it. While the number of certifications on a key is unlimited, performance issues can arise, which can be used in a denial-of-service attack [Gil19b; Han19; Fis19].

2.3.5. Revocation

Since packets cannot be removed effectively, additional packets need to be added to indicate if a particular component or the entire key should no longer be used. These packets are so-called “revocation signatures” issued by the primary key indicating that a certain component should no longer be used.

There are a number of reasons to revoke a key or parts of it. This thesis assumes the most severe one, namely the private key material being compromised for key revocations. In the case of certification revocations, the member might have left the organization the certification may have been issued erroneously.

As with the rest of the key, distributing these packets is often difficult and unreliable, which is especially problematic for revocation information. While omitting other packets provides little benefit to an attacker, omitting revocation information can allow them to continue using a compromised key even if it has already been revoked.

2.4. OpenPGP Trust and Verification[1]

As with all forms of asymmetric cryptography the most difficult aspect is verifying that the public key material to be used is actually from the intended recipient and not an attacker.

In this process the two terms “verification” and “trust” are commonly used interchangeably despite having a distinct meaning. In this document they are used as follows:

“Verification” is the process whereby a key’s actual holder (the person that can use the private key) is bidirectionally mapped to an entity known to the user under a specific name or UserID. Once the mapping has been verified, the user can certify the mapping to remember the decision and optionally publish it.

It’s easy to verify that the keyholder claims to be a specific entity as they are required to sign the respective UserID. The reverse mapping is more difficult and either involves direct communication between the user and the individual where the individual confirms this mapping or a third party that confirms this mapping. The latter can be done by downloading the key over a secure connection from a website that confirms the mapping or by verifying the certifications on the key.

1 This section is similarly based on my previous work on “git-anon” [Esc21]. It was adapted and extended significantly for this thesis and few sentences are copied.

Certifications are, as described above, claims from a third party that the mapping from an entity as described by the UserID to the key is correct. When checking a certification, the user needs to decide whether they "trust" the certifying party to have correctly verified the mapping themselves. This trust decision can be expressed through a trust signature that specifies the extent of the trust and optionally scopes it to a specific domain. This trust signature can be stored or published similar to a certification.

2.4.1. Web of Trust

Traditionally OpenPGP proposes a system of certifying and potentially trust signing the key of as many people as possible, to create a so-called "Web of Trust". One problem with this system is that unless both communication partners are strongly connected to this system, they are unlikely to find a path of certifications and trust signatures leading from one key to the other, or if they do it might be very long offering many points of attack. Another problem is, that it is fairly complex and uses concepts that normal users are typically not familiar with.

2.4.2. Manual Verification

In practice, this usually means that certificates are downloaded from a website associated with the entity and verified based on this source. Alternatively, users might simply use the certificate they find first and become suspicious if the certificate changes.

2.4.3. Certificate Authorities

Certifications can not only be issued by individuals but also by organizations specialized in doing so. These are referred to as "Certificate Authorities" in the X.509 certificate system.

While they are equally possible and compatible with the Web of Trust, there seem to be only few of them and their usage seems limited. CAs will typically certify a large number of UserIDs, meaning trust in a limited number of entities can allow for the verification of many certificates. Additionally, the services of a CA are often available to a large population making it easy to have a UserID certified by a CA that the recipient might trust.

A notable example is the CA [Gov20a] operated by Governikus [Gov20b] on behalf of the Bundesamt für Sicherheit in der Informationstechnik [BSI20], which uses the German National Identity Card [Wik21h; BMI21] and its online authentication capabilities to verify the keyholder's name before certifying the matching UserIDs on the key and emailing the certification to the respective email addresses.

2.4.4. Partial Trust

OpenPGP also supports a concept of partial trust, whereby multiple partially trusted introducers need to certify a UserID for it to be trusted. The rest of this document assumes that all certifications and trust signatures use full trust, however CAs can also be trusted partially and combined with certifications from other sources if desired.

2.5. OpenPGP Key Distribution

Numerous methods for distributing and retrieving OpenPGP certificates exist and can be combined as needed. Different subsets of the packets composing a key can be made available in different places. Clients will treat certificates as append-only lists of packets, meaning they will add new information to the list without ever removing anything.

2.5.1. Manual Methods

The most basic method for key exchange consists of exchanging keys in the form of files or text blocks in direct communication or providing the key for download on websites or third-party platforms. Direct communication provides the key holder with control over who receives which version of their key, while publication on a website obviates the need for communication during key exchange. Depending on the circumstances and communication method, this might also serve to verify the certificate.

One important drawback of these manual methods is the lack of updates when the key changes. Updating a revoked or expired key requires the user to go through the process again. In addition, this does not provide automatic discovery for the keys of communication partners.

2.5.2. Autocrypt

Autocrypt is (among other things) a system for automatic key exchanges through metadata on messages. Supporting clients will add their OpenPGP key together with some metadata in a special header of each outgoing message and parse incoming messages for such headers [Aut20a]. When a supporting client encounters such a header, it will automatically encrypt outgoing emails to the sender [Aut20a].

Support is too rare to be meaningful and mostly provided by obscure and rarely used clients [Aut20b], though more notable clients such as K-9 Mail, FairEmail (both for Android) and KMail [Aut20b; KDE21a] are also included.

To avoid undecryptable emails, an unencrypted message without an Autocrypt header is seen as indication that the user no longer uses Autocrypt and should therefore receive unencrypted emails [Aut20a].

In its current form it provides no protection against active adversaries capable of sending email that looks genuine to the client [Aut20a].

2.5.3. Traditional Keyservers

To simplify key exchange and provide a convenient way of searching for keys, a system of OpenPGP keyservers was created. These keyservers form an append-only distributed storage system for OpenPGP packets related to keys, that synchronize using email and allow lookups based on partial UserIDs, fingerprints or KeyIDs.

These keyservers were collectively known and organized as the “SKS Keyserver Network/Pool”, however the lack of software maintenance combined with key flooding attacks and GDPR [GDPR] deletion requests caused the servers to cease operation [Lee19; Web21; Böc21]. They could not comply with deletion requests due to the append-only, distributed and synchronizing nature of the servers.

They have been replaced by more servers operating the actively maintained “hockeypuck” software [Mar21; MCS21]. It operates almost identically but allows keys to be blacklisted in an attempt to handle deletion requests [Mar20]. Unfortunately, the problem of key flooding is only solved by not distributing large keys [Mar20] and with no way of verifying that the keyholder intended to have their key stored on the keyservers, issues with GDPR [GDPR] compliance remain.

The above keyservers are described as “traditional” in the rest of this thesis to differentiate them from the more “modern” versions described below.

2.5.4. Modern Keyservers

A number of more modern keyservers are available that attempt to solve the problems associated with traditional keyservers.

2.5.4.1. Hagrid (keys.openpgp.org)

One such alternative offering attempting to solve both the key flooding and verification issues is the keyserver operated at `https://keys.openpgp.org`. It runs the keyserver software “Hagrid” [Seq21c], which strips public keys of certifications before storing them and requires the keyholder to verify their email address before publishing UserIDs [Seq21d]. Keys without UserIDs and certifications are published freely without verification [Seq21d; Seq21e]. Stripping certifications solves the key flooding issues but also breaks the Web of Trust and other verification options as described above.

Users can delete UserIDs matching their email address from the server if desired, however the rest of the key cannot be deleted [Seq21d]. The operators of the keyserver consider the rest of the key as anonymous data, that can be retained without consent [Seq21f], despite it uniquely identifying the keyholder and the “anonymization” being reversible by adding the UserID again. A justification for retaining the rest of the key material could likely be construed in the case of revoked keys because distributing this revocation information is in the justified interest of the public and anyone encountering the key, which should outweigh the keyholder’s interests.

2.5.4.2. PGP Global Directory

A similar offering is the “PGP Global Directory” [Sym11a] provided by Symantec Corporation at `https://keyserver.pgp.com` using proprietary software. Their keyserver only accepts non-revoked, non-expired keys and only publishes them after verifying the associated email addresses. It allows users to delete keys when desired and requires confirmation that the email addresses are still valid every six months. [Sym11b]

When a UserID gets validated for publishing on the keyserver, the keyserver certifies it and publishes the certification acting as a CA that performs email validation. [Sym11b; Sym11c]

It solves the problem of key flooding by requiring certifying keys to also be on the keyserver, which is harder to automate due to the requirement of email verification [Sym11b].

2.5.4.3. Privacy

Both above mentioned keyservers prevent enumeration by only responding to exact matches. Symantec’s keyserver only returns results if the query matches exactly one key, while Hagrid only allows lookups based on the email address [Sym11b; Seq21d].

From a privacy perspective the problem with using these keyservers operated by third parties is that looking up a key by fingerprint or email address reveals personal information and the intent to communicate with the keyserver. While consent might be construed if the key is actually present on the keyserver (especially if only the keyholder can upload it), this is not the case if no key can be found. In that case the entity looking for a key has revealed their interest in communicating with a certain email address without the recipient’s consent.

2.5.5. DNS-Based Automated Retrieval

One way to avoid the information leak when looking up keys by email address is to request them directly from the entity handling the user email. This means extracting the domain from the email address and then querying its servers or Domain Name System (DNS) entries directly for public keys. This ensures that the email address and the intent to communicate with it is only made available to the entity, that will later receive the encrypted email anyways, providing it with no meaningful additional information.

One can also reasonably assume, that the email holder trusts their email provider with their personal information, meaning that requesting a key by email address when verifying signatures should also be fine.

Asking the owner of the email domain for the user’s key provides a level of assurance, that this key actually belongs to the email address. The mail provider could respond dishonestly; however, they ultimately decide who owns a particular email address and could also forge any other email verification method. Additionally, the federated nature of DNS provides resilience against outages and attacks that more centralized systems cannot provide easily, while also providing authoritative information directly from the domain owner.

The main problems with these resource records is that they require a lot of control over DNS records for the domain, which typical DNS hosters might not provide for smaller organizations. They also provide no confidentiality for lookups as DNS requests are usually sent in the clear or at most encrypted between the client and its resolver. This exposes interest in keys associated with the email address[1] to the client's network (any involved DNS resolvers and often passive network observers), leaking communication patterns. The impact of such a leak should be low in most cases but is avoidable using technologies such as the WKD protocol described in subsection 2.5.6.

There are three different methods for publishing information about OpenPGP certificates in the DNS system ("Public Key Association" (PKA) [Koc06], CERT Resource Records [RFC4398] and OPENPGPKEY [RFC7929]). Out of these PKA is deprecated and CERT records don't seem to have been used widely.

OPENPGPKEY records are specified as part of the DNS-Based Authentication of Named Entities (DANE) system for verifying asymmetric cryptographic keys via DNS. It relies on Domain Name System Security Extensions (DNSSEC) for security and consists of DNS records using the special `OPENPGPKEY` type for the name `<hashed-localpart>._openpgpkey.<domain>` containing the public key [RFC7929]. Clients can find keys by performing a suitable query and verifying the DNSSEC signature.

This system requires detailed control over the DNS records for the domain, which many DNS providers don't provide easily and DNSSEC, which is similarly unavailable with many DNS providers. Frequent changes, as expected for larger domains, would either require an API to update DNS entries at the provider or the operation of a name server just to serve OpenPGP keys. Additionally, DNS lookups provide no confidentiality thereby exposing email addresses to the client's network.

2.5.6. Web Key Discovery/Directory (WKD)

The "Web Key Directory" specification [Koc21] describes a way for the domain owner to publish OpenPGP certificates for email addresses under the domain on a standard web server.

The WKD standard specifies two Uniform Resource Identifier (URI)s, where OpenPGP keys can be published. The "advanced" scheme uses a URI such as `https://openpgpkey.<domain>/.well-known/openpgpkey/<domain>/hu/<hashed-localpart>?l=<localpart>` [Koc21] and allows the domain owner to point the `openpgpkey` subdomain to a host

1 Some DNS-based systems hash the localparts, though guessing them based on the observed hashes will usually be easy due to their low entropy and the speed of the hash functions involved. The localpart can be combined with the domain to form the full email address.

that differs from the primary web server. The "direct" scheme on the other hand requires no DNS changes and expects certificates to be published under the URI `https://<domain>/.well-known/openpgpkey/hu/<hashed-localpart>?l=<localpart>` [Koc21].

A client attempting to locate a key for a specific email address will first attempt to fetch it using the "advanced" URI before falling back to the "basic" one [Koc21] and finally concluding that no key is available using WKD.

The advantages of this system are that it is as secure as TLS encrypted connections to the domain even if the domain is not signed by DNSSEC [RFC4033] or the client is not configured to check these signatures. An attacker would have to convince an X.509 Certificate Authority to issue a TLS certificate to them to serve invalid keys. Additionally, and as opposed to the DNS based methods, it provides confidentiality for the localparts used in lookups. Not having to use specific DNS records also makes it useful to smaller organizations that might not have full control over such records or whose DNS provider does not support them.

WKD is already supported by some mail providers and email clients [BSI21]. The modern keyserver "Hagrid" mentioned above also provides a hosted WKD service [Seq21g].

2.6. S/MIME (and X.509 Certificates)

Secure/Multipurpose Internet Mail Extensions (S/MIME) [RFC8551] is the other important encryption and signature system for email.

It relies on X.509 [RFC5280; Wik21l] certificates, which can be used for a number of additional purposes such as document signing and are most known for their use in TLS. TLS uses X.509 certificates to authenticate the server, however they can also be used to authenticate the client in a system called "mutual TLS" [RFC8446].

2.6.1. Certificate Composition

X.509 certificates consist of exactly one asymmetric keypair, attributes describing the keyholder and a signature from exactly one CA. In addition, all intermediary CAs needed to verify the signature chain to a root CA (see subsection 2.6.2) are typically included.

2.6.2. Trust and Verification

X.509 certificates rely on a hierarchical trust model where the operating system or application includes a set of trusted root CA certificates. Users or their administrators can add additional CAs to the trust stores distributed with the operating system or application and use them to sign internal certificates. These CAs then certify intermediary CAs, which issue certifications for user certificates.

The X.509 trust model puts ultimate trust in all root and intermediary CAs as each CA can sign any key. Options to scope the trust in a CA similar to the ones used in

OpenPGP do not exist. This means that a compromise of any CA will compromise the security of the entire X.509 trust system until the CA can be revoked. Revoking a CA also causes all certificates that use it to fail verification, making revocations of important CAs expensive and potentially infeasible.

2.6.3. Revocation

Each CA provides a centralized list of revoked certificates signed by the CA that clients can fetch. There is also a mechanism called Online Certificate Status Protocol (OCSP) [RFC6960], by which clients can ask the CA for the revocation status of a particular certificate.

2.6.4. Signatures

Signatures made using X.509 certificates include the user's certificate and the trust-chain to the root CA. This allows them to be verified without contacting external parties (apart from revocation checks).

2.6.5. Key Distribution/Exchange

Keys for email encryption using S/MIME are exchanged by sending a signed message first. The client extracts, verifies and stores the certificate contained in the signatures and allows it to be used for future encrypted communication.

Within organizations, keys can also be retrieved from Lightweight Directory Access Protocol (LDAP) directories [RFC4532].

2.6.6. Summary

The ultimate trust and centralized revocation simplify the verification process drastically compared to OpenPGP but require lots of trust to be placed in CAs.

3. Existing Solutions

There are a number of existing solutions for cryptographic protection of emails. This chapter gives a short overview of those and describes some relevant strengths and weaknesses.

3.1. S/MIME Using Public CAs

One of the two major systems for email encryption is S/MIME. It involves certificates issued to exactly one name and email address and signed by exactly one CA.

To obtain such a certificate a user must choose and pay one of the approx. 100 CAs [Wik21f; Moz20a; Mic20] that are widely trusted by mail clients, send them a certification request containing their public key, prove that they have access to the email address and verify their name by sending in scanned national identity cards and possibly signed documents. Higher security levels might require the user to physically meet the CA or prove their identity in a video chat.

For personal use some CAs offer to issue certificates containing only the email address free of charge.

The big advantage of this approach is that all email clients supporting S/MIME will immediately trust the certificate and mark the email as signed without requiring configuration on the user or even administrator level.

The main disadvantages are the relatively high cost per e-mail address (between 20 USD per year for email validation to about 90 USD per year when including names and organization information [Sec20; Sec21]) and the cumbersome and manual process of verifying each user's identity with an external party. Another problem is that the organization needs to fully trust all public CAs to verify the identity of their own members.

3.2. S/MIME Using Internal CAs

To reduce costs, maintain control over identity verification and automate parts of the issuing process, some larger organizations deploy their own internal CAs that are trusted across the organization through existing IT automation systems. Those are used for internal purposes ranging from client and server authentication to encrypted email.

Given proper tooling these systems can be used very effectively and conveniently to encrypt email within the organization. However, problems arise when signed or encrypted mails pass the boundaries between organizations. External users wanting to verify the certificates must either verify each individual certificate or trust the organization's CA, which means trusting it for everything. X.509 certificates don't currently support scoping trust to certain domains, which means a user trusting an organization's CA must also trust the organization to not impersonate their bank or any other organization.

This makes these certificates very useful within an organization but effectively useless outside of it.

3.3. Unmanaged OpenPGP

The other major system for email encryption is OpenPGP, which is designed around the concepts of decentralization and a web of trust. In practice however, the verification system in OpenPGP consists of manually verifying the keys of almost all communication partners. Trusted signatures are rarely available for verification. This process of identity verification for every communication partner scales very badly and will quickly become cumbersome as more members join a group. In theory every new member would have to contact all existing members and mutually verify their keys. In practice users often try to obtain the key from a somewhat trusted source or simply accept the first key they see for a user.

This is presumably the most common use of OpenPGP in organizations. Some users will generate and exchange their own keys without oversight or central management and verification is likely sporadic at best.

Some organizations collect their member's keys in a central, trusted list, which alleviates some problems [McC18].

3.4. Encryption Gateways

Multiple companies offer so-called "Encryption Gateways" [Tel18; Tot21; Cip20; Zer21] that work as part of the mail server similar to virus scanners and spam filters. This has the advantage of allowing other mail filters to be applied to unencrypted mails and not requiring special software or configuration on the clients. However, this also places all encryption keys and unencrypted messages on the mail server where they are vulnerable to attack by the mail server operator or in case of a server compromise. When all parties in a mail exchange use such gateways, there is little to no increased security compared to simply enforcing TLS between the mail servers. Since signing and encryption usually happen together, this also places signature keys outside the direct control of their official holders.

A common feature with encryption gateways is a fallback mode for recipients that don't have mail encryption set up. In these cases, the gateway can often be configured to store the contents of the email as a web page and send a link to the recipient instead. Usually, a password is required to view the contents, which is typically provided in a separate email to the same address. Unless the sender takes the time to provide this password over a different, secure channel and password changes can only happen this way as well, this provides no additional security over regular email. An attacker that can intercept emails can also follow the contained links to the respective web pages and read or request passwords. Instead, these gateways cause additional challenges with archival storage on the receiving end, where each end user would have to manually download and archive every single mail they receive through this system. Otherwise, the sender would have to keep the encryption gateway online and secure for the longest possible retention period of any receiver (multiple years for business email [Wik21b; Ion19; Jat21], potentially longer in the case of legal holds [Ext21])).

3.5. Encrypting Messengers

Messengers are commonly seen as a replacement for email due to their simplicity and convenience and have largely replaced it for non-business and personal communications between individuals.

In a business setting additional challenges arise such as difficulty of archiving and backing up messages, the requirement for legal holds or (with some messengers) the reliance on smartphones.

Some messengers such as Matrix [Mat21] or Wire [Wir21] are more suited for organizations by using federation and retaining messages on each organization's servers or implementing features such as legal holds.

Ultimately, the main problem with these systems is that all communication partners need to use the same system, which makes them useful for cohesive groups within an organization but much more difficult to use across organizations, where email is an established standard.

3.6. OpenPGP-CA

A rather new solution to this problem is the OpenPGP-CA project [Ope21g].

It uses the concept of trust signatures in OpenPGP whereby a user can trust a certificate authority to make trust decisions on their behalf by issuing a special trust signature on the CAs key. This CA can then in turn trust other CAs by issuing further trust signatures. The crucial aspect of these trust signatures is that they can be scoped to a domain.

This differentiates it from existing CA infrastructures and allows users and CAs to trust an external CA only for email addresses under their domain, thereby alleviating the problems usually associated with running internal CAs while retaining their flexibility and convenience.

It also creates a trust network that requires comparatively little trust in the CAs that create it. Every CA is only trusted by the rest of the network to correctly certify certificates for email addresses belonging to its domain, which the domain owner controls anyways. Members of the CA's organization can choose to trust their organization's CA unconditionally, which allows it to trust other CAs on behalf of the user. This lifts the user's burden of doing so themselves and allows the CA administrator to make trust decisions on behalf of the user.

3.6.1. Certification Process

3.6.1.1. User Generated Keys

To have their key certified by their organization's CA, a user has to generate it locally using existing tooling, manually trust-sign the CA and optionally generate revocation certificates in case they lose the key and need to revoke it. They then send their public key, the trust-signed CA key and optionally their revocation certificates to the CA administrator for certification and publishing via tools such as email.

The CA administrator then has to manually verify that the key was submitted by the claimed keyholder and that all UserIDs on the key are correct and can be certified. The administrator then imports the information provided by the user into the OpenPGP-CA database using the CLI tool, which certifies all UserIDs on the key.

Finally, they have to export the current version of the WKD folder structure and synchronize it with the web server.

This is a secure way to certify the user's key if the administrator correctly and carefully verifies the user's identity and all UserIDs submitted as part of the certification request.

3.6.1.2. CA Generated Keys

Alternatively, the CA administrator can generate keys for their users using the OpenPGP-CA CLI, which handles generation of the key, trust-signature and a large set of revocation certificates before certifying the UserIDs that the administrator specified and outputting the private key and a randomly generated passphrase for it to the terminal.

This leaves the administrator with the task of securely transmitting both the private key and the associated passphrase to the user while ensuring their confidentiality. Securing this private key transfer will be fairly difficult without pre-existing confidential communication channels and introduces a high risk of handling errors leaving private keys in insecure places. The most obvious solution of emailing the key would leave it directly next to the data it encrypts. Encrypting the key and transmitting the passphrase via telephone would mean reducing the security to that of an unencrypted telephone connection. Meeting the user in person might not always be feasible either.

Additionally, this leaves the administrator in a position, where they can retain a copy of the private key which allows them to use it without risk of detection. While this might be desired for encryption keys in some cases to ensure that communication can be decrypted as needed, it poses a large risk for signature keys. Access to the signature keys would allow an administrator to forge the signature of any user in the organization, allowing for plausible repudiation and reducing their usefulness as evidence.

While a CA administrator could also generate and certify a new key to forge a signature, this would leave evidence in the existence of an additional key and the forgery could easily be detected by comparing fingerprints.

3.6.2. Automation

There are multiple ways of automating the functionality of OpenPGP-CA ranging from automated interaction with the CLI to using the Rust library or the provided REST daemon. These options allow an administrator to automate parts of the certification process, however none of them provide assistance in authenticating users and authorizing their use of certain UserIDs, which is the most error-prone and crucial part of the CA's operation and necessary for every single certificate.

In organizations with lots of users or high fluctuation, this task will become quite tedious and risks the administrator becoming careless with the verification process.

3.6.3. Bridging Trust to Other Organizations

The CA administrator as well as individual users can issue a trust signature over a different organization's CA and scoped to their domain to trust them with certification of their members. If a CA trust-signs another organization, this trust is inherited by everyone that issued an unconditional trust signature over the CA. The process of trust-signing a CA is called "bridging" [Ope21b] and the CLI facilitates it.

This creates a system where few, carefully made trust decisions span a large network of verified certificates.

For the client to be able to verify such an external certificate, the full chain of certificates and trust signatures must be available to the client as clients will not request them during verification. This can be achieved by distributing them using existing client automation solutions or using software that fetches certificates from published keylists [MLW19; Mic21; g1020].

3.6.4. Interacting With the Rest of the World

Interactions with bridged organizations that also publish their keys using the WKD are simple as clients can automatically retrieve their key and will accept it as trusted without user intervention. When interacting with users from an organization that also uses OpenPGP-CA, but which has not been bridged yet, users can manually issue a scoped trust signature over the organization's CA to yield the same effect for themselves.

In all other cases, users still need to manually verify certificates.

3.6.5. Privacy

The WKD system suggested for publication of keys only allows keys to be located by email address, preventing key enumeration and lookups based on fingerprints.

However, if trust signatures of the CA key are provided to the CA, they will be included as part of the CA key and published along with it. This leaks the fingerprints of all keys (presumably organization members) that have trust-signed the CA and provided the signature to it. If such keys can be found by fingerprint elsewhere, this allows the enumeration of organization members.

Furthermore, the CLI offers no option to delete a key except manually manipulating the database file. The REST daemon allows a key to be marked as delisted, which excludes it from publication but retains it in the database.

3.7. Summary

Each of the solutions mentioned above has their own benefits and drawbacks and can be suitable for specific use-cases. This thesis aims to provide a new solution that combines some of these benefits in the form of a combined keyserver and certification authority suitable for use in organizations.

Requirements for such a solution are laid out in the next chapter.

4. Requirements

The following is a list of requirements that the keyserver described in this document should fulfill. They are provided in no particular order but rather grouped by topic.

4.1. Ease of Use

Interacting with the keyserver should be reasonably easy for the user and minimal OpenPGP specific knowledge should be required. Users should be able to follow organization specific instructions to generate and configure the keyserver as required.

4.1.1. Tool for Key Generation

The keyserver should offer a tool to generate keys and associated data as needed by the keyserver on the user's computer and under their control. This tool should require minimal OpenPGP specific knowledge to allow a broad range of users to generate keys themselves. Choices such as key size and subkey structure should be taken by the administrator to create uniform keys that meet the organizations regulatory and compliance requirements.

The generation tool should also generate trust-signatures over the configured CAs and offer the signed CA keys for download during the key generation process.

Keys generated using this tool should automatically be submitted to the keyserver for storage, certification and publishing. The private key must remain on the client and will be provided to the user to ensure the confidentiality of messages encrypted to and the non-repudiability of signatures issued by the certificate.

4.1.2. Interoperability with Typical Clients

The keyserver must make use of standard OpenPGP features to provide compatibility with existing clients where possible. Protocol extensions must not be required, and typical clients should support the keyserver as well as possible. This ensures that users can make use of the multitude of OpenPGP compatible clients that are available for a number of platforms and ensures that no special client software needs to be developed and maintained to use the server. It also ensures compatibility with the software used by external communication partners.

4.1.3. Delegating Trust Decisions

Users should be able to delegate trust decisions to the keyserver for cases, where they can be taken automatically based on the keyserver's configuration.

4.2. Locating and Verifying Keys of External Users

The keyserver developed in this thesis must offer a lookup interface compatible with the existing keyserver protocols supported by clients that returns not only internal but also external certificates retrieved from configured keyservers.

This ensures that clients can be configured to perform lookups through the keyserver, which can in turn be configured to respect privacy regulations by only querying certificates from sources associated with the requested email's domain rather than public keyservers operated by third parties.

It provides interoperability with other organizations operating WKD directories or similar keyservers for their members and limited compatibility with existing public keyservers.

4.2.1. Per Domain Configuration

The keyserver administrator should be able to configure how keys are retrieved on a per-domain basis to reflect the relevant domain's certification and publishing policies for OpenPGP certificates. A fallback configuration should be applied in case no special configuration is available for the domain.

This granular configuration allows the administrator to control how keys are retrieved and which keys will be accepted.

4.2.2. Web Key Discovery

The keyserver must be able to locate keys published according to the WKD standard [Koc21; BSI21]. This standard allows users to easily publish their OpenPGP certificates on their own domain, where they can be found by supporting clients and used for secure communication. It also provides a level of assurance, that the key was published with the email owner's consent and is controlled by them.

4.2.3. Keyservers

Regular keyservers should be supported for domains to facilitate fetching certificates from organizations that operate their own keyservers for users of their domain. While the WKD approach is preferred where possible due to its simplicity and assurances on email address control, existing organizations might already be publishing keys to a keyserver rather than the WKD. If the keyserver requires authentication or ideally even authorization before publishing keys, it can also provide assurance that the keys are authentic.

4.2.4. Verifying Certifications

The keyserver must be able to verify that UserIDs for a specific domain be certified by the domain's CA and filter other UserIDs from the certificate. This can be used to prevent forged certificates from being forwarded to users if it is known, that all genuine UserIDs using the domain are certified by the relevant CA. It can also be used to ensure that all certificates arriving on client systems for this domain are certified non-repudiably as belonging to the organization by their CA, assisting signature verification.

4.2.5. Assigning Validity Based on Retrieval Method

The keyserver should be able to accept a certificate as verified based on the fact that it has been retrieved from a keyserver or WKD host that has been configured for the domain. This, combined with the re-certification described below, increases the pool of verified certificates to all certificates for which the administrator has configured a trusted source even if that source does not offer usable certifications.

4.2.6. Recertification / Automating Trust

Such an accepted certificate should be certified on the fly by the keyserver using a configured private key. This would allow the client to automatically accept the key as verified by the keyserver without user intervention, taking the burden of verifying certificates away from the user and allowing such trust decisions to be taken automatically by the keyserver as configured by the administrator.

4.2.7. Network of Verified Certificates

Combined these options should allow the keyserver administrator to configure parameters for partner organizations, trusting them to certify their own users and thereby spanning a large network of trusted certificate authorities with relatively few trust decisions. Users should be able to join this network by having their certificate certified by their organization's CA and issuing an unlimited trust signature over the CA.

4.3. Authentication and Authorization

The keyserver must only store, publish and certify UserIDs if they consist of an email address and optionally a name and the keyholder has proven to be authorized to use both in the certificate. Additionally, the keyserver must require that the keyholder be a member of the organization operating the keyserver to reduce complications with privacy regulations and prevent others from abusing the keyserver.

4.3.1. Name Verification

Almost all organizations use Single-Sign-On (SSO) systems, that can be used to authenticate users and provide basic attributes about them. The keyserver must require that users confirm the name in their UserIDs by logging in using the organization's SSO system. The keyserver will accept a name if it matches the full name provided by the SSO system. Multiple SSO systems should be supported through an open protocol.

Information about mail addresses may also be verified, if the SSO system provides appropriate data.

This system ensures that only the legitimate holder of the given name can add it to a certificate stored on the keyserver and prevents attackers from impersonating users.

Verifying the name in addition to the email address on the UserID is especially important in cases where a manipulated name might not be obvious due to a mismatch with the email address. For example, if email addresses in the organization only include the user's initials and a random component, an inside attacker with the same initials as their victim could have the CA certify a key with the victim's name and their own email address to impersonate the victim. Requiring any name included in the certificate to be verified prevents such attacks.

The requirement to authenticate with a SSO account also proves that the keyholder is a member of the organization operating the keyserver, preventing external users or attackers from publishing certificates.

The advantage of using SSO instead of a custom approval system is that it requires no additional work on the part of the administrator and is very convenient for the user. In some cases, it might even be possible to automate the certificate issuance process, if appropriate client tooling is available and the client can obtain relevant tokens on their own.

4.3.2. Email Verification

Authorization to use an email address in UserIDs on a key can be verified by the keyholder proving their ability to read email sent to the address. This ensures that users can generate keys for any email address (in particular aliases), that they control in practice rather than just the email addresses known to the SSO system.

4.3.3. Certification

Once at least one UserID on a key has been authorized as described above, the approved UserIDs must be certified and the key with these approved and certified UserIDs must be published to the Web Key Directory. This certification can be relied upon by internal and external users and clients to verify keys belonging to members of the organization without contacting the key holder or relying on the Web of Trust. It also allows verification of the CAs assertions when the CA is not directly reachable for example when the client or keyserver are offline and forms a non-repudiable statement by the CA that it believes the mapping between UserID and key to be correct.

4.4. Interoperability With the OpenPGP Ecosystem

The keyserver must interoperate with the existing OpenPGP ecosystem, supporting existing clients and following protocols and standards.

4.4.1. Web Key Directory

The keyserver must make keys available through the WKD protocol, so that clients that support it can automatically retrieve keys without manual configuration. WKD is supported by many clients and in some cases (such as with GPG) configured as part of the default key lookup strategy for new installations. This ensures, that users of supported clients can easily obtain the organization's certificates without special configuration and in particular without special client software.

Such a WKD setup is also easy to implement and operate without requiring complex DNS setups or configuration, which might not be available to smaller organizations and might be difficult to implement and authorize in larger ones.

4.4.2. Compatibility With OpenPGP-CA

The keyserver must be compatible with the OpenPGP-CA project and other instances of OpenPGP-CA to reduce and avoid further fragmentation in the OpenPGP ecosystem. The intent of this project is not to replace OpenPGP-CA, which is appropriate for smaller organizations but to augment it with a solution that scales to larger organizations and requires no manual intervention for each user.

4.4.3. Trust From External Users

External users must be able to verify the CA's assertions using regular client software and trust the CA, if their client supports the necessary features.

4.4.4. Foreign Certifications

The keyserver must be able to accept and publish certifications from external users, if approved by the user (see subsection 4.5.4). This allows keyserver users to have their certificate certified by external CAs operated by different organizations or important individuals as desired and to redistribute these certifications together with their key.

The absence of such a feature might tempt users to upload their key to public keyservers in addition to the organization's keyserver, which would reduce their privacy.

4.5. Privacy

The keyserver should protect the privacy of its users and external communication partners, whose keys are handled by the server. It should limit the spread of personal information where reasonably possible.

4.5.1. Preventing Enumeration

The keyserver should take reasonable precautions to make enumeration of keys and other personal information on the keyserver difficult, protecting sensitive information such as the list of organization members and making spamming more difficult.

4.5.2. Hiding the Social Graph

The keyserver should only redistribute third-party certifications and other indications of communication from a third party to the keyholder with the keyholder's or administrator's approval. The former allows the user to choose which certifications they want to publish, while the latter simplifies the use of public certificate authorities.

This ensures that third parties cannot abuse the keyserver to distribute undesired information about the key holder.

4.5.3. Stopping Malicious Key Packets

The keyserver should strip malicious data such as invalid signatures or other packets that could be used to harm clients, reduce the usefulness of the key or even spread problematic data along with the key, causing legal issues for the keyserver operator and/or keyholder.

Packets generated by the keyholder (such as revocations or new subkeys) should be accepted if they can be cryptographically authenticated to be authorized by the keyholder to ensure the distribution of revocation information and for ease of use.

4.5.4. Publishing Personal Data Only With Approval From the Subject

Personal data as present in UserIDs should only be published if all parts of it are approved by the subject they describe as required in section 4.3. This ensures that only UserIDs approved by the persons described by them can be published. Combined with only publishing cryptographically authenticated packets of the key and requiring keyholder confirmation before publication, this ensures that no personally identifiable data is published without the approval from the subject that is described by it.

4.5.5. Deleting Personal Data

The keyserver should offer users to delete any personal data associated with a key or an email address if they can prove access to it. This implements the “right to be forgotten” [Wik20] and similar regulations about the deletion of personal data.

Data that has not been approved by the keyholder will be deleted after a reasonable amount of time.

5. Proposed Solution

This chapter describes the solution proposed and later implemented as part of this thesis. It does so by providing a high-level overview of a key's life cycle when using the proposed keyserver PortuLock from the user's perspective. Further technical details will be provided in chapter 6.

5.1. Key Generation

A key's life cycle starts with the generation of its primary key material followed by attaching its UserIDs, generating subkeys and then binding everything together with signatures. Most clients can do this, providing relatively simple wizards for basic functions or more complex interfaces for advanced options.

Usually, these clients will apply some defaults regarding key size, algorithm and preferred algorithms for hashing and encryption which should be fine for most cases and is very appropriate for individuals. Organizations however, especially ones that have to comply with regulations and standards might want to centrally define these parameters to ensure compliance and a unified appearance.

Once the key has been generated it has to be distributed to potential communication partners, in this case by submitting it to this project's keyserver, which will then verify and publish the key.

5.1.1. Web Generator

This project provides a simple web interface that generates a key in the user's browser according to the administrator's configuration and handles the more complicated processes behind the scenes. While a native client could provide more security, stronger isolation from the browser and direct access to smart cards or local software such as mail clients, implementation as a web client was chosen to simplify cross-platform use and avoid having to install or download and execute software for key generation.

The client will prompt the user for any UserIDs, they want included on the certificate before generating the certificate and a revocation signature. The client will use the newly generated key to issue a trust signature over one or more CAs configured by the administrator, optionally scoped to a domain. It then asks the user to download all of the above before allowing them to submit the key to the keyserver, which will in turn start sending verification emails to verify the user's names and email addresses.

The client will also use the private key to generate a link to the authenticated status page for the key, where the user can monitor the verification and publishing progress and manage their key. This page will allow the user to view and download all data stored for the relevant key as well as request deletion, if allowed by the administrator.

5.1.2. Status Page

The server provides a simple status page showing any data associated with the fingerprint of a given key. This includes published and pending key packets and data parsed from them as well as verified names and email addresses that are approved for use on the key. It allows the user to download all of these packets in a combined public key file as well as view and download any stored revocation certificates. If desired all data associated with the key's fingerprint can be deleted from this page as well.

Access to the status page is authenticated using a management token that can be obtained either by having it sent to one of the email addresses on the key or by requesting it in encrypted form and decrypting it using the private key. A form in the web interface allows the user to request links to the status page including suitable management tokens for any key associated with a particular email address, while the decryption challenge can be used by the web generator and other automated tools to prove access to the key.

5.1.3. Trust Signatures

Any OpenPGP key can issue a trust certification over any other key, which can either be scoped to an email domain or unlimited. To configure a client to trust a particular CA, trust certifications have to be issued over the CA key. Typically, these would be scoped to the domains for which the CA is authoritative, however the user can also choose to issue an unlimited trust certification over a CA, which causes their client to follow all (presumably scoped) trust certifications of the CA recursively. This allows the CA operator to issue scoped trust certifications over CAs of other organizations that are frequently communicated with, causing everyone with such an unlimited trust in the CA to also trust the other organization's CA.

The web client can be configured to generate such scoped and unscoped trust signatures for any number of CAs when generating a new key. It will then offer these trust-signed CA certificates for download, which when imported together with the new private key, will cause the client to automatically trust the CAs accordingly.

5.1.4. Manual Key Generation

The above-mentioned steps can all be performed manually given good knowledge of OpenPGP and a client that supports them and makes them available to the user. This allows the user to skip some steps, gives them more control over key generation and does not expose the private key to a browser environment. Further, this allows keys to be generated directly on smart cards or tokens in environments where this is required or desired. Keys generated like this can then be submitted to the system using `gpg --keyserver <KEYSERVER> --send-keys <KEYID>`. The system will then email the user with further instructions to verify their email address and name before certifying and publishing the key.

5.2. Authenticating Users and Authorizing UserIDs

UserIDs are UTF-8 strings, that should follow the format `Name <localpart@domain>`. This keyserver assumes this format and requires the name (if present) to be verified using a SSO system. An email address with one of the organization's domains must be present and verified to prove membership in the organization and for publishing in the WKD. Comments are not accepted as they cannot be verified automatically.

Keys can contain any number of UserIDs, that will be treated independently. Each must be verified before being published. Any number of names and email addresses can be authorized for any number of certificates. Multiple keys can exist and be published for one UserID or email address, which can be important to verify old signatures. However, only one of these keys should be valid at any given time to avoid confusion about which key to use[Koc21].

5.2.1. Name Verification

To verify that the name in a UserID actually belongs to the keyholder, the server relies on an existing SSO system in the organization.

The user will be redirected to the organization's SSO system, which will verify the user's identity using their existing credentials before sending the user back to the keyserver where they can confirm the connection between their name and their key's fingerprint. If an email address is provided as part of the SSO systems response, this is also accepted as verified and stored.

Name verification is triggered automatically by sending an email to the keyholder when they upload a UserID, whose name has not been verified yet.

5.2.2. Email Verification

Authorization to use an email address is verified by sending an email with a confirmation link to the email address. After clicking the link, the user is asked to confirm that the fingerprint on the page matches the one on their key to avoid them accidentally confirming malicious keys after which the mapping is stored in the database as well.

Verification emails are automatically triggered when a new UserID is uploaded and the relevant email address is not already verified.

5.3. Certifying and Publishing Keys

Once a UserID on a key gets verified completely and the keyserver authorizes it for certification and publishing, the keyserver passes the authorized packets to an instance of OpenPGP-CA responsible for the relevant domain, which will certify the UserID and store the key in its internal database.

At this point any client that is configured to trust certifications issued by the CA for the relevant email domain and is presented with this certification will automatically do so, validating signatures and using the key for encryption without prompting the user.

The keyserver then exports the certificate into a WKD folder, where it is picked up by a bundled web server and published according to the WKD standard, allowing any WKD clients to obtain the key.

5.4. Finding and Verifying Keys of Organization Members

Keys for organization members can be retrieved using the WKD protocol, which is supported by most modern clients [BSI21], if their email address is known.

For GPG the relevant command is `gpg --locate-external-key <email>` with the additional option `--auto-key-locate clear,nodefault,wkd` ensuring that only the WKD is used. Most clients provide a graphical way of doing the same thing.

At this point organization members will already trust the new certificate because they have imported and trusted the CA during key generation. External users can manually obtain the CA using a WKD lookup for the email address `openpgp-ca@<domain>` or from other trusted sources before trust-signing it with the scope limited to the organization's domain. Once the trust certification has been issued, their client will also trust all the organization's certificates.

If their client does not support trust signatures, they can also accept the key manually based on the fact that they have received it directly from the organizations keyserver using the WKD protocol or after manually checking the CA's certification.

Finally, all certificates should be updated regularly using the WKD protocol to receive new certifications and potential revocations.

5.5. Finding and Verifying Keys for External Communication Partners

Of course, internal users will also need to securely find and verify the keys for external communication partners.

If the desired key is available through the WKD, keys can be obtained directly by the client. This leaks no information to third-parties and provides some assurance, that the key does indeed belong to the email address.

If the external communication partner is also using an instance of PortuLock or OpenPGP-CA, that has been trust-signed ("bridged to") by the local CA and whose up-to-date CA certificate is available, the client will automatically trust the new certificate.

Otherwise, the certificate will be fetched but might not be accepted as verified without manual action by the user depending on the client.

Keys stored on an organization's own keyserver cannot be requested by current clients directly as they do not support different keyservers per requested domain. All queries are sent to all configured keyserver, which would leak the email address to other organizations.

Similarly, public keyservers cannot be queried at all without revealing the email address to a third party.

Moreover, for domains that are known to certify all associated keys with a CA, clients cannot enforce this and require such certifications. Instead, they will use their normal verification system, which can cause fake keys to be accepted as valid despite the existence of an authoritative CA for the domain.

5.5.1. PortuLock's Aggregation Feature

To control key resolution, support organizations operating their own keyservers and make clients accept keys as valid based on how they were retrieved, PortuLock implements a keyserver proxy.

Clients can be configured to use their organization's keyserver in addition to or likely instead of other lookup methods, which will cause all key lookups to be performed via the keyserver.

The keyserver will then extract the domain from the requested email address and check its configuration for matching entries. An entry for a domain can specify a set of keyservers to query for certificate data and whether to perform a WKD lookup.

Key data retrieved from these sources can then be filtered to only include UserIDs, that have been certified by a configured CA. This provides an additional layer of protection over the secure connection and ensures that the target organization has non-repudiably certified the certificate and its organization membership.

Afterwards, the remaining UserIDs can be certified on the fly with a configured certification key. This ensures that clients trusting this key will accept the certificate as verified, following the keyservers decision. It is also useful in cases where the target organization does not certify keys, but they are known to be authentic because they came from the organizations keyserver.

If no configuration entry can be found for the domain, a fallback configuration will be applied instead. It provides the same options as described above. In most configurations, this would be set up to perform a WKD lookup and then potentially certify the key based on it. It could also be configured to query public keyservers, however this would leak the email address as described above.

This system allows the keyserver to take trust decisions as configured by the administrator for the user, simplifying the process for the users and providing them with a large pool of verified certificates.

Manual verification and key retrieval might still be necessary in some situations however these should be less frequent allowing more care to be taken.

5.5.2. Undiscovered Keys

Unfortunately, this system cannot find all keys that might exist. In particular keys on public keyservers or keys that are exchanged manually cannot be found. In the former case this is not possible without leaking the email address and in the latter case automation too difficult.

In these cases, it is left up to the user to ask their communication partner for their key and verify it manually.

If their email providers support WKD, or they have control over the domain's web server, external users can publish their certificates to the WKD to solve the problem.

As the adoption of the WKD and keyservers like PortuLock grows, this should become less common. In the meantime, the server could be extended to provide an interface, where external users can manually upload their key for use by the organization and have their email address verified. This pool of keys could then be included in lookups performed by the aggregation feature.

One complication of this feature would be the need to authenticate key lookups as they would include unpublished data. Further complicating matters would be the lack of authentication support in the keyserver protocol. Currently, this authentication isn't required because the keyserver only queries public sources and can leak at most the aggregation configuration.

5.5.3. Finding Keys by Fingerprint

Unfortunately, a federated system where keys are stored on separate keyservers for each domain does not allow lookups based on KeyIDs or fingerprints without asking all keyservers, which would leak personally identifiable data and the intent to communicate with that person to all of them.

Clients that use fingerprints or KeyIDs to refresh keys or clients using the same information to fetch the keys needed for signature verification will therefore fail to find any keys. This could be solved by including the signers email address in each signature and having clients attempt to locate keys not just using their fingerprint but also any known email addresses.

PortuLock's aggregation feature can also be configured to lookup keys by fingerprint on a subset of the configured keyservers, however this should be limited as it leaks the fingerprint and interest in it to all this subset of keyservers.

5.6. Privacy

Privacy with regard to OpenPGP keys is a trade-off between making it possible to locate keys and not revealing more information than necessary. At a minimum the keyserver needs to provide keys when they are requested by email address. Only UserIDs matching the requested email address need to be returned, hiding any others. This allows legitimate users to find encryption keys for communication and (with caveats described in subsection 5.5.3) finding keys needed to validate signatures. It also allows an attacker to verify that a specific email address exists and reveals the keyholder's name. While this cannot be avoided, it also provides little information that the attacker could not obtain otherwise as mail servers typically indicate their existence by refusing delivery to non-existent addresses as well [RFC5321]. Furthermore, email addresses are usually derived from names, which means that names can usually be guessed from them as well.

Using one key for multiple email addresses could reveal their connection if the attacker can guess them all, however this can be prevented by simply generating independent keys.

5.6.1. Hiding the Social Graph

Traditional OpenPGP keyservers accept any packet related to a key, store it indefinitely and return all of them when a key is requested. These packets include certification signatures from other users, which indicate that they have verified that the key belongs to its indicated key holder. However, they also suggest, that the certifying party has been in contact with the key holder, which can be used to build a social graph of keys and therefore users that have presumable been in contact with one another.

This server only accepts signatures issued by certificates whitelisted on an instance level or certification signatures that have been attested by the key holder through an attestation signature. The former is intended for commonly used CAs, while the latter is intended for personal certifications where they are desired.

This limits the number of certification signatures that can be used to build a social graph and allows the keyholder to choose which certifications and therefore relationships with others they want to publish. Effectively this builds a system, where both users must agree to have their relationship published. Of course, this does not prevent unattested certifications from being published by other means outside the organization's control.

5.6.2. Limiting Lookups and Preventing Enumeration

The keyserver contains a list of all OpenPGP keys in the organization and presumably covering a large portion or ideally all valid email addresses, information that would be valuable to potential attackers.

To protect this information, while still allowing encrypted communication the keyserver only responds to exact matches of the email address (as defined by the WKD specification and hashing algorithm) and provides no interface to search by any other component of the key.

When a key is requested by email address, only the UserIDs matching this email address are provided and others are withheld. This avoids an attacker learning about different roles that the keyholder might hold and that are associated with different email addresses.

To ensure that attackers cannot circumvent this protection by querying keys using fingerprints or KeyIDs, such queries are not allowed. Lookups of the entire key by fingerprint are only possible using the status page (see subsection 5.1.2), which is only available to the authenticated keyholder.

If necessary, additional protection against attackers guessing valid email addresses and confirming their guess using the keyserver could be provided by rate-limiting requests.

5.7. Revoking a Key

OpenPGP keys are revoked by adding a signed revocation statement to them. This keyserver offers to store such revocation certificates for later retrieval by the administrator (and by extension the user) should the private key not be available. The web client for key generation automatically generates and stores revocation certificates for the user. Future versions of the keyserver could require revocation certificates to be escrowed to the server before publishing keys to ensure their availability to the administrator, should revocation become necessary.

OpenPGP also supports the concept of designated revokers. This allows the keyholder to designate a different public key to revoke their key should it become necessary. This would make the process more elegant as it allows revocation reasons and times to be specified later and avoids having to store multiple certificates. Instead, users could (be required to) designate the CA key as a designated revoker, allowing the CA to revoke the key when needed. Unfortunately, this feature does not seem to be supported by enough clients to replace traditional revocations securely.

5.8. Deleting Personal Data

OpenPGP keys are designed as an append-only set of packets that can be distributed in full or in part in a number of ways. When clients update keys from keyservers, they will obtain all packets from the keyserver and add them to their local set. When a user chooses to send their local version of a key to a keyserver (with or without consent from the keyholder), their client sends all locally known packets which the keyserver will store indefinitely. The idea behind the append-only approach was to prevent attackers from removing keys and important parts of them such as revocation signatures. Removing a key could cause mails to be sent unencrypted and omitting revocation information could give an attacker more time to use compromised keys.

With traditional keyservers such as the SKS keyserver network, this means that users cannot delete their key from the servers should they wish to do so. More modern servers such as keys.openpgp.org allow the user to delete their UserIDs but retain everything else to allow revocation information to be distributed freely. They claim that this anonymizes the key by removing names and email addresses making it ok to retain it despite General Data Protection Regulation (GDPR) regulations on deleting data upon request. However, unique personal identifiers such as KeyIDs and fingerprints remain, and the key can be linked back to the individual given the right information (such as a copy of the removed UserID and its signature retained elsewhere).

5.8.1. For Keyserver Users

This server only publishes keys for which at least one email address has been validated and allows the administrator (or the keyholder depending on configuration) to delete the entire key. For existing keys only packets authorized by the keyholder or administrator are accepted and published. These packets range from signed UserIDs to revocation certificates. In particular only authorized certifications are accepted (see subsection 6.3.7).

This means that the keyholder controls which information the keyserver can publish, with the exception that once they accept or create a certain packet, they cannot prevent others from adding it to a keyserver, where the rest of the key already exists.

Keys can be deleted by the user using the status page described above, however this can complicate verification of existing signatures, prevent distribution of revocation information and risks emails being sent unencrypted despite a key existing. For this reason, the feature can be disabled administratively at which point the user will simply be asked to contact the administrator for deletion.

5.8.2. For External Contacts

External contacts usually have a right to have data about them deleted. PortuLock solves this by not centrally storing any data about external users. In particular the aggregator server does not retain or even cache key data it retrieves but requests it again for each request.

Clients typically retain any key they have ever seen. They would have to be set up to regularly update keys and delete ones, they haven't been able to find for a while (see chapter 9) to ensure that they don't retain data that is no longer publicly available.

6. Implementation

6.1. Web Client

PortuLock uses a web client for key generation and some less important operations such as requesting a link to the authenticated management and status page or looking up keys manually using the aggregation feature.

This web client is a JavaScript application using the Vue framework [You21] for visuals and interacting with the keyserver using a REST API.

Implementation as a web application was chosen primarily to avoid having to install software on the user's computer for the one-off key generation process.

Appendix C shows screenshots of the key generation process.

6.1.1. Configuration Management

When the Web Client gets loaded in the user's browser it requests a configuration file from the domain it is running on and parses it to configure itself.

This allows the configuration to be changed at runtime without having to recompile the software and also allows different configurations to be used depending on where the web client is loaded from. This can be useful in cases where multiple organizations share a PortuLock instance but want individual configuration.

See subsection 6.6.4 for details on what can be configured using this system.

6.1.2. OpenPGP Library

The Web Client uses the openpgp.js library for key generation and parsing. This library is maintained by the ProtonMail Team [But16] and used by a number of projects such as Mailvelope, ProtonMail, passbolt and others [Pro21b].

It provides support for common OpenPGP operations such as key generation, certification, encryption and decryption in JavaScript environment such as the browser. While it doesn't support all OpenPGP operations in particular more complex ones such as generating trust signatures, it is very convenient to use in the context of a browser application.

6.1.3. Advanced OpenPGP Operations in a Browser using WASM

Some advanced OpenPGP operations, such as in this case trust signatures, are not supported by JavaScript libraries such as openpgp.js yet.

Creating trust signatures in the browser thus meant either implementing the trust signature process in JavaScript or using an existing library from a different language such as sequoia-openpgp from Rust as a WebAssembly (WASM) component. WASM is a bytecode format for execution in sandboxed environment such as browsers that high level languages can be compiled into.

In this case, WASM was chosen as this allows PortuLock to leverage the existing OpenPGP library used in the rest of the project. This also demonstrates the feasibility of using WASM in such situations and provides an extensible solution for other advanced OpenPGP operations in the future.

The WASM program implementing trust signatures can be found in the folder `openpgp-trustsign-wasm` and consists of a single, small source code file and the metadata and dependency information for Cargo. It can be compiled to a JavaScript module for use in the Web Generator using `wasm-pack build`.

Future versions might switch from the openpgp.js library to a WASM component written in Rust for the entire key generation process to leverage Rust's type system and the sequoia-openpgp libraries extensive OpenPGP support.

6.1.4. Security Implication

PortuLock uses a web application to generate private keys for users. While this is very convenient for users it also brings a number of security implications.

Web applications are loaded directly from the web server each time they are used (ignoring caching) and could easily be replaced with a malicious server if it were compromised. In the case of this key generator this would allow an attacker to compromise the user's private key and exfiltrate it without being detected. They could then use it to decrypt communications and forge signatures for the entirety of the key's lifespan.

Such attacks on the key generation process through manipulations of the web generator are particularly dangerous in situations, where one organization hosts the keyserver for others. In these cases, the web generator itself could be served from each organization's own servers.

6.1.4.1. Detecting Attacks

Manipulated web applications are difficult for users to detect. Careful examination of the application's network traffic using the browser's developer tools could reveal attempts to exfiltrate private keys. However, sophisticated attackers could also compromise the key generation process to generate guessable or predictable keys, which would be even harder to detect.

Modification to the Web Generator could be detected by requesting it from the keyserver at random intervals while pretending to be a normal client and verifying that it matches the expected code. If the attacker cannot distinguish these checks from normal users that they want to attack, they have to risk detection.

6.1.4.2. Window of Opportunity and Impact

Attacks are also made more difficult by the small window of opportunity as keys are usually only generated once every few years and after generation, the private key is never available to the Web Generator or any other component of PortuLock ever again. This short window of opportunity combined with the long lifespan of keys also ensures that very few keys are affected if an attack is noticed in a reasonable amount of time and these keys can be detected by their generation date or certification date and revoked.

6.1.4.3. Native Applications

To prevent such attacks entirely, a native application for key generation could be created. It should be possible to convert the existing Web Generator into a somewhat native application using wrappers such as Electron [Ope21a] with little modifications. The advantages of a fully native app would include the ability to interact directly with smartcards and the user's existing OpenPGP client at the expense of considerably more development effort.

These native apps would have to be distributed to users through a trusted channel such as their operating system's package manager or through existing client automation solutions in place at the organization. If the user is expected to download the client's software from the keyserver itself, it would be just as easy to compromise and the consequences would be much more severe as native applications typically inherit the user's permissions. Here the advantage of operating in the web browser is that the user is protected by its sandboxing system and security guarantees.

6.2. Server

This section describes important aspects of the server's implementation such as the language used, important libraries and the chosen architecture.

6.2.1. Language

PortuLock uses the high-level programming language Rust [Rus21c] used most notably in Firefox [Moz20b] but also a number of other security tools such as Wire and Threema [Rus21b].

Rust manages memory by using an ownership system, where each variable is owned by a section of code and cleaned up when the program leaves the respective section. Ownership of variables can be transferred between sections and the compiler ensures that there is only ever one section with write permissions. It automatically handles memory allocation and deallocation at compile time and provides reference counted wrappers for the few cases where this is infeasible.

This ownership system works in conjunction with a strict type system to guarantee memory-safety and thread-safety for all Rust programs at compile time. This eliminates entire classes of bugs some of which could otherwise turn into security vulnerabilities.

Rust also provides zero-cost abstractions that provide convenient syntax for the programmer while also being evaluated at compile time and introducing no additional runtime cost. The automatic memory allocation is one such abstraction. It efficiently, conveniently and correctly handles memory allocation without relying on computationally expensive systems such as garbage collection.

Rust is the language of choice for PortuLock due to its memory-safety, its approach to avoiding runtime errors and to a degree for its performance and efficiency. However, the latter was not tested.

6.2.2. Web Server Framework

PortuLock uses the Rocket web server framework [Ben21c] to parse requests and call matching handlers.

Rocket is an easy-to-use web server framework for Rust that provides high performance, dependency injection and integrated support for databases as well as templating.

The database support manages database pools and provides connections where needed while the templating support enables the server to format responses according to templates parsed during startup. It replaces placeholders in the templates with data provided by the application before sending them to the client and can be used support translations or adapt the web pages to the organization's preferred styling.

Rocket leverages Rusts type system to provide strongly typed access to request parameters and validate requests, calling the associated handlers only if the parameters are valid according to the type specifications.

It also provides a configuration system based on Tom's Obvious, Minimal Language (TOML), that provides multiple ways of providing configuration values [Ben21b].

The performance was not tested and should improve further in Rocket's upcoming version 0.5 [Ben21a] (see subsection 6.2.4).

6.2.3. OpenPGP-CA Library

PortuLock uses the OpenPGP-CA library [Ope21g; Ope21d] for key storage and certification. It handles defines the format for the persistent databases used by PortuLock, ensuring interoperability with other OpenPGP-CA instances and allowing their databases to be imported. OpenPGP-CA also defines and provides the trust model of independent CAs per organization and trust scoped to domains.

Finally, OpenPGP-CA handles part of the exportation process of certificates to the WKD folders.

PortuLock wraps OpenPGP-CA to keep it easily replaceable and can be extended to support multiple similar backends at a time, selecting which one to use based on the email domain.

6.2.4. Cooperative Multitasking

PortuLock uses Rust's `async/await` feature and uses asynchronous code for most of its codebase. This avoids network and file system operations blocking the Thread and allows the server to perform other tasks while waiting.

The `async/await` feature is a form of cooperative multitasking which executes tasks on a runtime. Executing tasks on an in-thread runtime reduces the overhead incurred by the operating system switching tasks. When the operating system switches tasks, it has to uphold certain security and isolation requirements, which are not necessary in the case of such a runtime.

Currently, this is implemented by creating a new asynchronous runtime for each request, which will then be used to execute tasks associated with the request. Implementing it this way removes all performance benefits but allows most of PortuLock's code to be structured asynchronously.

Rocket's upcoming version 0.5 [Ben21a] will include support for asynchronous operations using a shared runtime for all requests, which can then make use of the asynchronous code and should increase the performance drastically.

6.2.5. OpenPGP Library

PortuLock uses the Sequoia library [Seq21j; Seq21b] for OpenPGP operations. These operations mostly consist of key parsing, cryptographic verifications and key filtering. In particular, PortuLock uses the libraries `sequoia-openpgp` for low-level interactions with OpenPGP and `sequoia-net` for key lookups from keyservers and via WKD.

Sequoia is a cryptographic library written in Rust that implements the full OpenPGP specification and exposes all of its features and options to the user. It can also be used in other languages through its C compatible function bindings. It provides extensive documentation for its usage and relevant parts of the OpenPGP specification including user guides for common interactions. Higher level libraries are available for other projects that might not want to deal with the low-level details of OpenPGP.

Importantly, Sequoia is a pure rust library with some low-level cryptography being handled by established C libraries. It also supports pure Rust cryptography libraries as a backend, which is useful in the WASM implementation mentioned in subsection 6.1.3.

Sequoia is owned and financed by the PEP Foundation [PEP21; Seq21k] and licensed under the GPL license [Seq21b; GPL], with some parts of the code being available under the LGPL license [Seq21h].

6.2.6. Architecture

PortuLock is split into two binaries that can be used independently or together and integrate with the help of a web server. This allows organizations to choose which components they need and scale them independently as needed. Each of the binaries provides a web server delivering a subset of the combined functions.

This design also provides resilience against failures of individual components.

6.2.6.1. Verifier

One binary, called “Verifier”, allows users to submit or update their key and verify their identity. This process has high security requirements and needs to maintain state; however it does not occur frequently and unavailability causes little disruption to normal operations.

It temporarily stores certificate data pending verification and verified identity information in an SQLite database and published certificate data in independent SQLite databases for each domain. Certificates can be exported from these databases into a directory structure to be published at the respective domain to make the certificates available via WKD.

The WKD used for publishing is simply a directory, that can be served and cached by any number of web servers. Delayed updates would be annoying but also don’t cause significant problems.

The Verifier service also handles key management such as updates, deletion and recertification.

6.2.6.2. Aggregator

The other binary, called “Aggregator”, allows clients to look up keys based on email addresses, fingerprints or KeyIDs by querying keyservers and WKD servers according to its configuration. Looking up keys will be a fairly common operation especially in larger organizations and also one, where availability is important. However, it can also be performed without keeping state, making it easy to scale.

6.2.6.3. Web Server

A generic web server will be deployed in front of the binaries. It can serve static data such as the web client for key generation as well as the certificates exported for publishing directly from their respective directories. For requests that require a dynamic response from one of the binaries, it will act as a reverse proxy, relaying requests to the correct binary that serves the relevant function, optionally caching some responses and balancing the load between multiple instances of the aggregator as needed.

As with the Aggregator, the web server does not need to keep state and serving outdated data does not cause significant problems. This makes it easy to scale as well.

6.2.6.4. External Reverse Proxy

PortuLock expects to be served behind an external reverse proxy that provides TLS termination.

Deploying web services behind reverse proxies is common in many organizations as this allows multiple services to be provided on the same IP address and port. It also allows TLS certificates to be handled in one place, which typically supports automatically issuing and renewing certificates as needed.

Organizations that already use such a reverse proxy for other services are unlikely to want to deploy the keyserver without one and requiring TLS termination to be handled externally ensures that PortuLock does not need to maintain TLS related code for the few remaining use-cases.

Such a reverse proxy can also be used to deploy PortuLock and its most of its APIs behind an additional authentication layer if desired. Unfortunately, the keyserver interface can only be protected using network-based access control as providing authentication information is not supported by typical clients.

Further information is provided in subsection 6.8.8.

6.3. Security

Security is a vital part of any CA's operation. It must ensure that it only certifies genuine keys and protect the private certification key at all costs.

The security implications of SSO and using a web generator are described in the respective sections (see subsection 6.5.6 and subsection 6.1.4).

6.3.1. Mail Verification

Authorization to use a particular email address in a UserID is verified by sending confirmation link to the email address. This ensures that users can obtain certificates for all email addresses that they control in practice which is particularly useful for aliases, that might not be known to the SSO system. It also allows shared keys to be created for group email addresses despite no user with the respective email address being known to the SSO system.

Similar email verification procedures are used by many other organizations including public X.509 CAs.

If only email addresses provided by the SSO system were accepted, this might prevent users from generating keys for their alias addresses, preventing them from receiving encrypted emails. It would also complicate the SSO system, requiring it to provide all valid email addresses that the user should be allowed to generate keys for.

Protecting the confidentiality of these emails in transit and ensuring that they are delivered to the correct target is left up to the configured email server.

Verification emails could be encrypted to the new certificate to prevent users from accidentally confirming a key that they don't control. However, this would also introduce challenges for keys that are not capable of encryption (such as pure signing keys) and might cause confusion if the user has not imported their private key yet.

Emails sent during name verification contain no secret information and are simply used to inform the user of the necessary verification despite responses from the keyserver not being shown to the user when uploading existing keys from OpenPGP clients.

6.3.2. Storing Certification Keys on the Keyserver

Storing certification keys on an internet connected server as implemented by PortuLock exposes them to additional risk compared to storing them on a system that does not provide public services accessible to attackers. In particular, it exposes the keys to more risk compared to using a dedicated, offline environment to store and use the key.

However, for this keyserver the usability of being able to certify user's keys immediately is worth it. This is a trade-off between the security of the CA keys on the one hand and their availability to certify keys on the other hand. Having keys available for immediate certification rather than for example collecting keys for offline certification by an administrator improves the usability and comfort when using the keyserver drastically.

PortuLock therefore uses keys stored on an online keyserver where they are more vulnerable but also available for immediate certifications and without requiring the administrator's intervention before signing keys. This trade-off decision is commonly taken by public CAs in the X.509 ecosystem for the same reason, though they often employ additional protections.

A few additional protections for the CA key could be implemented in future versions. The improvements outlined below can also be combined as needed.

6.3.2.1. Hardware Security Modules

The CA key could be stored in an HSM, which allows the host to sign arbitrary data but prevents key extraction. While this prevents an attacker from stealing the key, it cannot protect the key against misuse. An attacker compromising the keyserver could issue certifications that would be difficult to revoke individually, forcing revocation of the CA key. The attacker would gain very little benefit from also obtaining the actual private key.

Additional advantages of an HSM such as brute-force protection or human-interaction requirements cannot be leveraged in a server environment as the key needs to be available for use by the keyserver. HSMs are also difficult to use in virtualized environments, complicating the keyserver's operation.

6.3.2.2. Software Security Modules

CA keys could also be protected by using them in an independent software environment (similar to an HSM) that the keyserver can interact with. This environment could be a process running under a different user or available over a network interface and would provide most of the benefits of an HSM while being easier to use in virtualized environments.

It would also be able to log events and certifications independently and could impose limits on what can be signed, for example requiring certifications to include the correct time or refusing trust-signatures. It could even email the address in the UserID to aid in detection of malicious certifications.

6.3.2.3. Intermediary CAs

It would also be possible to implement a system of intermediary CAs where an offline root CA for the domain would be trust-signed by users and would in turn issue an unlimited trust signature over an intermediary CA used for daily operations. This would allow the intermediary CA to be revoked and regenerated as necessary without losing trust. Since all user certificates are stored on the keyserver, OpenPGP supports many certifications on each key and clients should refresh keys periodically. Certifications could simply be reissued using the new intermediary CA. Users or their clients would then only have to fetch the new intermediary CA, which would immediately be trusted due to its certification from the root CA.

6.3.2.4. Offline Certification

Finally, PortuLock could also just maintain a list of certifications to issue that would then be signed manually by an administrator from time to time.

This would force users to wait for the next certification round before their certificates are trusted by others.

It would also require the administrator to manually check this list before issuing the certifications as they otherwise simply turn into a “Human Security Module”.

6.3.3. Malicious Administrators

A malicious administrator with access to the server that PortuLock runs on, the SSO system or unencrypted backups of the CA key can certify arbitrary keys. Ultimately, PortuLock cannot prevent this, but users can detect attacks by additionally certifying each other's keys and occasionally comparing fingerprints for certificates directly.

In environments with high security requirements, the CA keys could also be partially trusted, obtaining the rest of the verification from the web of trust or even other instances of PortuLock operated by different administrators.

Mail server administrators can obtain certifications for arbitrary email addresses and manage arbitrary keys; however, they cannot have names certified without also compromising the SSO system.

More dangerous attacks would target the user's private keys and could not be detected after they have occurred. With PortuLock such an attack would involve serving a malicious Web Generator to users when they generate their key. The attack and possible mitigations are described in subsection 6.1.4.

Finally, anyone with administrative access to the user's computer can easily compromise any secret information including private keys handled on it, which cannot be prevented and only partially mitigated through the use of HSMs (also known as smartcards).

6.3.4. Omission Attacks

OpenPGP relies on revocation packets distributed along with the key to revoke parts of a key or entire keys that should no longer be used for example because they were compromised. Combined with the fact that there is no canonical location for keys to be stored and that keys are usually update infrequently (if at all), revocation in OpenPGP becomes unreliable.

An attacker that has compromised a private key might actively prevent the spread of revocation information to extend the useful lifetime of their compromised key. Further, an attacker might want to prevent users from finding keys for their communication partners in the hope of having them send unencrypted information.

Clients mitigate such omission attacks by treating keys as append-only and therefore retaining any revocation information they have ever seen.

PortuLock's aggregation feature makes omission attacks more difficult by only requesting information from authoritative sources for the relevant domain, ensuring that keyholders can easily publish revocation information without having to rely on third parties. It currently does not store any information between queries to avoid privacy complications but could be extended to cache some information, in particular revocation certificates. The problem with caching information from previous key lookups is that it can reveal which keys have previously been requested through timing attacks.

6.3.5. Expiring Certifications

Certifications issued by PortuLock confirm not only the keyholder's name and email address but also their membership in an organization, which can change in particular as members leave the organization. To prevent the keyholder from continuing to pose as a member of the organization, the CA can revoke its certification on their key. However, this causes the same omission risks as described in subsection 6.3.4 and causes problems if the user requests to have their key deleted or publication stopped. This would allow them to continue using their key while omitting the revocation information.

To avoid these problems, PortuLock issues certifications with an expiration date, after which they are no longer considered when validating the key. This ensures that revocation information about certifications only needs to be distributed until the certification expires, allowing the key to be deleted sooner. It also ensures that certifications don't remain valid forever simply because they were forgotten.

PortuLock issues certifications with a validity of one year by default and automatically recertifies all certificates in its database when they reach the last 10 percent of their validity. Both values can be configured to suit the organization's needs.

Revoking certifications and stopping recertification are currently manual processes but future versions of PortuLock might expose them as a feature in the admin interface or even perform them automatically when detecting that a user no longer appears in a list of organization members.

6.3.6. Stopping Fake Keys

One simple denial of service attack that can be performed using OpenPGP and key servers that don't verify control over an email address before accepting keys for it is simply generating and publishing fake keys.

If no key for the email address exists yet, the attacker can simply upload a single key, tricking communication partners into thinking that this might be a genuine key and using it to encrypt messages to the email address. The recipient, who might not even know about OpenPGP, will receive encrypted emails that they cannot decrypt.

If a genuine key already exists, the attacker can still generate a number of fake keys, forcing communication partners to find out or guess which one is correct. If they pick the wrong one, the same problem occurs for the recipient.

Of course, using an attacker-controlled key also allows the attacker to read the messages.

The obvious solution as implemented by PortuLock as well as keys.openpgp.org is to only publish keys if the key holder has proven their control over the email address. Clients should similarly be configured to only accept such keys.

6.3.7. Closing the Flood Gates

Another denial-of-service attack, known as "Key Flooding", can be performed against keys on traditional keyservers is generating an unreasonable number of certifications and publishing them to the server, "flooding" the key. This is problematic as traditional keyservers (except keys.openpgp.org) accept any OpenPGP packet related to a key with little verification, store it indefinitely and return all of them when a key is requested. [Gil19b; Fis19; Han19; Edg19]

When this key gets fetched by or imported into a client such as GPG, it will verify all certifications against locally available keys to calculate a trust score for the new key. This process seems to be fairly inefficient as discovered in a previous project [Esc21] and can take many minutes. In some cases, the key can even become too big to import at all.

Different solutions have been proposed or implemented to solve this problem. The keyserver operated at keys.openpgp.org simply refuses to accept certifications and distributes keys without them [Seq21d]. Another proposal suggests only accepting attested certifications, where the keyholder signs a list of certifications that they want to have distributed with their key [Gil19a].

Unfortunately, client support for creating such attested certifications is still limited, however a tool called "sq" [Seq21j] can be used to issue them [Seq21a].

PortuLock uses the latter approach combined with a whitelist of certification keys operated by the administrator of the keyserver. This allows certifications from CAs to be distributed without manual action from the user while also allowing individual users to accept certifications as they desire.

Future versions might also accept certifications from any key, that is already on the keyserver. This would allow members of the organization to easily certify each other's keys, which could be useful in situations where the server isn't fully trusted and members also rely on certifications from other members.

The same system also limits the number of certifications an attacker can use to generate a social graph of key holders as described in subsection 5.6.1.

6.4. Privacy

Keyservers such as PortuLock need to make trade-offs and balance privacy, security and usability by limiting what information they store and who they provide it to.

6.4.1. UserIDs

PortuLock protects UserIDs from enumeration and from being linked together due to their appearance on the same key by only returning them for direct matches of the email address. In particular, it does not include additional UserIDs for other email addresses when a key is requested. This allows users to conveniently use one key for multiple email addresses without revealing the whole set to anyone that obtains one email address.

6.4.2. Public Key Material

PortuLock treats public key material as personally identifiable data due to its uniqueness and because it refers to one specific individual (the keyholder). As a consequence, it allows the entirety of a public key to be deleted upon request to comply with privacy regulations.

Other keyservers either don't permit any deletion of keys or retain public key material such as Hagrid. See section 2.5 for details on the other keyservers.

6.4.3. Private Key Material

The most sensitive data in OpenPGP is the private key material. Not because of the personally identifying information it contains but because of the data it allows access to. PortuLock does not receive or store private key material and only briefly provides the code used in the user's browser for key generation. Details on security measures for this process are provided in subsection 6.1.4.

This ensures that PortuLock can never compromise past communication or signatures and at most generate and certify new, malicious keys.

6.4.4. Social Graph

PortuLock prevents attackers from constructing a graph of social interactions based on certifications on keys by requiring users to accept certifications on their keys before publishing them. This allows them to control which certifications and therefore which relationships with other keyholders will be publicly available.

See subsection 5.6.1 and subsection 6.3.7 for details.

6.4.5. Distributing Trust Signatures

The trust signatures issued by members of the organization over the CA key during key generation are statements from members that they trust the CA key to make verification and trust decisions on their behalf. These signatures instruct the user's clients to trust the key accordingly and should be available on all of them.

They can also be used by others to verify the CA key using a few otherwise trusted member keys. For example, new members might verify the certifications of other members whose key they have verified manually before trusting the CA.

OpenPGP does not permit these trust signatures to be distributed with the signer's key and instead requires them to be attached to the CA's key. OpenPGP-CA operates this way and distributes all trust signatures known to the CA directly on the CA's key. Unfortunately, this leaks the fingerprints of all keys that have trust signed the CA and submitted their trust signature to it, indicating that they are most likely organization members. While OpenPGP-CA does not allow certificates to be queried by fingerprint, this still leaks the number of users in the organization. Keys might also be found from other sources based on their fingerprint, providing additional personal information.

PortuLock does not accept trust signatures for publication on the CA key but could allow them to be stored alongside revocations, where they are only available to the authenticated keyholder for download and use in new clients.

PortuLock could also provide an interface to obtain trust signatures based on the signer's email address. Given the email address, the member's key can already be retrieved confirming their organization membership and providing their key's fingerprint. Thus, no additional information would be leaked by publishing the trust signature this way. However, this would require client support to be useful.

For now, the Web Generator will simply include the trust-signed CA key in the key bundle that the user downloads and the user is expected to move it to new clients together with their private key.

6.4.6. Signature Verification Versus Right to Deletion

Another trade-off that certificate storage systems such as PortuLock need to make is how long to keep data needed for signature verifications. Deleting data too quickly violates the non-repudiation guarantees of signatures because they can then no longer be validated and it can no longer be proven, that they were valid when the signature was initially received and acted upon.

On the other hand, users have a right to limit the use of their data to such necessary uses and have it made unavailable for any other uses.

As a multi-purpose keyserver, PortuLock leans on the side of deleting member data when requested (though this can be disabled) and not storing data about third-party certificates at all. It suggests that data needed for signature verification be stored along with the signatures to ensure their non-repudiation. This also ensures that this data will only be used to verify the associated signature and be deleted together with the signature when it is no longer needed.

If needed, an archival system could be placed between PortuLock and clients and record any certificates that pass through it, however finding out when this data can and must be deleted will be quite difficult in most situations.

6.4.7. Revocation Information and Revoked Data

The distribution of revocation information is another case where security and the right to deletion are in conflict.

In the case of encryption subkeys or certifications with an expiration data, this means distributing the revocation information at least until the component has expired. With signing subkeys (which can be used to sign backdated signatures) or primary keys (which can extend their own expiration date), revocation information has to be distributed indefinitely.

Currently, this requires administrators to disable self-service deletion and manually check if revocation information needs to be redistributed before complying with deletion requests. Future versions of PortuLock could check if a key contains revocation information and conditionally reject deletion or only delete data that is not needed to distribute revocation information.

6.4.8. Caching Lookup Results

PortuLock's aggregation feature does not cache any lookup results as this could be used in a timing attack to reveal which email addresses other users on the server have recently been interested in. How sensitive such information is depends on the environment and can range from benign to life threatening.

To securely operate such a cache, cached lookups would have to be limited to authenticated users and potentially scoped to the authenticated user, limiting the usefulness of the cache and complicating lookups.

Finally, caching also increases the risk of serving outdated information such as a valid version of a revoked key or a key with expired certifications despite new ones existing.

An alternative implementation of caching would only store and redistribute revocation information about keys, subkeys and certifications. In this case, users can reliably be made aware of revocations for other keys even if the original source no longer provides them at the expense of attackers being able to detect interest in the key. Depending on the circumstances, such a caching feature might be worthwhile.

6.4.9. Access Logs

Logs generated during keyserver usage can be useful when tracing errors or investigating potential security breaches. However, they also contain personal data such as submitted certificates and client IP addresses and can reveal communication patterns based on which certificates are requested or uploaded from which IP addresses.

All services including the bundled web server print log statements to their standard output, where they will be collected by Docker and can be collected, rotated or deleted as desired by the administrator.

By not passing client IP addresses to PortuLock from the reverse proxy described in subsection 6.8.8, administrators can ensure that the server only receives personally identifiable information during sensitive operations, where their storage can be justified more easily.

Administrators can decide not to forward client IP addresses to PortuLock from the reverse proxy described in subsection 6.8.8 for some or all requests to decide which ones will be logged. This can be used to only log personally identifiable information for actions that modify state while ignoring harmless read operations.

6.5. Single Sign On

PortuLock uses SSO to verify user's names before authorizing them for use in UserIDs.

6.5.1. Choosing OpenID Connect for SSO

For SSO there are three main protocols, that are commonly used and that are vendor-independent: LDAP, Security Assertion Markup Language (SAML) and OpenID Connect (OIDC).

Out of these LDAP [RFC4511] is the oldest and widely supported, however it lacks support for two-factor authentication or any other modern sign on features such as passwordless logins via Fast IDentity Online (FIDO) Web Authentication (FIDO2/WebAuthn). The premise behind LDAP is that the user provides his username and password to the web server, which relays it to the LDAP server for checking.

This system not only prevents modern authentication features and protocols such as two-factor authentication, especially phishing resistant two-factor authentication using FIDO Universal 2nd Factor (FIDO/U2F) or FIDO2/WebAuthn, but also requires the web server to securely handle the user's password.

SAML [OAS05; Wik21j] is quite a bit newer and supported by most major SSO systems, but also fairly complicated due to its use of XML. It is based on a system of redirects where the relying party (or service provider) redirects the user to the Identity Provider (IDP), which authenticates the user before redirecting them back to the relying party together with a token.[1] This cryptographically signed token includes details about the user such as their name and email address, which can be used to verify the user's identity.

Authentication can consist of any number of things defined entirely by the identity provider. If the user is already authenticated with the IDP via a Cookie, it might simply redirect the user back without further actions. The user might also be prompted to confirm which data to provide to the relying party if they have not used the application before. If authentication with the IDP is required, it typically consists of providing a username and associated password followed by a two-factor authentication step. However, some systems are starting to replace passwords (and sometimes usernames) with FIDO2/WebAuthn. These authentication options combined with not needing to handle

1 Simplified description. The system is much more complicated and highly configurable including backchannel communication via SOAP and many other authentication flows. See [OAS05] for details.

the user's password in every application make the system much more secure than LDAP and, since the user tends to have an existing session with the IDP, even more convenient.

OIDC [Sak+14] is the newest of the systems and simplifies the SAML protocol in many ways such as by not relying on SOAP and XML. Its tokens and procedures are reduced to the basics making it easier to implement and support. From a coarse overview and the user's perspective it works the same as SAML with all of its benefits.

Currently, the system only supports OIDC directly, which is provided by a lot of modern SSO systems. Where OIDC is not available, applications such as Gitlab [Git21] and Keycloak [Key21b] can often be used as an adapter by authenticating user against them using OIDC and having them authenticate users against other SSO systems. This is particularly relevant when authenticating against Microsoft systems, which usually rely on SAML and vendor-specific protocols, only supporting OIDC in some configurations. Future versions should likely support SAML to facilitate such integration.

6.5.2. Identity Provider Requirements

In addition to OIDC compatibility, SSO systems used by PortuLock must provide the name exactly as it should be used in the certificate. Future versions might offer configuration options for more lenient name comparisons to allow name components such as titles to be included or omitted.

Systems must not allow the user to change their name themselves and must only provide names that have been appropriately verified by the organization before storing them in the SSO system. Otherwise, the name verification becomes pointless. In most organizations, both of these requirements should already be fulfilled with identities being verified when members join the organization and attributes in the SSO system managed by administrators or other dedicated members.

6.5.3. Flexible Interface for Name Verification

The SSO interface also allows administrators to connect different tools for name verification to the keyserver. These could verify the user's identity and authorize their use of certain names internally, before sending them back to the keyserver with an appropriate, OIDC compliant token to confirm their name.

6.5.4. Governmentally Issued Identity Cards

In cases where identity verification using governmentally issued national identity cards is required or desired, SSO compatible identity verification services such as those defined in the go.eIDAS initiative [goe21] and implemented by services such as SkIDentity [ecs21]. These can abstract various different national identity cards, handling the governmental approval process for interacting with the identity cards and provide requested attributes such as the user's name to the requesting application [ecs21]. PortuLock supports such services as long as their responses can be adapted to conform to the OIDC protocol using suitable adapters. Future versions might also support SAML.

6.5.5. Manual Approval

In cases where names cannot be verified using the SSO system such as when generating group or role specific keys that use the group or role name rather than the keyholder's name, administrators can directly sign and submit the relevant internal tokens to approve the name or add it directly to the database. Future versions might offer an administration interface for such actions as well.

6.5.6. Security Implications

Name verification using SSO systems was chosen over manual verification methods for its high usability, availability and scalability. Almost all organizations already deploy SSO systems, many of which will be compatible with the OIDC implementation used by PortuLock and are familiar to users due to their other uses in the organization.

The main advantage of manual verification that bypasses the SSO system would be to ensure the integrity of OpenPGP certificates in the organization even if the SSO system was compromised.

However, in organizations that deploy them, SSO systems are usually already trusted with protecting access to almost all other web-based systems in the organization and the same credentials usually also provide access to endpoints, file shares and email systems. Consequently, SSO systems and their administrators are already ultimately trusted in the organization as they can impersonate any user including administrators to any of the organization's systems, operating in their name with little to no risk of detection.

Similarly, an attacker compromising the SSO system likely already has access to any system in the organization including client automation systems that allow them to compromise any endpoint and therefore compromise any secret or cryptographic key handled on these systems. They could likely also leverage those same credentials to log into the server that runs PortuLock, accessing the CA keys or issuing certifications as desired.

6.6. Configuration

This project uses the Rocket framework [Ben21c] for its web server components and can be configured through its configuration system. This means creating configuration files in TOML format for each of the services or providing configuration using environment variables. The configuration files are called `aggregator.toml` and `verifier.toml` respectively and are mapped into the containers as read-only files.

Ports, listening addresses and paths are provided in the docker environment but need to be configured otherwise.

Details on configuration options can be found in the Rocket documentation [Ben21b].

6.6.1. Verifier

The configuration consists of the domains, that the keyserver is responsible for and will accept in UserIDs in the key `allowed_domains`, a list of ASCII armored public keys, whose certifications are redistributed even without attestation in the key `allowed_certifying_keys` as well as configuration for OIDC (`oidc_*`) and mail delivery (`smtp_*`).

Additionally, two secret keys need to be provided in `secret_key` and `token_signing_key`, that are used to encrypt a cookie keeping temporary state for the OIDC process and sign tokens used for verification.

An example configuration is provided in Listing 6.1.

```
[global]
secret_key = "Ie<REDACTED>Bo=" # 256bit, base64
token_signing_key = "ZGVtbw==" # base64("demo")

external_url = "https://keyserver.example.org"

allowed_domain = "example.org"
allowed_certifying_keys = [
  "<ASCII Armored Public Key for a CA>"
]

smtp_host = "smtp.example.org"
smtp_port = 465
smtp_user = "openpgp-ca"
smtp_pass = "<REDACTED>"
smtp_from = "openpgp-ca@example.org"

oidc_issuer_url = "https://sso.example.org"
oidc_client_id = "<REDACTED>"
oidc_client_secret = "<REDACTED>"
```

Listing 6.1: Example Configuration for the Verifier Service

6.6.2. Aggregator

The configuration for the Aggregator service consists of a map containing special configuration for domains with known keyservers and CAs followed by a set of fallbacks that will be applied if no domain specific configuration can be found.

For each entry (both in the special domains and fallbacks) a set of keyservers can be defined and the administrator can choose whether to use WKD. The server will retrieve key information from all of these sources simultaneously before filtering any UserIDs it encounters by the requested domain. Afterwards, if a set of CAs is configured, only UserIDs certified by one of the CAs will be retained. Any remaining UserIDs will be certified as configured before returning keys to the client, that still have at least one UserID on them.

This provides a flexible system that can be adapted to the organization's needs regarding security, privacy and usability.

The example in Listing 6.2 shows the Aggregator configuration used by `example.org`.

```
[global.lookup_config.special_domains."example.org"]
use_wkd = true
expect_one_certification_from = [
"<Armored Public Key for openpgp-ca@example.org>"
]

[global.lookup_config.special_domains."alice.tld"]
use_wkd = true
expect_one_certification_from = [
"<Armored Public Key for openpgp-ca@alice.tld>"
]

[global.lookup_config.special_domains."bob.tld"]
keyservers = ["hkps://keys.bob.tld"]
certifier = "<Armored Private Key used for Certification>"

[global.lookup_config.fallbacks.wkd_fallback]
use_wkd = true
certifier = "<Armored Private Key used for Certification>"

[global.lookup_config.fallbacks.keyserver_fallback]
keyservers = ["hkps://keys.openpgp.org"]
```

Listing 6.2: Example Configuration for the Aggregator Service

Keys belonging to the organization can be queried from the WKD and are known to be signed by the organization's CA.

Keys for `alice.tld`, a partner organization that also uses tPortuLock's keyserver (or a different solution using OpenPGP-CA), are queried directly from their domain's WKD keystore as well and are similarly expected to be signed by their CA. Since their CA has been trust signed by the `example.org` CA, all of its members already trust the certifications and no on-the-fly certification is required.

Keys for `bob.tld` are not available via WKD but offered on the organization's keyserver using a secure connection. Unfortunately, these keys are not certified so the Aggregator service will sign UserIDs retrieved from this server that match the correct domain automatically as they pass through the service. The private key configured here has been trust signed by the CA for the respective domain.

For requests that don't match one of the above domains, both the WKD and the specified public keyservers are queried. Keys retrieved via WKD are certified as they pass through the service, while others are passed through without further trust being applied.

6.6.3. Generating the CA Certificate

When the verifier starts for the first time for a specific domain, it generates a CA certificate automatically.

The CA certificate will use the UserID `OpenPGP-CA <openpgp-ca@<domain>`, to customize this, in particular to change the name from "OpenPGP-CA" to the name of the organization, the OpenPGP-CA CLI can be used to generate the certificate before starting the service. See subsection 6.9.1 for further information.

This certificate will be used to sign all other certificates and will be published along with them. It can be exported using the OpenPGP-CA CLI and should be backed up securely.

6.6.4. Web Generator

Each time the web generator gets loaded in a user's browser, it fetches its configuration from the path `/config/ui.json` on the same domain that it was loaded from.

The `key_generation` object gets passed directly to the "openpgp.js" library during key generation and can be used to configure the key composition as desired. Subkeys, key usage flags and expiration times among other things can be defined using this key. See [Hui+21] for details.

The `trust_sign` array should contain a list of certificate authorities managed by this server. The generator will trust-sign these during key generation and provide the signed certificates to the user for download and importation into their client. They will also be provided in unsigned form on a "CA List" page.

Any requests will be performed against the PortuLock keyserver operating on the same domain, the web generator was loaded from.

The example provided in Listing 6.3 shows an appropriate configuration for the `example .org` organization.

```json
{
  "key_generation": {
    "type": "rsa",
    "rsaBits": 4096,
    "keyExpirationTime": 94608000
  },
  "trust_sign": [
    {
      "ca": "<Armored Public Key for the CA>",
      "domain_scope": "example.org",
      "name": "Example.org CA"
    }
  ]
}
```

Listing 6.3: Example Configuration for the Web Generator

The web generator can be served from multiple domains with independent configuration files if different configuration for multiple domains operated by the same keyserver is desired. Directing their users to the correct web generator instance is then up to the organization.

6.7. Distribution

PortuLock is provided as an open-source project under the GNU General Public License version 3.0 (GPL) [GPL].

6.7.1. Source Code

Source code is available in a git repository on Gitlab.com at `https://gitlab.com/portulock/portulock-keyserver`.

Gitlab.com was chosen because it already hosts the source code for other important and related projects in the OpenPGP ecosystem such as the Sequoia Project [Seq21b] and OpenPGP-CA [Ope21d]. This simplifies collaboration as potential collaborators would already be familiar with the platform and have appropriate accounts set up.

6.7.2. Packages

PortuLock is distributed as source code and no packages, containers or binaries are currently provided.

Compilation into docker containers (as described in subsection 6.8.2) is recommended to simplify dependency management and create a reproducible environment on the server.

Having users compile the source code provides them assurance that the resulting binaries actually match the source code and reduces the potential for undetected supply chain attacks [Gre21; ENI21; NCS18].

Rust crates and containers might be provided on crates.io [Rus21a] and Docker Hub [Doc21a] later to simplify use without Docker. However, using containers makes it difficult to verify that the source matches the container.

For development individual components can be compiled without Docker using `cargo build` or compiled and executed using `cargo run`, reducing compile times and allowing for debugging.

6.7.3. License

This project is released under the GPL [GPL] license as required by the OpenPGP-CA [Ope19] library for the verifier component in which it is used.

Choosing a different license would require removing the dependency from the verifier component and instead writing a suitable wrapper licensed under the GPL that wraps the library and provides its services over an inter process communication channel. All uses of the OpenPGP-CA library are concentrated in the `portulock-keyserver/verifier/src/key_storage/openpgp_ca_lib.rs` file.

The sequoia library used throughout the program is available under the "GNU Library General Public License version 2.0 or later" (LGPL) [Seq21h]. This means that it can be used in software licensed under different licenses as long as the user is provided with the tools to modify the library used by the application as they desire. This could be achieved through dynamic linking (which is somewhat difficult in Rust) or by providing the necessary source code under any license that allows at least unmodified recompilation.

To simplify licensing and because no serious issues are expected with the GPL license in an application rather than a library, the entire source code is provided under the GPL license. The library in the "shared" crate, that is shared by both components and might be suitable for use in other projects, could be provided under a license more suited for use in other applications such as the LGPL.

6.8. Deployment

This project uses Docker and Docker Compose to simplify dependency management and provide a consistent and isolated runtime environment.

6.8.1. Obtaining the Code

The source code for PortuLock can be obtained from GitHub using
`git clone https://gitlab.com/portulock/portulock-keyserver.git`.

All commits and tags in this repository are signed by the author of this thesis to ensure code integrity.

6.8.2. Compiling

This project uses Docker [Doc21b] and Docker Compose [Doc21c] to simplify installation and dependency management.

The services can be built from source by running `docker-compose build`. This fetches the dependencies needed to compile and run the project, compiles the rust binaries from source and builds docker containers, that will be executed later.

6.8.3. Starting the Services

The services can be started by running `docker-compose up -d`. This will compare the currently running services with the ones defined in the docker-compose file, creating, starting and stopping them as needed to achieve the defined configuration.

6.8.4. Scaling Horizontally

Docker Compose allows the administrator to start multiple instances of the same service using the `--scale` parameter when starting such as –scale aggregator=5 –scale nginx=10. Load will be balanced between these instances using the round-robin system of the docker internal DNS service. Additional tools such as Docker Swarm or Kubernetes could be used to deploy the containers if required.

This can also be used to omit unused services by scaling them to zero.

6.8.5. Updating Code and Dependencies

Updates to the source code can be retrieved using `git pull`, after which the release notes and change log should be checked for any changes that need to be made to the configuration or migrations that need to be applied manually.

Afterwards, the images can be recompiled and the service restarted in its new version as described above.

As this project uses Docker images, effectively installing entire operating systems (without their kernel), images should be regularly rebuilt even if the project itself did not change.

Unused images, especially the ones used for compilation can occupy a lot of space. They can be cleaned up using `docker system prune --all`.

6.8.6. Persistent Data

Only the configuration for the services and the OpenPGP-CA database needs to be persisted and backed up. The rest of the data is only temporary and can be recreated as needed. The WKD directory is derived from the OpenPGP-CA database and the verifier database only stores pending verifications that can simply be restarted as needed.

6.8.7. Use Without Docker

Should Docker not be desired, all of these steps can be performed without it by using the `docker-compose.yml` and `Dockerfile` files as a reference for what needs to be done.

6.8.8. Reverse Proxy for TLS Termination

PortuLock expects to be run behind a reverse proxy that provides TLS termination to simplify the setup and allow PortuLock to coexist with other services on the same IP address and port. Encrypted connections are required to ensure integrity, authenticity and confidentiality for clients performing lookups or submitting keys and especially preventing interception of SSO token information. Furthermore, WKD requires certificates to be served over TLS to ensure that they are indeed published by the domain and prevent attacks.

The same reverse proxy can also be used to log requests for audit purposes as required. For example, one might want to log requests to the certificate update and verification endpoints including their payload but might not want (or be allowed) to log lookup requests. Separating these logs from the application ensures their availability and integrity even if the keyserver itself might be compromised.

This reverse proxy must receive and forward traffic for the domain `openpgpkey.<domain>`, that will be used by WKD clients to locate keys and an additional domain such as `keyserver.<domain>` that clients will use to contact the server for API interactions and the web generator.

6.8.9. Multi-Tenancy

PortuLock supports multi-tenancy by allowing multiple domains to be configured and operating independent CAs for them. The databases storing persistent data are independent and self-contained, storing only data relevant to the respective domain and can be migrated between instances as needed. Published keys are similarly independent and there is no way to find out if a key has UserIDs on more than one domain, without guessing all email addresses or authenticating as the keyholder.

This allows an organization to use one instance of this keyserver for multiple email domains and receive independent certifications for them.

It can also be used to provide PortuLock's functionality as a service to partner organizations as long as a shared SSO system can be used to verify names. An example of such a SSO system usable for scientific institutions is the federated authentication and authorization system operated by the "Deutsches Forschungsnetz e.V." [DFN21]. Such a setup requires a lot of trust between the organizations as everyone must trust the hoster not to abuse its access to the CA keys and the SSO service to provide correct names for all users.

The user's interactions with the keyserver do not currently separate actions by organization or domain and instead work on combined information from all CA databases. Similarly, the database tables of approved names and email addresses as well as pending key packets are shared among all domains on the server. However, data stored in these tables expires after a relatively short period of time.

Future versions of PortuLock might extend this multi-tenancy support by allowing independent SSO providers to be configured for each domain and requiring those for name verifications. Similarly, other configuration options could be made available based on the domain such as token-signing keys or whitelisted certification authorities.

The configuration for the web generator can already be separated by organization as described in subsection 6.6.4.

6.9. Management and Administration

While PortuLock handles certificate issuance and key lifecycles automatically, some operations such as configuring trust in other CAs and some client automation must be handled by the administrator or existing client automation tools.

6.9.1. Certificate Authority Management

Ongoing management and administration of the CA currently needs to be performed on the command line using the OpenPGP-CA CLI, which can be installed by following the instructions on their website [Ope21c] or by running `cargo install openpgp-ca` (which requires Rust and Cargo to be installed).

Potential operations include:

- configuring bridging/trust between organizations [Ope21b]
- finding certificates that will expire soon and notifying users [Ope21f]
- bulk-importing existing keys [Ope21e]
- generating a keylist of all keys on the server [Ope21h]
- backing up the CA

OpenPGP-CA documentation [Ope21g] provides further details and syntax.

Future versions might provide an admin panel for some of the more common tasks.

6.9.2. Integration and Automation

PortuLock uses self-contained and cryptographically signed tokens to authorize actions. These tokens follow the JSON Web Token Standard [RFC7519] and can be created manually when integrating with other services or automating tasks given knowledge of the token-signing key from the configuration (see subsection 6.6.1).

These tokens can be used to authorize names and email addresses for use on keys by generating them and submitting them like a user would in the verification process. Names and Email addresses can also be approved by adding them directly to the database.

Additionally, these tokens are used to authenticate keyholders when performing some actions such as listing full keys or deleting them. Administrators can manually generate tokens if they want to perform such actions on behalf of the user.

This can also be used to extend the server with additional functionality by creating and submitting the relevant tokens as needed.

6.9.3. Preparing Clients

Clients using PortuLock should be configured to use the keyserver when publishing updates to organization certificates and when attempting to locate keys for communication partners. See Listing 6.4 for an example of a minimal configuration for GPG to use the PortuLock keyserver.

```
# Specify PortuLock as the keyserver
keyserver hkps://keyserver.example.org

# Configure PortuLock as the only source
auto-key-locate local,keyserver

# Note: Other settings such as cryptographic preferences are
   ommitted here but should also be configured if the defaults
   are deemed insufficient.
```

Listing 6.4: Example Configuration for a GPG client.

Additionally, keys for trusted CAs should be distributed to all clients. This includes the organization's CA as well as any bridged CAs from other organizations. These are needed as intermediary certificates during key verification but will not usually be fetched automatically. Such certificates could be listed in a keylist [MLW19] and fetched automatically using GPG Sync [Mic21].

Clients should also regularly delete local key packets, that they have not been able to refresh for a few refresh periods to ensure that they don't retain data, that is no longer publicly available and that the keyholder might not want to share anymore. Locally created or manually imported packets should be excluded from this deletion. Unfortunately, there seems to be no client software that currently does this.

6.9.4. Supported Mail Clients

Mail clients that rely on GPG as a backend are supported without further configuration. These include among others the Outlook plugin distributed with GPG4Win [Int21] and many clients on Linux systems such as KMail [KDE21b] and Evolution [GNO21].

Modern versions of Thunderbird integrate their own OpenPGP backend which does not support certifications (neither from CAs nor as part of the Web of Trust) and requires the user to manually verify and accept keys instead. To support these versions of Thunderbird, its cryptographic backend can be replaced with an alternative backend called “octopus” [Seq21i] that retrieves certificates and their trust level from the local GPG installation and makes them available to Thunderbird. Details and instructions can be found in [Seq21i].

Support for trust signatures is often limited with other mail clients as well. However, this can sometimes be mitigated by using the aggregation feature provided by PortuLock to re-certify verified keys and trusting this recertification certificate directly.

7. Evaluation

This chapter compares the requirements laid out in chapter 4 with PortuLock as described in chapters 5 and 6 before comparing it to the other existing solutions described in chapter 3.

7.1. Requirements

The following section evaluates how PortuLock meets the requirements laid out in chapter 4.

7.1.1. Ease of Use

It provides a browser-based key generator (as described in subsection 5.1.1) that can be configured by the administrator and requires as little interaction from the user as possible. The user simply provides name and email address combinations that they want to use and chooses a passphrase to protect the key. Afterwards, the web generator creates the key, trust-signs the CA and offers the data as a bundle for download and import into the user's regular OpenPGP client. By importing these trust-signed CAs together with the new private key, the user delegates trust decisions to the CA and instructs their client to trust them.

Finally, the tool submits the newly generated certificate to PortuLock, providing the user with a link to the status page and informing them of the verification process that follows. The private key never leaves the user's browser and their direct control.

During the verification process PortuLock sends emails to the user requesting them to click the contained link and follow a few simple instructions. Users can monitor the verification process from the status page, where they can also download the certified key once the verification has been completed.

Once the key has been verified completely, the user will not have to interact with the keyserver directly and can perform all operations using their OpenPGP client.

Providing organization-specific instructions is recommended to simplify the process further, provide guidance for configuration and use of the OpenPGP client and inform the user about any organization-specific policies.

A detailed description of a key's lifecycle is provided in chapter 5.

7.1.2. Locating and Verifying Keys of External Users

PortuLock provides an aggregation feature using the internal “Aggregator” service, that can be configured to retrieve keys for email addresses directly from the relevant domain according to a granular configuration defined by the administrator.

It allows these certificates to be retrieved via either the WKD or from a set of keyservers defined per domain. Once retrieved it can verify existing certifications on the keys and issue new ones on the fly as needed.

This allows PortuLock to take trust decisions on behalf of the user as configured by the administrator and in particular it allows validity to be assigned to certificates based on the fact that they were retrieved from a source that is authoritative for the domain and only provides genuine certificates for it.

The ability to verify that a certificate for a domain is certified by the domain’s CA also ensures non-repudiation of the CA’s statement about the identity and organization membership of the certificate holder.

This aggregation feature ensures that keys are only retrieved from the relevant domain, preventing leaks of communication patterns and partners to third-party keyservers. It also provides clients with a large pool of verified certificates that can be used without manual intervention and are known to be authentic.

7.1.3. Authentication and Authorization

Authorization to use a UserID is determined by PortuLock as required in section 4.3 and described in section 5.2.

UserIDs will only be accepted if the keyholder can successfully authenticate using OIDC confirming their membership in the organization and the mapping from their name to their key.

Once at least one UserID on a key has been authorized, it will be certified by the CA and published. These certifications expire after a configurable duration that defaults to one year and are automatically recreated before their expiration. This ensures that they cannot be used for an indefinite amount of time after the user leaves the organization and that keys will be refreshed periodically by relying parties to obtain new certifications.

7.1.4. Interoperability With the OpenPGP Ecosystem

PortuLock assures interoperability with the existing OpenPGP ecosystem and its clients by only relying on standards such as certifications, trust-signatures, keyserver interfaces and the Web Key Directory.

User certificates are certified by the CA key, which can be trust-signed by users and is published through the WKD. Users wanting to locate keys for external users can use the keyserver interface provided by the PortuLock to do so.

Compatibility with OpenPGP-CA is ensured by having its library perform the certification and storage operations once the certificates are approved by PortuLock.

Certifications from external users are published under the constraints described in subsubsection 7.1.5.2.

7.1.5. Privacy

The keyserver protects the privacy of its users and external communication partners and limits the spread of personal information where possible.

7.1.5.1. Preventing Enumeration

Keys are available only through the WKD protocol, which offers no interface to list keys. Additionally, certificates provided for WKD queries only contain the UserIDs matching the requested email address, hiding any other email addresses associated with the key.

7.1.5.2. Publishing Only Approved Data

PortuLock only publishes data that is cryptographically authenticated to have been accepted (attested certifications from third parties) or created (self-signatures, subkeys, revocations) by the keyholder. For UserIDs it also requires approval from the subject described by the UserID through name and email verification. Combined these checks ensure that no personally identifiable data is published without the approval from the person that is described by it.

To simplify the use of public certificate authorities, the administrator can define a list of keys at the instance level, whose certifications are accepted on all keys without individual approval.

Requiring that third-party certifications are attested by the keyholder (or whitelisted by the administrator) also gives the user control over which connections to other people they want to publish with their certificate, defining and limiting the social graph that can be generated from this information.

7.1.5.3. Deleting Personal Data

The key status page, which is accessible to users after they have proven their control over the key or an email address on it, allows users to trigger deletion of all data associated with the given key. A list of keys with a given email address can be requested from the server and will be emailed to the address to locate such keys.

If this feature is not desired, the administrator can disable it at which point the user will simply be instructed to contact the administrator for deletion.

7.2. Comparison

In the following PortuLock is compared to the existing solutions in the realm of email security and digital certificates described in chapter 3.

7.2.1. S/MIME Using Public CAs

Compared to using S/MIME with X.509 certificates issued by public CAs, PortuLock offers a lower cost per user and ease of use through integration with the existing SSO system used by the organization.

Rather than having to pay an external CA to verify member's identities and organization membership, PortuLock uses the existing SSO system in a self-service identity verification process.

The model of independent CAs for each domain operated directly by the organization also reduces the trust necessary in the CAs. In the X.509 certificate system, users must trust the certificate authorities to issue certificates for any entity under any domain and no options are provided to limit this trust. A single rogue CA could compromise all communications based on the trust model employed with X.509 certificates.

With the certification model as employed by PortuLock and OpenPGP-CA, a compromised CA only affects members of the CA's organization completely while others are only affected in that certificates for the compromised organization can be forged. This drastically reduces the impact.

The disadvantage of OpenPGP based encryption and signatures systems is that support in mail clients is not nearly as wide-spread and in mail clients that support it, OpenPGP requires configuration before trusting any certificate. S/MIME has the advantage that many mail clients support it out of the box and trust certificates issued by public CAs without additional configuration.

7.2.2. S/MIME Using Internal CAs

Using internal CAs with S/MIME brings a number of benefits for the internal use of the certificates similar to those provided by PortuLock. Certificate issuance can be made convenient and automated and no costs from external parties are incurred. The certificates can also be reused for other internal applications such as client-certificate-based authentication in mutual TLS or document signing. A trusted internal CA can also be used to sign certificates for server authentication to be used in the internal network.

However, problems arise when signed or encrypted mails pass the boundaries between organizations. External users wanting to verify the certificates must either verify each individual certificate or trust the organizations CA, which means trusting it for everything. X.509 certificates don't support scoping trust to certain domains, which means a user trusting an organization's CA must also trust the organization to not impersonate their bank or any other organization.

This makes these certificates very useful within an organization but effectively useless outside of it.

The certification model implemented by PortuLock and OpenPGP-CA solves this by offering most of the benefits of an internal CA, while supporting scoped trust to limit the trust put in another organization's CA to their own domain.

7.2.3. Unmanaged OpenPGP

Compared to unmanaged OpenPGP, PortuLock provides a scalable trust system that automates verification to ensure its consistency and drastically reduces the temptation to just accept the first key that is found because proper verification would be cumbersome and might not seem worth it.

This keyserver lowers the barrier of obtaining certifications for the user's key that can be trusted by the relying party by providing an easy-to-use certification service that can be used by authenticating with the existing SSO system and whose certification will be trusted by everyone in the organization.

Within the organizations certifications from PortuLock can replace the traditional web of trust as the keys for organization members can be expected to carry an appropriate certification.

PortuLock also ensures that the keys for all members of the organization are available on a keyserver, where they can easily and reliably be found when encrypting emails or verifying signatures. In environments without keyservers, requesting keys from their keyholders might take a significant amount of time and work to the point where the user might not even bother.

In addition, the absence of a current CA certification on an organization member's key can serve as an indication that their key might be fake or that they might not work for the organization anymore, something that the traditional Web of Trust can rarely be relied upon. It would typically not be considered suspicious if an organization member lacks certifications from close colleagues as this might legitimately occur for a number of reasons. Given the availability of this keyserver's certifications, it would be suspicious if an organization members certificate did not carry such a certification.

If desired PortuLock can also be used to augment the Web of Trust by assigning only partial trust to its CA and obtaining the other part of the trust from other sources.

7.2.4. Encryption Gateways

While encryption gateways only provide protection for data in transit between the gateway and the recipient's decryption point (gateway or client), they provide a number of benefits, when:

- encryption at rest is undesired
- protection against mail server compromise is deemed unnecessary
- interacting with external contacts that don't use S/MIME or OpenPGP, where password-based encryption or password protected web interfaces are often used
- both S/MIME and OpenPGP need to be supported depending on recipient
- messages need to be archived in unencrypted form
- scanning of unencrypted message contents is required

One important issue with encryption gateways is that they reduce the value of signatures drastically as signing keys are stored and used by the keyserver outside the user's control and could be abused without evidence by administrators or in the case of a server compromise. This reduces the integrity, authenticity and especially non-repudiation guarantees otherwise afforded by digital signatures.

In cases where encryption gateways are only deployed to ensure that email can be archived in an unencrypted format and to a limited degree in cases where they are deployed to be able to scan decrypted message contents, PortuLock extended with key escrow and escrow enforcement features could be used instead. Here PortuLock would only publish keys, for which the user has escrowed all key material needed for decryption (but not to create signatures or modify the key) and pass these decryption keys to the archival or scanning system. These archival (and to a limited degree scanning) systems could protect private keys much better than other parts of the mail server, that are more directly exposed to an attacker. Crucially, even if the system holding the decryption keys was compromised, message integrity could still be ensured due to the signature keys remaining in the sole control of the keyholder.

7.2.5. Encrypting Messengers

The cryptographic properties and ease of use of messengers are often excellent compared to email which is more constrained by its operating environment and design and messengers also tend to offer useful or convenient features. While this makes them useful for communication within organizations and teams (especially smaller ones), the need for all communication partners to use the same systems makes them difficult to use across organization boundaries.

While encrypting messengers have often been adopted in addition to email, they are unlikely to replace emails and the need to cryptographically protect them anytime soon.

7.2.6. OpenPGP-CA

PortuLock combines the benefits of OpenPGP-CA with self-service certification and lifecycle management.

From the perspective of an outside observer, PortuLock behaves exactly the same as OpenPGP-CA. PortuLock uses it internally and takes advantage of its database, certification and trust system.

The differences are mostly in the key generation, certification and key lifecycle management for organization members and in the additional key lookup features.

7.2.6.1. Certification Process

The key generation and certification process employed by PortuLock combines the security benefits of generating keys under the sole control of the user in their browser with the convenience and usability of key generation decisions taken by the CA as well as complicated operations such as trust-signature generation being performed automatically.

The web key generator provided by PortuLock generates keys as configured by the administrator including trust-signatures of any number of CAs and revocation certificates in the user's browser without transmitting their private key material to the CA.

With OpenPGP-CA these operations either have to be manually performed by the user or the administrator generates the keys and has the ability to retain the private key.

7.2.6.2. Automation

PortuLock allows users to perform all interactions with OpenPGP-CA that typically occur during the course of a key's lifecycle themselves. Administrator intervention is only required in special cases for example when role names should be approved, that are not provided by the SSO system or if errors occur.

In particular all verification and authorization to use specific UserIDs is automated and users can update or delete their key themselves.

Given suitable client tooling some operation could even be automated completely.

7.2.6.3. Bridging Trust to Other Organizations

PortuLock uses the same trust model as OpenPGP-CA but extends it by using its aggregation feature that can certify keys based on where they were retrieved from. The processes are identical and administrators are expected to use the existing OpenPGP-CA tooling and instructions for them.

This increases the network of verified certificates from other organizations using OpenPGP-CA by also including all organizations that publish their keys using the WKD or on a keyserver that validates identities before publication.

7.2.6.4. Interacting With the Rest of the World

Interactions with bridged organizations that also publish their keys using the WKD are simple as clients can automatically retrieve their key and will accept it as trusted without user intervention. When interacting with users from an organization that also uses OpenPGP-CA but which has not been bridged yet, users can manually issue a scoped trust signature over the organization's CA to yield the same effect for themselves.

Interactions with other organizations that also use OpenPGP-CA are exactly the same when using PortuLock. When interacting with organizations that don't use OpenPGP-CA however, PortuLock's aggregation feature becomes useful. As described above it extends the pool of verified certificates drastically by including organizations that provide centralized, validating keyservers and in particular by including all certificates published through the WKD.

In the remaining cases, users still need to obtain and verify certificates manually, however these cases should occur less frequently than with OpenPGP-CA.

7.2.6.5. Privacy

PortuLock fixes some data leaks in OpenPGP-CA, where the library provides more data than necessary.

UserID Enumeration OpenPGP-CA exports all UserIDs of a certificate for each entry in the WKD directory rather than just the ones matching the respective email address. This causes WKD requests to reveal UserIDs, that the requesting party might not have known about, potentially leaking other roles of the keyholder.

Fingerprint Enumeration OpenPGP-CA also stores trust-signatures of the CA key in its database and provides them together with the CA key. While storing such information on the key that is signed rather than the signer's key is an unfortunate design decision of OpenPGP, distributing trust-signatures from all keys in the database leaks their KeyIDs and fingerprints. This allows an attacker to enumerate the fingerprints of all keys in the database simply by checking the CA key's trust signatures, information which can be enhanced by querying the rest of the key from other sources such as public keyservers. PortuLock in its current form refuses to store trust-signatures (unless the CA key attests them), however future versions might offer a custom API that allows their retrieval given knowledge of the trusting key.

Key Deletion OpenPGP-CA also does not offer support for deleting keys from the database. The REST daemon offers to deactivate a key, which excludes it from future WKD exports but retains it in the database. An administrator having to delete a user's key is thus forced to manually edit the database using SQLite tools. PortuLock performs these manipulations automatically when a user deletes their key using the web interface, leaving no traces of the key's existence on the keyserver.

Privacy Preserving Keyserver Usage PortuLock also provides an aggregation feature, that can be configured to query specific keyservers for lookups of specified domains, allowing organizations to avoid the email address leaks associated with using keyservers from regular OpenPGP clients, that only support a global list of keyservers to query for everything.

7.3. Summary

PortuLock meets all requirements defined in chapter 4 and is ready for deployment and productive use in organizations. It provides a number of benefits over existing solutions in the field of email encryption and digital certificates.

PortuLock extends the OpenPGP-CA project with its certification system and trust model by providing automated identity verification, certification and key lifecycle management. It turns key generation, verification and certification into a self-service process that can be completed by users without administrator assistance, which allows the trust model to scale to large organizations, where manually verifying each certificate would take too much time.

For OpenPGP users in organizations it provides a trust model that scales well with the organization's size and automates verification to ensure its consistency. It makes obtaining certifications from a CA easy and convenient, making it feasible to require such certifications and reject uncertified keys as suspicious. The absence of a (recent) certification from the CA can be seen as an indication, that the user might not be a member of the organization (anymore). PortuLock also assists users in obtaining verified keys from a large network spanned by different certificate authorities without leaking email addresses and other personally identifiable information to third parties.

Compared to S/MIME the trust model brings the advantage of simultaneously simplifying certificate issuance and allowing certificates to be trusted externally, whereas with S/MIME the choice of an internal or public CA dictates one or the other. Additionally, the scoped trust reduces the amount of trust that has to be placed in individual CAs and limits the impact of their compromise from compromising the entire X.509 system to just affecting one organization's certificates.

If extended to support encryption key escrow, PortuLock could reduce the need for Encryption Gateways and make the decryption of mails on mail systems more secure by at least protecting signature keys and therefore the integrity and authenticity of messages.

8. Potential Extensions

8.1. Key Escrow

Another issue that might be holding back adoption of email encryption is the need for administrators to be able to decrypt most email communication without support from the keyholder.

This can happen for a number of reasons. For example, organizations are often required to archive emails for long periods of time (multiple years for business emails [Wik21b; Ion19; Jat21], potentially longer in the case of legal holds [Ext21]) and provide them in case of legal or authority inquiries. Other reasons can include the keyholder loosing access to their key, forgetting their passphrase or otherwise being unavailable when access to the data is required.

One way of supporting these scenarios is called "key escrow", whereby the keyholder provides their private key to an administrator for archival, who can then store and provide it as needed. Compared to the alternative of decrypting emails on an encryption gateway as discussed in section 3.4, this has the benefit that private keys can be stored much more securely in offline systems as they are needed much less frequently.

To ensure continued availability of communications when required organizations might have to enforce key escrow by ensuring that only escrowed keys can be used for encryption. PortuLock could be extended to receive, verify and store escrowed (sub)keys ensuring that only escrowed keys will be certified and published and effectively preventing the use of other keys.

8.1.1. Only Escrowing Encryption Subkeys

One advantage for OpenPGP when it comes to key escrow compared to X.509 is that certificates can contain multiple key pairs used for different purposes. For example, one might have a primary key capable of certifying other keys and authorizing changes to the key, while signatures and encryption are handled by independent subkeys.

This allows for the encryption subkey to be escrowed to ensure that encrypted data can be decrypted when needed while ensuring that only the key holder can perform signatures or authorize changes to the key, preventing impersonation of the user by an administrator.

8.1.2. Procedure

With the proposed system, the client would generate a key with a primary key incapable of encryption and a subkey capable of only encryption. It would then submit the full public key to the keyserver, which would publish the primary key but withhold the encryption subkey.

The client would then export any encryption capable subkeys individually and without passphrase protection but encrypted to a specific escrow public key. This, combined with some metadata to identify the associated subkey, would form an "escrow message", which would then be submitted to the server.

The server would verify this escrow message (see subsection 8.1.3) before storing it (see subsection 8.1.4), noting that escrow data for the subkey with given fingerprint is available and publishing the withheld subkey.

8.1.3. Verifying Escrow Messages

One significant problem with (mandated) key escrow is verifying that the escrowed private key actually works without compromising its security. The private key has to be briefly available in unprotected form to verify that it matches the associated public key as claimed before it can be encrypted for an offline key and stored securely.

Depending on the confidentiality and availability requirements in an organization, different approaches are possible ranging from trusting the user/client to provide a valid escrow message to verifying the escrow message in a separate component addressed over the network or in a trusted computing environment.

Only subkeys escrowed during a potential server compromise will be vulnerable either way, reducing the impact of a breach substantially. Furthermore, these keys will likely be relatively fresh, meaning little data has been encrypted to them yet.

8.1.4. Securely Storing Escrow Messages

Securing the confidentiality of the escrow message during long-term storage is more important but also much easier to achieve. Escrow messages should be encrypted to an offline asymmetric keypair, which should in turn be kept under dual control [Bar+05; Dic02; Wik21k] to avoid compromise and abuse. Secret sharing [Sha79; Wik21i] could also be employed. For example, the escrow message could be encrypted symmetrically, with the symmetric key then split up using secret sharing schemes, that require a threshold of shares to be available for decryption. These shares could then in turn be encrypted to one individual each, ensuring that at least a threshold number of people must concur to decrypt any escrow message.

8.2. Enforcing Policies for Key Structure

The keyserver could also be extended to enforce policies for the required structure of keys. Such policies could include key algorithms, key sizes, user preferences, key flags, subkeys or any other aspect of the key. In particular one could require proof from the client that parts of the key such as the signing subkey were created on a supported hardware token by requiring an appropriate attestation (see [Yub21]). This would prove that only one instance of the private key exists.

Such policy enforcement would be particularly useful in regulated industries or when an organization wants to ensure a uniform external representation.

8.3. Multi-Tenancy

The server could be extended to support more separation between domains to enable multi-tenancy and allow one organization to operate the servers for many other organizations. Currently, multiple domains are supported with independent permanent keystores and CAs, however many other settings and databases are shared among the domains. PortuLock's current version shares the database of certificates pending verification as well as approved names and the configuration for email sending, SSO authentication and some other values.

Even with more separation between domains, the hoster still needs to be fully trusted as they have full and unmonitored access to the CA and could retain unencrypted copies of any escrowed keys without detection being possible on the client. At least the verifier service should be operated by the organization, while WKD and aggregator services could be outsourced more securely.

8.4. Support for Web Key Submission

The specification for WKD also describes a protocol by which mail clients can send keys to the directory and verify that control over the email address, so they can be published automatically.

This involves the client emailing the key to a submission address and then proving control over the email address through a series of encrypted emails exchanged with this address.

This could be implemented in the form of an additional service interacting with PortuLock. It would complete the exchange with the mail client before submitting the key to the keyserver and marking the email addresses as verified. The user could then use their mail client's features to generate a key and submit it to the keyserver through the Web Key Submission protocol, meaning they only have to verify their name address using the link emailed to them. However, this process would not generate trust signatures or import the CA into the client.

8.5. Native Clients

Switching from a web client to a native client that can be distributed like normal software rather than being fetched from the keyserver each time would make it much more difficult for a compromised or malicious keyserver to serve a manipulated key generation software that exfiltrates the private key.

A native client could also automatically configure any existing client software as needed.

For this initial version the web generator offers a cross-platform compatibility and convenience by not requiring local installation and potentially permissions.

9. Future Work

9.1. Client Improvements for Interoperability

For ideal interoperability with keys published in WKD rather than on centralized key-servers, clients should:

- regularly update local keys by querying the WKD for all email addresses in the key's UserIDs
- delete local keys, that could not be refreshed for a while to ensure that keys, that the owner no longer wants to be published, are eventually deleted (revoked keys may have to be treated differently)
- fetch keys from a configurable keylist [MLW19; McC18] to ensure all (bridged) CA keys are present
- support fetching keys mentioned in the keylist via WKD in addition to or instead of from a keyserver
- support verifying and issuing trust signatures
- support issuing certification attestations to indicate which certifications are accepted for redistribution ("Attested Key" signature type) [Koc+20]
- include the signer's email address in signatures ("Signer's User ID" subpacket) [RFC4880]
- fetch signature keys using included email addresses when validating signatures
- (optional) fetch certifying keys using the email address added to the certification signature when verifying the validity of keys (this can reveal interest in a key to anyone that certified it)

Some of these features are already supported by some clients or can be added by installing and running additional software such as "GPG Sync" [Mic21]. Others are implemented but need to be triggered manually by the user or mail client.

9.2. Indicating Signature Policies on OpenPGP UserIDs

One user experience problem when verifying email signatures is that it is usually unclear if an email should have been signed and if the signature was omitted because the sender forgot it or because an attacker manipulated the email. In these cases, it is up to the user to detect the absence of a signature, which is much harder and less likely than noticing a security warning.

This could be implemented by defining a signed subpacket of the signature that binds the UserID to the key containing the appropriate information. This could then be evaluated by mail programs to compile a list of email addresses, from which signatures are expected and warn the user if an email claiming to be from one of these addresses is not signed.

Such checks could even be performed in the Simple Mail Transfer Protocol (SMTP) stage by having the mail server query the WKD for the sender's email address and check for suitable signatures. It could then reject the email and augment other systems to detect email forgeries such as DMARC [RFC7489].

9.3. Support for Scoped Trust in X.509

The X.509 certificate systems could be extended to include similar scoped trust systems to the one implemented in OpenPGP. This would allow public CAs to issue subordinate CAs to customers that are scoped to their domain, allowing the customer to issue certificates for email encryption or subdomains with a domain verification level of assurance themselves. It would also allow CAs to limit their own subordinate CAs to specific top-level domains to reduce the impact of their compromise. Finally, it would allow operating systems and browsers to limit trust in less trusted CAs to the countries' TLDs, where they are commonly used rather than accepting them for any TLD.

9.4. Automatic Discovery of Certification Policies and CAs

A standard for documenting, publishing and discovering certification policies for domains could be defined and implemented. This would allow organizations to inform clients that all UserIDs containing their domain should be signed by a specific CA and that other certificates are not to be accepted. A client could then refuse to accept such UserIDs even if they would otherwise be considered valid.

Additionally, an organization could inform clients that there are public keys for all email addresses associated with the domain and that all outgoing emails should be expected to be signed. This would allow clients to refuse sending unencrypted email to the domain if no keys are found and reject unsigned emails as forgeries.

These policies should be cached and continue to be enforced even if they are currently unavailable from the target domain due to an attack.

10. Conclusion

PortuLock enhances the OpenPGP ecosystem by combining a keyserver with a certificate authority and providing a self-service solution for key generation, certificate issuance, publication and life-cycle management of OpenPGP keys in organizations.

It allows users to easily generate their OpenPGP key before authenticating them using the organization's existing SSO solution and certifying and publishing their key. This makes it the first OpenPGP keys certificate authority, that can be operated by any organization and provides self-service name verification, certification and publication.

It respects and protects the user's privacy throughout their key's lifecycle, limiting publicly available information to a reasonable minimum and allowing the user to control what information can be attached to their key. It ensures that information is only published with the approval of the subject to which it pertains and allows users to delete their keys if they no longer want them to be published.

PortuLock leverages the network of verified certificates spanned by the OpenPGP-CA project's trust and certification system. It allows organizations of all sizes to join the network easily and while using the scalable self-service certification system offered by PortuLock rather than manually verifying and certifying keys as required by OpenPGP-CA itself.

Another important feature of PortuLock is its ability to fetch certificates directly from the email domain for which they were issued according to a flexible configuration. This prevents the email address leakage when querying centralized keyservers and provides a degree of verification that the certificate is genuine. This verification can be extended to the client by issuing certifications on the fly, delegating trust decisions from the user to the keyserver and its administrator and increasing the pool of verified certificates.

PortuLock increases the usability of OpenPGP by making it simple for less experienced users to obtain a certificate, that is trusted by others and get verified keys from a large network of keys certified by other organizations and trusted by their own organization's CA.

In summary, PortuLock solves both the problem of privacy preserving key distribution and lookup as well as the problem of verifying keys of internal and external communication partners, by providing a combined keyserver and automated certificate authority which is fully compatible with the existing OpenPGP ecosystem.

11. Outlook

While OpenPGP has been around for decades and successfully used in many areas, its adoption by a broader audience of has long been hindered by the lack of simple and usable tooling hiding the complexity and making complex concepts understandable.

Existing OpenPGP tools often confront users with complex decisions and questions, the answers to which are clear to their developers but much less clear to the user, that just wants to know if a particular document was signed by a specific person.

There are a number of projects that aim to increase the adoption of OpenPGP such as Keybase [Key21a] and Keyoxide [Yar21], both of which link digital identities on popular platforms to certificates, allowing users to verify a key by automatically comparing it to statements posted under the user's known accounts on these platforms.

The Mailvelope browser plugin [Mai21; BSI19], Roundcube webmail client [Rou21; SER21] and a number of email services [1121; Pos21; Pro21a] also provide support for OpenPGP when using webmail.

Autocrypt [Aut20a] provides opportunistic encryption between supporting mail clients, protecting against passive adversaries and from leaks of emails stored on a server.

In general, there are many applications that would benefit from usable digital signatures being available to users.

Currently, three types of signature systems seem to be used. The most secure ones are often governmentally regulated, suffering from very high security requirements at the cost of usability and availability to end users. They often require special signature cards to be used in expensive card readers and can often be used for limited applications and with special signature software, making them only useful for groups such as doctors or lawyers.

On the other end of the spectrum are signature systems that rely on emailing scanned documents that were signed on paper or overlaying a picture of a signature onto a document.

Somewhere in between are cloud-based signature services that verify identities using email addresses or phone numbers and record signatures as an independent third party. However, if such a service gets compromised or simply ceases operation, signatures recorded by it might be lost and nothing protects users from a rogue signature service. Signing documents this way also reveals their sensitive contents to the signature service.

Certificates issued by an independent party (or the user's organization with records kept elsewhere) already provide much more security and resistance against attacks, while still being easy to use and accessible.

A. Example of an OpenPGP Key's Composition

```
:public key packet:
        version 4, algo 1, created 2019-10-05
        pkey: [[4096 bits],[17 bits]]
        keyid: 2DD7C9487DFCC04D
:user ID packet: "Erik Escher <erik@erikescher.de>"
:signature packet: algo 1, keyid 2DD7C9487DFCC04D
        version 4, created 2021-10-02, sigclass 0x13
        digest algo 10, begin of digest 09 0c
        hashed subpkt 27 (key flags: 01)
        hashed subpkt 11 (pref-sym-algos: 9 8 7 2)
        hashed subpkt 21 (pref-hash-algos: 10 9 8 11 2)
        hashed subpkt 22 (pref-zip-algos: 2 3 1)
        hashed subpkt 30 (features: 01)
        hashed subpkt 23 (keyserver preferences: 80)
        hashed subpkt 25 (primary user ID)
        hashed subpkt 33 (issuer fpr v4
           F961186F6EE185D7732F12FA2DD7C9487DFCC04D)
        hashed subpkt  9 (key expires after 4 years)
:signature packet: algo 1, keyid 5E5CCCB4A4BF43D7
        version 4, created 2019-10-05, sigclass 0x13
        digest algo 8, begin of digest 65 59
        hashed subpkt  5 (trust signature of depth 1, value 60)
:public sub key packet:
        version 4, algo 1, created 2019-10-05, expires 0
        pkey: [[4096 bits],[17 bits]]
        keyid: E97382BF2E407967
:signature packet: algo 1, keyid 2DD7C9487DFCC04D
        version 4, created 2019-10-05, sigclass 0x18
        digest algo 10, begin of digest 91 3e
        hashed subpkt 27 (key flags: 0C)
```

Listing A.1: Example of a an OpenPGP Key's Composition.

Output as generated by `gpg --export <key-identifier> | gpg --list-packets`. (shortened and formatted for presentation and clarity).

Note: A similar version of this appendix entry has been published in my previous project "git-anon" [Esc21].

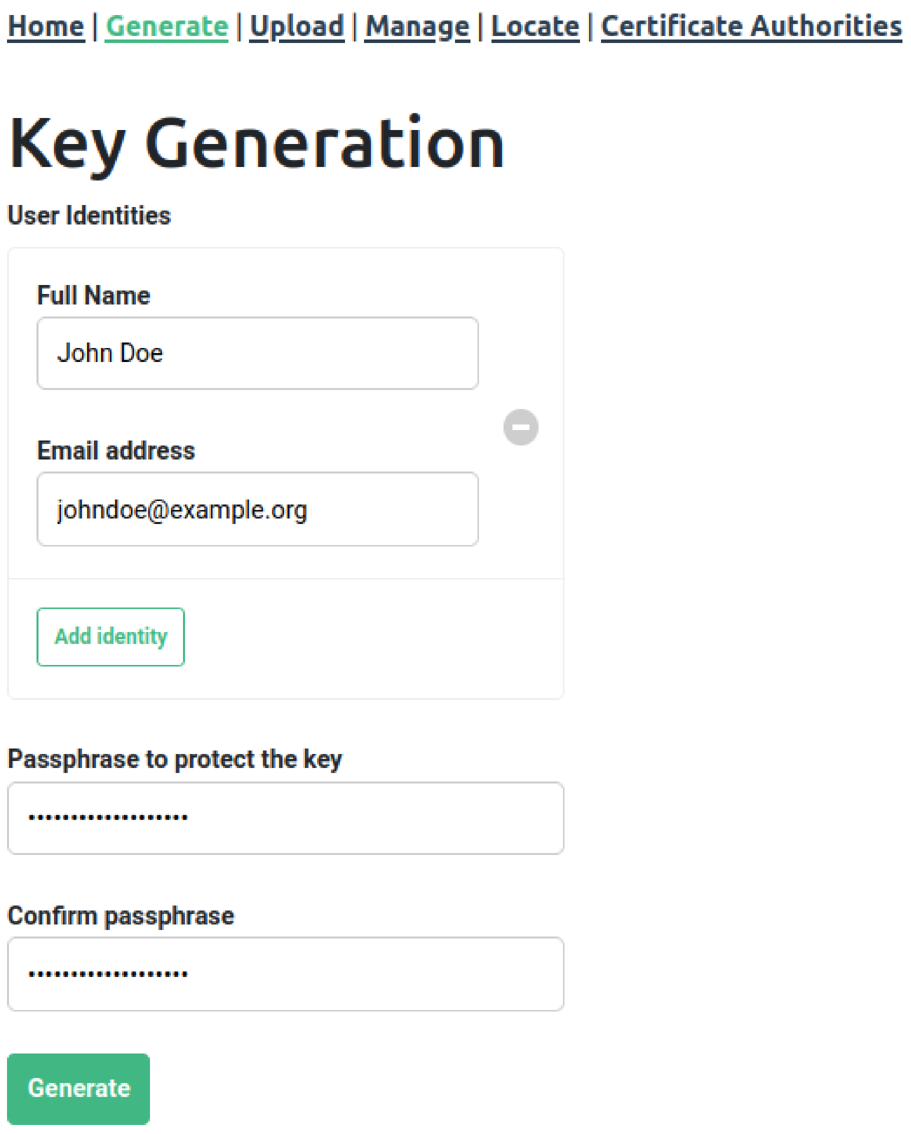
B. Open-Source Release

The source code for PortuLock is available in the form of a git repository at `https://gitlab.com/portulock/portulock-keyserver`.

Hashes for the initial commit and version 0.1.0 are available below and all commits are signed using OpenPGP.

Commit for v0.1.0	`c94e74424928f00138ea7fbf6015b21baf4e445b`
OpenPGP Fingerprint	F961 186F 6EE1 85D7 732F 12FA 2DD7 C948 7DFC C04D

C. Web Generator Screenshots


Home | Generate | Upload | Manage | Locate | Certificate Authorities

Key Generation

User Identities

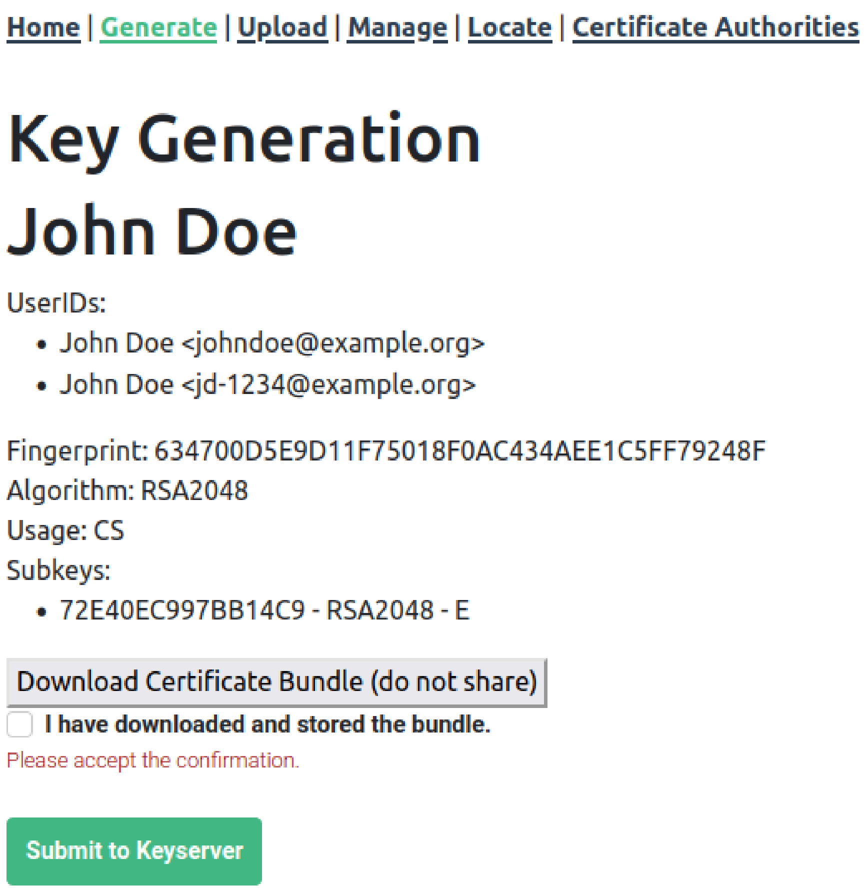
Full Name: John Doe

Email address: johndoe@example.org

Passphrase to protect the key: ••••••••••••••••••

Confirm passphrase: ••••••••••••••••••

Figure C.1.: Screenshot of the web generator while providing UserIDs and a passhprase.



Home | Generate | Upload | Manage | Locate | Certificate Authorities

Key Generation

John Doe

UserIDs:

- John Doe <johndoe@example.org>
- John Doe <jd-1234@example.org>

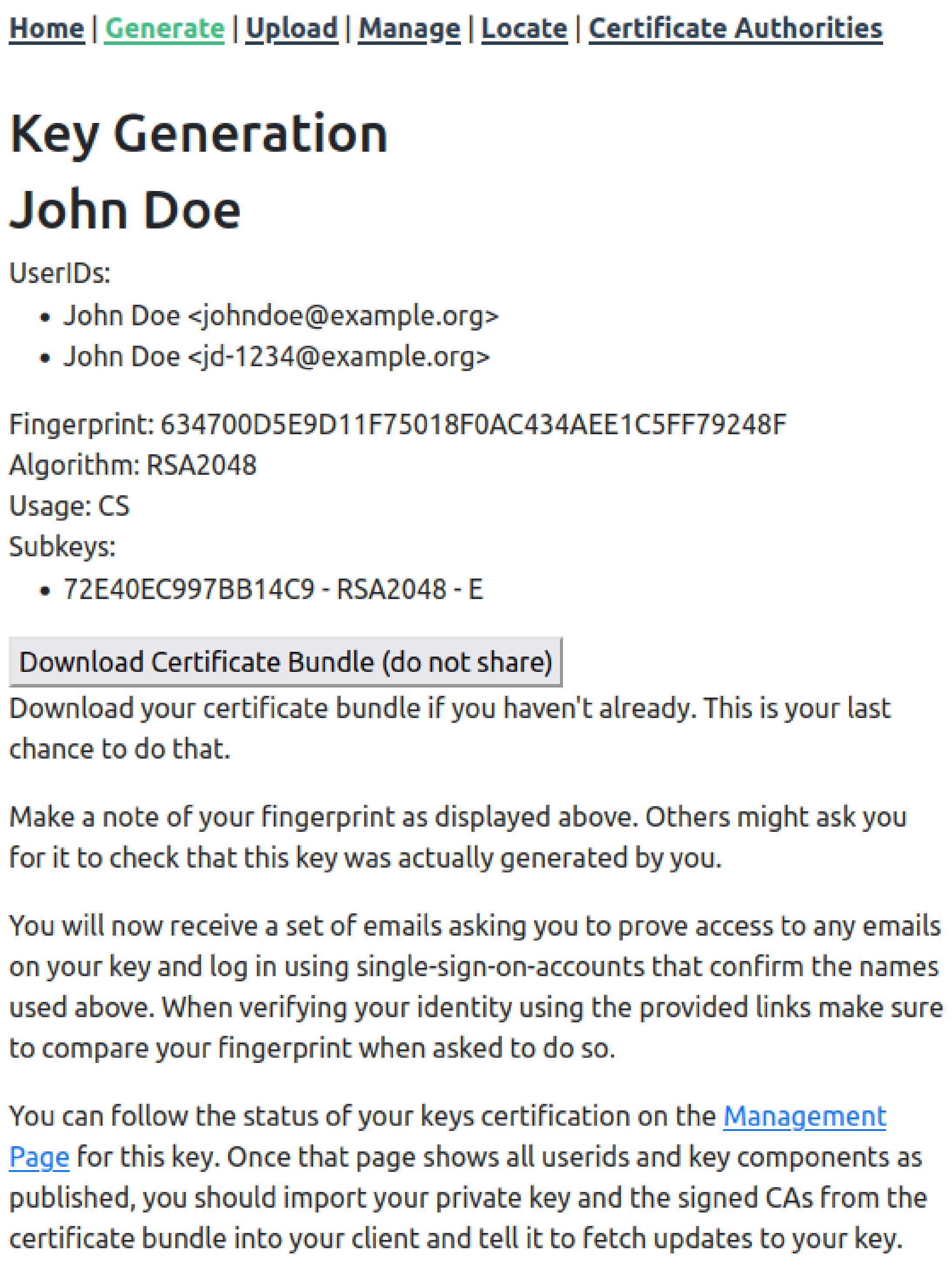
Fingerprint: 634700D5E9D11F75018F0AC434AEE1C5FF79248F
Algorithm: RSA2048
Usage: CS
Subkeys:

- 72E40EC997BB14C9 - RSA2048 - E

Download Certificate Bundle (do not share)
☐ I have downloaded and stored the bundle.

Please accept the confirmation.

Figure C.2.: Screenshot of the web generator after key generation.


Home | Generate | Upload | Manage | Locate | Certificate Authorities

Key Generation

John Doe

UserIDs:

- John Doe <johndoe@example.org>
- John Doe <jd-1234@example.org>

Fingerprint: 634700D5E9D11F75018F0AC434AEE1C5FF79248F
Algorithm: RSA2048
Usage: CS
Subkeys:

- 72E40EC997BB14C9 - RSA2048 - E

Download Certificate Bundle (do not share)

Download your certificate bundle if you haven't already. This is your last chance to do that.

Make a note of your fingerprint as displayed above. Others might ask you for it to check that this key was actually generated by you.

You will now receive a set of emails asking you to prove access to any emails on your key and log in using single-sign-on-accounts that confirm the names used above. When verifying your identity using the provided links make sure to compare your fingerprint when asked to do so.

You can follow the status of your keys certification on the Management Page for this key. Once that page shows all userids and key components as published, you should import your private key and the signed CAs from the certificate bundle into your client and tell it to fetch updates to your key.

Figure C.3.: Screenshot of the web generator after submitting the key to PortuLock.

D. List of Listings

E. Abbreviations and Glossary

API

Application Programming Interface

ASCII

Character encoding for common English characters.

CA

“Certificate Authority”. Entity or Public Key that is confirms the mapping between an entity and a different public key by issuing a signed statement.

Cargo

Package manager and compilation orchestration tool for Rust.

CLI

Command Line Interface

DANE

“DNS-Based Authentication of Named Entities”. Standard for publishing information about public keys in DNS. See [RFC7671].

DKIM

“DomainKeys Identified Mail”. See [RFC6376].

DMARC

“Domain-based Message Authentication, Reporting and Conformance”. See [RFC7489].

DNS

“Domain Name System”. See [RFC1034; RFC1035].

DNSSEC

“Domain Name System Security Extensions”. Standard for cryptographically signing DNS entries to confirm their authenticity and integrity from the domain owner to the end user. See [RFC4033].

FIDO

“Fast IDentity Online”. See [FID21].

FIDO/U2F

“FIDO Universal 2nd Factor”. Standard for phishing resistant two-factor authentication. Uses a challenge response scheme with cryptographic signatures whose keys are unique per domain and user account. Typically used with and implemented by hardware tokens. See [FID21].

FIDO2/WebAuthn

“FIDO Web Authentication”. Extension of FIDO/U2F intended to be used as the only means of authentication. Hardware tokens typically authenticate the user using a PIN or biometrics. See [FID21]

GDPR

“General Data Protection Regulation”. See [GDPR].

GPG

“GNU Privacy Guard”. Most common client implementation of the OpenPGP standard. [g1020]

HSM

Hardware Security Module

IDP

“Identity Provider”. As used by SAML, OIDC and SSO systems in general.

IP

Internet Protocol

LDAP

“Lightweight Directory Access Protocol”. Database access protocol used among other things for authentication of users. See [RFC4511].

OIDC

“OpenID Connect”. SSO protocol. See [Sak+14].

OpenPGP

Specification for asymmetric cryptography using various algorithms. See [RFC4880].

OpenPGP-CA

Project and Tool to operate a CA for OpenPGP. See section 3.6.

REST

Representational State Transfer

S/MIME

“Secure/Multipurpose Internet Mail Extensions”. Standard for email encryption based on X.509 certificates. See [RFC8551].

SAML

“Security Assertion Markup Language”. SSO protocol. See [OAS05; Wik21j].

SMTP

“Simple Mail Transfer Protocol”. See [RFC5321].

SOAP

Messaging protocol for web services based on sending messages via one of numerous communication methods.

SPF

“Sender Policy Framework”. See [RFC7208].

SQLite

File-based database format and associated tooling.

SSO

“Single-Sign-On”. Unified authentication system in an organization. Users have accounts with the system and use these to authenticate against (almost) all services in the organization rather than using independent accounts and credentials for each of them.

TLD

Top Level Domain

TLS

Transport Layer Security

TOML

“Tom’s Obvious, Minimal Language”. Format for configuration files.

URI

“Uniform Resource Identifier”. More commonly known by its old name of “Uniform Resource Locator” (URL).

UserID

String describing a user, typically containing a name and an email address. See subsection 2.3.1.

UTF-8

Character encoding for international characters.

WASM

“WebAssembly”. Open standard for bytecode to be executed in sandboxed environments such as browsers. Programs can be compiled to WASM from a number of programming languages and will then be compiled to platform specific code by the execution environment. Can increase performance over JavaScript and permits the use of existing libraries from other programming languages. See [W3C19].

WKD

“Web Key Directory/Discovery”. A system for retrieving OpenPGP certificates by email address directly from the email domain. WKD can refer to both the directory storing and providing the certificates as well as the process of retrieving certificates given an email address. See subsection 2.5.6 and [Koc21; BSI21].

X.509

Standard for digital certificates using a hierarchy of CAs. See [RFC5280; Wik21l].

XML

Extensible Markup Language

F. Bibliography

[1121] 1&1 Mail & Media GmbH. *Verschlüsselte Kommunikation: Einrichtung*. German. 2021. URL: https://hilfe.web.de/sicherheit/pgp/verschluesselte-kommunikation-einrichten.html (visited on 2021-11-15).

[Ado21] Adobe. *E-signatures & digital signing software | Adobe Sign*. 2021. URL: https://www.adobe.com/sign.html (visited on 2021-11-03).

[air21] airSlate Inc. *eSignature tools for every business | signNow*. 2021. URL: https://www.signnow.com/ (visited on 2021-11-03).

[Aut20a] Autocrypt Team. *Autocrypt Level 1 Specification*. Version 1.1.0. 2020-12-21. URL: https://autocrypt.org/autocrypt-spec-1.1.0.pdf (visited on 2021-11-08).

[Aut20b] Autocrypt Team. *Autocrypt-capable MUAs level 1 implementation status*. 2020-12-20. URL: https://autocrypt.org/dev-status.html (visited on 2021-11-23).

[Bar+05] Elaine Barker et al. *Recommendation for Key Management – Part 2: Best Practices for Key Management Organization*. NIST Special Publication NIST Special Publication 800-57 Part 2. Gaithersburg, MD: National Institute of Standards and Technology, 2005-08-01, p. 15. 80 pp. DOI: 10.6028/NIST.SP.800-57p2. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p2.pdf (visited on 2021-11-06).

[Ben21a] Sergio Benitez. *Changelog for Version 0.5.0-rc.1 (Jun 09, 2021)*. Version 3b129343e9e68d249b92652e7f7680cefde194a0. 2021-06-18. URL: https://github.com/SergioBenitez/Rocket/blob/v0.5-rc/CHANGELOG.md (visited on 2021-11-21).

[Ben21b] Sergio Benitez. *Configuration - Rocket Programming Guide*. 2021. URL: https://rocket.rs/v0.4/guide/configuration/#configuration (visited on 2021-10-22).

[Ben21c] Sergio Benitez. *Rocket - Simple, Fast, Type-Safe Web Framework for Rust*. 2021. URL: https://rocket.rs/v0.4/ (visited on 2021-10-22).

[BKA16] BKA - Bundeskriminalamt - Abteilung Schwere und Organisierte Kriminalität- SO 31 Auswertung Wirtschaftskriminalität. *Warnhinweis CEO-Fraud*. German. 2016-02-29. URL: https://www.wirtschaftsschutz.info/DE/Themen/Wirtschaftskriminalitaet/PDFCEOFraudbka.pdf?__blob=publicationFile&v=7 (visited on 2021-10-20).

[BMI21] Federal Ministry of the Interior, Building and Community. *Personalausweisportal - Homepage*. English. 2021. URL: https://www.personalausweisportal.de/Webs/PA/EN/home/home-node.html (visited on 2021-11-08).

[Böc21] Hanno Böck. *Das Ende der alten PGP-Keyserver*. German. 2021-06-25. URL: https://www.golem.de/news/sks-das-ende-der-alten-pgp-keyserver-2106-157613.html (visited on 2021-11-08).

[BSI19] BSI - Bundesamt für Sicherheit in der Informationstechnik. *BSI-Projekt - Weiterentwicklung von Mailvelope*. German. 2019. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Freie-Software/E-Mail-Verschluesselung/Mailvelope/mailvelope_node.html (visited on 2021-11-15).

[BSI20] BSI - Bundesamt für Sicherheit in der Informationstechnik. *BSI - Bundesamt für Sicherheit in der Informationstechnik*. 2020. URL: https://www.bsi.bund.de/ (visited on 2020-07-31).

[BSI21] BSI - Bundesamt für Sicherheit in der Informationstechnik. *BSI-Projekt "EasyGPG": E-Mail Verschlüsselung vereinfachen*. German. 2021. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Freie-Software/E-Mail-Verschluesselung/EasyGPG/easygpg.html (visited on 2021-11-08).

[But16] Bart Butler. *ProtonMail now the maintainer of OpenPGPjs email encryption library*. Ed. by ProtonMail Technologies AG. 2016-08-02. URL: https://protonmail.com/blog/openpgpjs-email-encryption/ (visited on 2021-11-20).

[Cip20] Ciphermail B.V. *CipherMail Email Encryption Gateway*. 2020. URL: https://www.ciphermail.com/gateway.html (visited on 2021-11-08).

[Cit21] Citrix Systems Inc. *RightSignature.com Home - Citrix*. 2021. URL: https://rightsignature.com/ (visited on 2021-11-03).

[Ded21] Roman Dedenok. *Email spoofing: how attackers impersonate legitimate senders*. Ed. by Securelist by Kaspersky. 2021-06-03. URL: https://securelist.com/email-spoofing-types/102703/ (visited on 2021-10-20).

[DFN21] Deutsches Forschungsnetz e.V. *DFN-AAI.* 2021-07-13. URL: `https://doku.tid.dfn.de/de:dfnaai:start` (visited on 2021-11-18).

[Dic02] Oxford Dictionaries. *two-person rule.* 2002. DOI: `10.1093/acref/9780199891580.013.8536`. URL: `https://www.oxfordreference.com/view/10.1093/acref/9780199891580.001.0001/acref-9780199891580-e-8536` (visited on 2021-11-06).

[DMA20] DMARC360. *Global DMARC Adoption Rate Report - Banking Sector Edition (January - June 2020).* Research rep. DMARC360, 2020. URL: `https://www.dmarc360.com/images/2020-DMARC-Adoption-Report--Banking-Sector-Edition-Jan-June.pdf` (visited on 2021-10-20).

[DMA21] DMARC360. *Global Banks DMARC Adoption Rate.* 2021-09-01. URL: `https://www.dmarc360.com/dmarc-adoption-rate` (visited on 2021-10-20).

[Doc21a] Docker Inc. *Docker Hub Container Image Library.* 2021. URL: `https://hub.docker.com/` (visited on 2021-11-14).

[Doc21b] Docker Inc. *Empowering App Development for Developers | Docker.* 2021. URL: `https://www.docker.com/` (visited on 2021-10-22).

[Doc21c] Docker Inc. *Overview of Docker Compose | Docker Documentation.* 2021. URL: `https://docs.docker.com/compose/` (visited on 2021-10-22).

[Doc21d] DocuSign Inc. *DocuSign | No. 1 Electronic Signature and Agreement Cloud.* 2021. URL: `https://www.docusign.com/` (visited on 2021-11-03).

[ecs21] ecsec GmbH. *SkIDentity - Sicherer Identitätsnachweis im Netz.* 2021. URL: `https://www.skidentity.de/` (visited on 2021-11-20).

[Edg19] Jake Edge. *OpenPGP certificate flooding.* 2019-07-02. URL: `https://lwn.net/Articles/792366/` (visited on 2021-11-10).

[ENI21] ENISA - European Union Agency for Cybersecurity. *Understanding the increase in Supply Chain Security Attacks.* 2021-07-29. URL: `https://www.enisa.europa.eu/news/enisa-news/understanding-the-increase-in-supply-chain-security-attacks` (visited on 2021-11-14).

[Esc21] Erik Escher. *Git-Anon: Anonymous Git with Signatures.* English. Research rep. Version `491f3896860943fca4208147abecfb9337180164`. Hochschule Ravensburg-Weingarten, 2021-03-15. URL: `https://github.com/erikescher/git-anon/blob/master/project-report.pdf` (visited on 2021-10-20).

[Ext21] Exterro Inc. *Legal Hold (Litigation Hold) - The Basics of E-Discovery.* English. Version 2nd Edition. 2021. URL: `https://www.exterro.com/basics-of-e-discovery/legal-hold` (visited on 2021-11-09).

[FID21] FIDO Alliance. *FIDO Alliance- Open Authentication Standards More Secure than Passwords.* English. 2021. URL: `https://fidoalliance.org/` (visited on 2021-11-17).

[Fis19] Dennis Fisher. *OpenPGP Certificate Attack Worries Experts.* Ed. by Duo Security. 2019-07-01. URL: `https://duo.com/decipher/openpgp-certificate-attack-worries-experts` (visited on 2021-11-04).

[g1020] g10 Code GmbH. *The GNU Privacy Guard.* 2020-07-09. URL: `https://gnupg.org` (visited on 2020-07-30).

[GDPR] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).* English (many translations available). Official Journal of the European Union, L119/1 Volume 59 on 2016-05-04. ISSN 1977-0677. 2016-04-27. URL: `https://eur-lex.europa.eu/eli/reg/2016/679/oj` (visited on 2021-11-15).

[Gil19a] Daniel Kahn Gillmor. *Abuse-Resistant OpenPGP Keystores.* Internet-Draft draft-dkg-openpgp-abuse-resistant-keystore-04. Work in Progress. Internet Engineering Task Force, 2019-08. 58 pp. URL: `https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-abuse-resistant-keystore-04`.

[Gil19b] Daniel Kahn Gilmore. *OpenPGP Certificate Flooding.* 2019-06-28. URL: `https://dkg.fifthhorseman.net/blog/openpgp-certificate-flooding.html` (visited on 2021-11-04).

[Git21] Gitlab B.V. *GitLab as OpenID Connect identity provider.* 2021-09-02. URL: `https://docs.gitlab.com/ee/integration/openid_connect_provider.html` (visited on 2021-11-04).

[Gle93] Ian Saffer Glenn Clark. “Trademark "PGP".” 74467812. INC CA. Registration date: 1995-08-29. Status date: 2015-05-11. Registration Number: 1914615. 1993-12-09. URL: `https://trademarks.justia.com/744/67/pgp-74467812.html` (visited on 2021-11-01).

[GNO21] GNOME Project. *GNOME Evolution.* 2021-01-27. URL: `https://wiki.gnome.org/Apps/Evolution` (visited on 2021-11-19).

[goe21] go.eIDAS e.V. *go.eIDAS.* 2021. URL: `https://go.eid.as/` (visited on 2021-11-20).

[Gov20a] Governikus GmbH & Co. KG. *Beglaubigung OpenPGP-Schlüssel.* 2020. URL: `https://pgp.governikus.de/pgp/` (visited on 2020-07-31).

[Gov20b] Governikus GmbH & Co. KG. *Governikus KG.* 2020. URL: `https://www.governikus.de/` (visited on 2020-07-31).

[GPL] Free Software Foundation Inc. *GNU General Public License.* Version 3. Free Software Foundation, 2007-06-29. URL: `http://www.gnu.org/licenses/gpl.html`.

[Gre21] Andy Greenberg. *Hacker Lexicon: What Is a Supply Chain Attack?* English. Ed. by WIRED. 2021-05-31. URL: `https://www.wired.com/story/hacker-lexicon-what-is-a-supply-chain-attack/` (visited on 2021-11-14).

[Han19] Robert J. Hansen. *SKS Keyserver Network Under Attack.* 2019-06-29. URL: `https://gist.github.com/rjhansen/67ab921ffb4084c865b3618d6955275f` (visited on 2021-11-04).

[Hui+21] Daniel Huigens et al. *JSDoc: Global - Function "generateKey".* Version 5.0.0. 2021-09-02. URL: `https://openpgpjs.org/openpgpjs/global.html#generateKey` (visited on 2021-11-10).

[Int21] Intevation GmbH. *Gpg4win - Secure email and file encryption with GnuPG for Windows.* 2021. URL: `https://www.gpg4win.org/` (visited on 2021-11-19).

[Ion19] Ionos Inc. *Email archiving: what all companies should know.* 2019-10-23. URL: `https://www.ionos.com/digitalguide/e-mail/technical-matters/email-archiving-laws-and-practices-you-should-know/` (visited on 2021-11-09).

[Jat21] Jatheon Technologies. *All You Need to Know About Email Compliance.* 2021-03-11. URL: `https://jatheon.com/blog/email-compliance-email-archiving/` (visited on 2021-11-09).

[Jog21] Bharat Jogi. *21Nails: Multiple Critical Vulnerabilities in Exim Mail Server.* en. 2021-05-04. URL: `https://blog.qualys.com/vulnerabilities-research/2021/05/04/21nails-multiple-vulnerabilities-in-exim-mail-server` (visited on 2021-05-05).

[KDE21a] KDE e.V. *KDE Gear 21.04.* 2021-04-22. URL: `https://kde.org/announcements/gear/21.04/` (visited on 2021-11-23).

[KDE21b] KDE e.V. *KMail | KContact Suite- The Powerful PIM Solution.* 2021. URL: `https://kontact.kde.org/components/kmail.html` (visited on 2021-11-19).

[Key21a] Keybase. *Keybase - End-to-end encryption for things that matter. Keybase is secure messaging and file-sharing.* 2021. URL: `https://keybase.io/` (visited on 2021-11-16).

[Key21b] KeyCloak Project. *Server Administration Guide - Identity Brokering.* 2021-08-20. URL: `https://www.keycloak.org/docs/latest/server_admin/#_identity_broker` (visited on 2021-11-05).

[Koc+20] Werner Koch et al. *OpenPGP Message Format.* Internet-Draft draft-ietf-openpgp-rfc4880bis-10. Work in Progress. Internet Engineering Task Force, 2020-08. 111 pp. URL: `https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-rfc4880bis-10`.

[Koc06] Werner Koch. *Public Key Association.* German. g10 Code. 2006-02-16. URL: https://www.g10code.com/docs/pka-intro.de.pdf (visited on 2021-11-08).

[Koc21] Werner Koch. *OpenPGP Web Key Directory.* Internet-Draft draft-koch-openpgp-webkey-service-12. Work in Progress. Internet Engineering Task Force, 2021-05. 18 pp. URL: https://datatracker.ietf.org/doc/html/draft-koch-openpgp-webkey-service-12.

[Lee19] Micah Lee. *The Death of SKS PGP Keyservers, and How First Look Media is Handling It.* 2019-08-20. URL: https://code.firstlook.media/the-death-of-sks-pgp-keyservers-and-how-first-look-media-is-handling-it (visited on 2021-11-08).

[Mai21] Mailvelope GmbH. *Communicating securely with Mailvelope.* 2021. URL: https://mailvelope.com/en/ (visited on 2021-11-15).

[Mar20] Casey Marshall. *New features in 2.1.0.* 2020-12-10. URL: https://github.com/hockeypuck/hockeypuck/releases/tag/2.1.0 (visited on 2021-11-08).

[Mar21] Casey Marshall. *Hockeypuck OpenPGP Public Keyserver.* 2021-03-18. URL: https://hockeypuck.io/ (visited on 2021-11-08).

[Mat21] Matrix.org Foundation. *Matrix.org - An open network for secure, decentralized communication.* 2021-10-20. URL: https://matrix.org/ (visited on 2021-10-20).

[McC18] Miles McCain. *Keylist RFC: Distributing OpenPGP Keys with Signed Keylist Subscriptions.* 2018-08-24. URL: https://code.firstlook.media/keylist-rfc-explainer (visited on 2021-10-20).

[MCS21] Casey Marshall, Paul Collins, and Joel Sing. *Hockeypuck. Hockeypuck is an OpenPGP public keyserver.* Version Commit: 58d985e7d872fd95d3bf3f2c628a13fc502c76e7. 2021-10-14. URL: https://github.com/hockeypuck/hockeypuck (visited on 2021-11-08).

[Mic20] Microsoft Corporation. *Microsoft: Included CA Certificate List.* Ed. by Common CA Database (operated by Mozilla Corporation). 468 records. 2020-10-20. URL: https://ccadb-public.secure.force.com/microsoft/IncludedCACertificateReportForMSFT (visited on 2021-10-20).

[Mic21] Nat Welch Micah Lee Sam Couch. *GPG Sync.* 2021-02-08. URL: https://github.com/firstlookmedia/gpgsync (visited on 2021-11-01).

[MLW19] R. Miles McCain, Micah Lee, and Nat Welch. *Distributing OpenPGP Key Fingerprints with Signed Keylist Subscriptions.* Internet-Draft draft-mccain-keylist-05. Work in Progress. Internet Engineering Task Force, 2019-09. 11 pp. URL: https://datatracker.ietf.org/doc/html/draft-mccain-keylist-05.

[Moz20a] Mozilla Corporation. *Mozilla: CA Certificates In Firefox.* Ed. by Common CA Database (operated by Mozilla Corporation). 131 records. 2020-10-20. URL: https://ccadb-public.secure.force.com/mozilla/CACertificatesInFirefoxReport (visited on 2021-10-20).

[Moz20b] Mozilla Corporation. *Oxidation.* 2020-11-17. URL: https://wiki.mozilla.org/index.php?title=Oxidation&oldid=1232260 (visited on 2021-11-21).

[MS21] Microsoft Corporation. *CVE-2021-26855. Microsoft Exchange Server Remote Code Execution Vulnerability.* Ed. by National Institue of Standards and Technology. 2021-05-21. URL: https://nvd.nist.gov/vuln/detail/CVE-2021-26855 (visited on 2021-10-15).

[MX 21] MX ToolBox Inc. *MxToolbox Report on DMARC Adoption by the Fortune 500 and Alexa 1000.* 2021. URL: https://mxtoolbox.com/c/landing/tl/fortune-alexa-dmarc (visited on 2021-10-20).

[Naj20] Armen Najarian. *DMARC Adoption Slows, 80 percent of Fortune 500 Email Senders Remain Unauthenticated.* 2020-08-17. URL: https://www.agari.com/email-security-blog/h2-2020-email-trends-report-dmarc/ (visited on 2021-10-20).

[NCS18] NCSC - National Cyber Security Centre (of the United Kingdom). *Supply chain security guidance.* Version 1.0. 2018-01-28. URL: https://www.ncsc.gov.uk/collection/supply-chain-security (visited on 2021-11-14).

[OAS05] OASIS - Organization for the Advancement of Structured Information Standards. *Security Assertion Markup Language (SAML) v2.0.* English. Ed. by John Highes et al. 2005-03-15. URL: http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip (visited on 2021-11-10).

[Ope19] OpenPGP-CA Project. *License for OPENPGP-CA. GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007.* 2019-04-14. URL: https://gitlab.com/openpgp-ca/openpgp-ca/-/blob/main/COPYING (visited on 2021-11-14).

[Ope21a] OpenJS Foundation. *Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS.* 2021. URL: https://www.electronjs.org/ (visited on 2021-11-19).

[Ope21b] OpenPGP-CA Project. *Bridging OpenPGP CA instances.* 2021. URL: https://openpgp-ca.org/doc/bridging/ (visited on 2021-10-22).

[Ope21c] OpenPGP-CA Project. *Getting started.* 2021. URL: https://openpgp-ca.org/doc/start/ (visited on 2021-10-22).

[Ope21d] OpenPGP-CA Project. *Gitlab.com Group: openpgp-ca.* Gitlab Group ID: 6970752. 2021. URL: https://gitlab.com/openpgp-ca (visited on 2021-11-14).

[Ope21e] OpenPGP-CA Project. *Importing user keys.* 2021. URL: https://openpgp-ca.org/doc/keys-import/ (visited on 2021-11-09).

[Ope21f] OpenPGP-CA Project. *Inspecting OpenPGP CAs state.* 2021. URL: https://openpgp-ca.org/doc/keys-inspect/ (visited on 2021-10-22).

[Ope21g] OpenPGP-CA Project. *OpenPGP CA.* 2021. URL: https://openpgp-ca.org/ (visited on 2021-10-20).

[Ope21h] OpenPGP-CA Project. *Publishing keys as WKD or Keylist.* 2021. URL: https://openpgp-ca.org/doc/keys-publish/ (visited on 2021-11-09).

[PEP21] PEP Foundation. *PEP Foundation - Home.* 2021. URL: https://pep.foundation/ (visited on 2021-11-21).

[Pos21] Posteo e.K. *How do I install end-to-end encryption for the Posteo webmail interface with Mailvelope (PGP)?* 2021. URL: https://posteo.de/en/help/how-do-i-set-up-end-to-end-encryption-in-the-browser (visited on 2021-11-15).

[Pro21a] Proton Technologies AG. *How to use PGP.* 2021. URL: https://protonmail.com/support/knowledge-base/how-to-use-pgp/ (visited on 2021-11-15).

[Pro21b] Proton Technologies AG. *OpenPGP.js | OpenPGP JavaScript Implementation.* 2021. URL: https://openpgpjs.org/ (visited on 2021-11-19).

[RFC1034] P. Mockapetris. *Domain names - concepts and facilities.* RFC 1034. 1987-11. DOI: 10.17487/RFC1034. URL: https://rfc-editor.org/rfc/rfc1034.txt.

[RFC1035] P. Mockapetris. *Domain names - implementation and specification.* RFC 1035. 1987-11. DOI: 10.17487/RFC1035. URL: https://rfc-editor.org/rfc/rfc1035.txt.

[RFC4033] Scott Rose et al. *DNS Security Introduction and Requirements.* RFC 4033. 2005-03. DOI: 10.17487/RFC4033. URL: https://rfc-editor.org/rfc/rfc4033.txt.

[RFC4398] Simon Josefsson. *Storing Certificates in the Domain Name System (DNS).* RFC 4398. 2006-03. DOI: 10.17487/RFC4398. URL: https://rfc-editor.org/rfc/rfc4398.txt.

[RFC4511] Jim Sermersheim. *Lightweight Directory Access Protocol (LDAP): The Protocol.* RFC 4511. 2006-06. DOI: 10.17487/RFC4511. URL: https://rfc-editor.org/rfc/rfc4511.txt.

[RFC4532] Kurt Zeilenga. *Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates.* RFC 4523. 2006-06. DOI: 10.17487/RFC4523. URL: https://rfc-editor.org/rfc/rfc4523.txt.

[RFC4880] Hal Finney et al. *OpenPGP Message Format.* RFC 4880. 2007-11. DOI: 10.17487/RFC4880. URL: https://rfc-editor.org/rfc/rfc4880.txt (visited on 2020-07-30).

[RFC5280] Sharon Boeyen et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. 2008-05. DOI: 10.17487/RFC5280. URL: https://rfc-editor.org/rfc/rfc5280.txt.

[RFC5321] Dr. John C. Klensin. *Simple Mail Transfer Protocol*. RFC 5321. 2008-10. DOI: 10.17487/RFC5321. URL: https://rfc-editor.org/rfc/rfc5321.txt.

[RFC6376] D. Crocker, T. Hansen, and M. Kucherawy. *DomainKeys Identified Mail (DKIM) Signatures*. STD 76. http://www.rfc-editor.org/rfc/rfc6376.txt. RFC Editor, 2011-09. URL: http://www.rfc-editor.org/rfc/rfc6376.txt.

[RFC6960] Stefan Santesson et al. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 6960. 2013-06. DOI: 10.17487/RFC6960. URL: https://rfc-editor.org/rfc/rfc6960.txt.

[RFC7208] S. Kitterman. *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*. RFC 7208. http://www.rfc-editor.org/rfc/rfc7208.txt. RFC Editor, 2014-04. URL: http://www.rfc-editor.org/rfc/rfc7208.txt.

[RFC7489] M. Kucherawy and E. Zwicky. *Domain-based Message Authentication, Reporting, and Conformance (DMARC)*. RFC 7489. RFC Editor, 2015-03.

[RFC7519] Michael Jones, John Bradley, and Nat Sakimura. *JSON Web Token (JWT)*. RFC 7519. 2015-05. DOI: 10.17487/RFC7519. URL: https://rfc-editor.org/rfc/rfc7519.txt.

[RFC7671] Viktor Dukhovni and Wes Hardaker. *The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance*. RFC 7671. 2015-10. DOI: 10.17487/RFC7671. URL: https://rfc-editor.org/rfc/rfc7671.txt.

[RFC7929] Paul Wouters. *DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP*. RFC 7929. 2016-08. DOI: 10.17487/RFC7929. URL: https://rfc-editor.org/rfc/rfc7929.txt.

[RFC8446] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. 2018-08. DOI: 10.17487/RFC8446. URL: https://rfc-editor.org/rfc/rfc8446.txt.

[RFC8551] Jim Schaad, Blake C. Ramsdell, and Sean Turner. *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification*. RFC 8551. 2019-04. DOI: 10.17487/RFC8551. URL: https://rfc-editor.org/rfc/rfc8551.txt.

[Rou21] Roundcube.net. *Roundcube - Free and Open Source Webmail Software*. 2021. URL: https://roundcube.net/ (visited on 2021-11-15).

[Rus21a] Rust Foundation. *crates.io: Rust Package Registry*. 2021. URL: https://crates.io/ (visited on 2021-11-14).

[Rus21b] Rust Team. *Production users - Rust Programming Language*. 2021. URL: `https://www.rust-lang.org/production/users` (visited on 2021-11-21).

[Rus21c] Rust Team. *Rust Programming Language*. 2021. URL: `https://www.rust-lang.org/` (visited on 2021-11-21).

[Sak+14] Nat Sakimura et al. *OpenID Connect Core 1.0*. English. Version incorporating errata set 1. OpenID Foundation, 2014-11-08. URL: `https://openid.net/specs/openid-connect-core-1_0.html` (visited on 2021-11-10).

[Sch21] Dennis Schirrmacher. *Jetzt patchen! Kritische Root-Lücken bedrohen Exim-Mail-Server*. de. Heise Medien, 2021-05-05. URL: `https://www.heise.de/news/Jetzt-patchen-Kritische-Root-Luecken-bedrohen-Exim-Mail-Server-6036724.html` (visited on 2021-05-05).

[Sec20] Secorio AG. *S/MIME E-Mail certificates for GDPR | immediate issuance*. 2020. URL: `https://secorio.com/en/certificates/smime-email/` (visited on 2021-11-17).

[Sec21] Sectigo Limited. *Sectigo Secure Email Solutions (S/MIME)*. 2021. URL: `https://sectigo.com/ssl-certificates-tls/email-smime-certificate` (visited on 2021-11-17).

[Seq21a] Sequoia Project. *Documentation for "sq" - Subcommand key attest-certifications*. 2021. URL: `https://docs.sequoia-pgp.org/sq/index.html#subcommand-key-attest-certifications` (visited on 2021-11-05).

[Seq21b] Sequoia Project. *Gitlab.com Group: sequoia-pgp*. Gitlab Group ID: 2151052. 2021. URL: `https://gitlab.com/sequoia-pgp` (visited on 2021-11-14).

[Seq21c] Sequoia Project. *Hagrid as in, "keeper of keys". Verifying OpenPGP key-server, written in Rust, running on https://keys.openpgp.org*. Version Commit: `90356ddb282874d5be5f7406e49e203f3caf437c`. 2021-07-20. URL: `https://gitlab.com/hagrid-keyserver/hagrid` (visited on 2021-11-08).

[Seq21d] Sequoia Project. *keys.openpgp.org - FAQ*. Version Hagrid version 1.1.0 . Commit: `58585dd41f52e391177c32856a418c2c6bd6d86f`. 2021-06-24. URL: `https://keys.openpgp.org/about/faq` (visited on 2021-11-08).

[Seq21e] Sequoia Project. *keys.openpgp.org - Overview*. Version Hagrid version 1.1.0 . Commit: `58585dd41f52e391177c32856a418c2c6bd6d86f`. 2021-06-24. URL: `https://keys.openpgp.org/about` (visited on 2021-11-08).

[Seq21f] Sequoia Project. *keys.openpgp.org - Privacy Policy*. Version Hagrid version 1.1.0 . Commit: `58585dd41f52e391177c32856a418c2c6bd6d86f`. 2021-06-24. URL: `https://keys.openpgp.org/about/privacy` (visited on 2021-11-08).

[Seq21g] Sequoia Project. *keys.openpgp.org - Usage*. Version Hagrid version 1.1.0 . Commit: `58585dd41f52e391177c32856a418c2c6bd6d86f`. 2021-06-24. URL: `https://keys.openpgp.org/about/usage` (visited on 2021-11-08).

[Seq21h] Sequoia Project. *License for Sequoia. GNU LIBRARY GENERAL PUBLIC LICENSE Version 2, June 1991.* 2021-10-18. URL: `https://gitlab.com/sequoia-pgp/sequoia/-/blob/main/LICENSE.txt` (visited on 2021-11-14).

[Seq21i] Sequoia Project. *sequoia-octopus-librnp - A Sequoia-based OpenPGP Backend for Thunderbird*. Version 1.1.0. 2021-09-17. URL: `https://crates.io/crates/sequoia-octopus-librnp` (visited on 2021-11-19).

[Seq21j] Sequoia Project. *Sequoia-PGP - A New OpenPGP Libary*. 2021. URL: `https://sequoia-pgp.org/` (visited on 2021-11-05).

[Seq21k] Sequoia Project. *Sequoia-PGP - Contribute*. 2021. URL: `https://sequoia-pgp.org/contribute/` (visited on 2021-11-21).

[SER21] SERVERMX (Coranto Informatica di Antonio Cordeddu). *How to use roundcube.servermx.com to exchange encrypted emails using PGP*. 2021. URL: `https://www.servermx.com/en/help/documentation/pgp-on-roundcube/` (visited on 2021-11-15).

[Sha79] Adi Shamir. "How to Share a Secret." In: *Commun. ACM* 22.11 (1979-11), pp. 612–613. ISSN: 0001-0782. DOI: `10.1145/359168.359176`. URL: `https://doi.org/10.1145/359168.359176` (visited on 2021-11-06).

[Sym11a] Symantec Corporation. *PGP Global Directory*. 2011. URL: `https://keyserver.pgp.com/vkd/GetWelcomeScreen.event` (visited on 2021-11-08).

[Sym11b] Symantec Corporation. *PGP Global Directory - Frequently Asked Questions (FAQ)*. 2011. URL: `https://keyserver.pgp.com/vkd/VKDHelpPGPCom.html` (visited on 2021-11-08).

[Sym11c] Symantec Corporation. *PGP Global Directory - Key Verification Policy*. 2011. URL: `https://keyserver.pgp.com/vkd/VKDVerificationPGPCom.html` (visited on 2021-11-08).

[Tel18] Telekom Deutschland GmbH. *E-mail Encryption Gateway. Sicherer E-mail Verkehr*. German. 2018-04-13. URL: `https://geschaeftskunden.telekom.de/security/e-mail-encryption-gateway-whitepaper` (visited on 2021-11-08).

[Tot21] Totemo AG. *External Email Encryption*. 2021. URL: `https://www.totemo.com/en/products/email-encryption/external-encryption` (visited on 2021-11-08).

[Tru17] Trusted Domains Project. *Alexa Top Sites*. 2017. URL: `https://dmarc.org/stats/alexa-top-sites/` (visited on 2021-10-20).

[Ver19] Matthew Vernhout. *Global DMARC Adoption 2019*. Research rep. 250ok, 2019. URL: https://s3.amazonaws.com/250ok-wordpress/wp-content/uploads/2019/07/09140509/Global-DMARC-Adoption-2019.pdf (visited on 2021-10-20).

[W3C19] World Wide Web Consortium. *WebAssembly Core Specification*. W3C Recommendation. Version 1. 2019-12-05. URL: https://www.w3.org/TR/2019/REC-wasm-core-1-20191205/ (visited on 2021-11-18).

[Web21] Kristian Fikerstrand Webs. *SKS Keyservers - Homepage*. 2021-06-21. URL: https://sks-keyservers.net/ (visited on 2021-11-08).

[Wik20] Wikipedia contributors. *Right to be forgotten*. In: *en.wikipedia.org/*. Page Version ID: 979590717. 2020-09-21. URL: https://en.wikipedia.org/w/index.php?title=Right_to_be_forgotten&oldid=979590717 (visited on 2020-10-09).

[Wik21a] Wikipedia contributors. *2021 Microsoft Exchange Server data breach — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=2021_Microsoft_Exchange_Server_data_breach&oldid=1042157523. 2021. (Visited on 2021-10-15).

[Wik21b] Wikipedia contributors. *Aufbewahrungsfrist — Wikipedia, die freie Enzyklopädie*. German. 2021-07-24. URL: https://de.wikipedia.org/w/index.php?title=Aufbewahrungsfrist&oldid=214156929 (visited on 2021-11-09).

[Wik21c] Wikipedia contributors. *Broadcom Inc. — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Broadcom_Inc.&oldid=1051143690. 2021. (Visited on 2021-11-01).

[Wik21d] Wikipedia contributors. *Business email compromise — Wikipedia, The Free Encyclopedia*. 2021. URL: https://en.wikipedia.org/w/index.php?title=Business_email_compromise&oldid=1042510209 (visited on 2021-10-20).

[Wik21e] Wikipedia contributors. *CA Technologies — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=CA_Technologies&oldid=1042226176. 2021. (Visited on 2021-11-01).

[Wik21f] Wikipedia contributors. *Certificate authority — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-October-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Certificate_authority&oldid=1050666323 (visited on 2021-10-20).

[Wik21g] Wikipedia contributors. *Email spoofing — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Email_spoofing&oldid=1042510979. 2021. (Visited on 2021-10-20).

[Wik21h] Wikipedia contributors. *German identity card — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=German_identity_card&oldid=1048379272. 2021. (Visited on 2021-11-08).

[Wik21i] Wikipedia contributors. *Secret sharing — Wikipedia, The Free Encyclopedia.* 2021. URL: https://en.wikipedia.org/w/index.php?title=Secret_sharing&oldid=1041115340 (visited on 2021-11-06).

[Wik21j] Wikipedia contributors. *Security Assertion Markup Language — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Security_Assertion_Markup_Language&oldid=1053574871. 2021. (Visited on 2021-11-10).

[Wik21k] Wikipedia contributors. *Two-man rule — Wikipedia, The Free Encyclopedia.* 2021. URL: https://en.wikipedia.org/w/index.php?title=Two-man_rule&oldid=1043577310 (visited on 2021-11-06).

[Wik21l] Wikipedia contributors. *X.509 — Wikipedia, The Free Encyclopedia.* 2021. URL: https://en.wikipedia.org/w/index.php?title=X.509&oldid=1047480104 (visited on 2021-11-15).

[Wir21] Wire Swiss GmbH. *The most secure collaboration platform - Wire.* 2021. URL: https://wire.com/en/ (visited on 2021-11-09).

[Yar21] Yarmo Mackenbach. *Keyoxide docs.* 2021. URL: https://docs.keyoxide.org/getting-started/ (visited on 2021-11-16).

[You21] Evan You. *Vue.js - The Progressive JavaScript Framework.* 2021. URL: https://vuejs.org/ (visited on 2021-11-20).

[Yub21] Yubico. *OpenPGP Attestation.* 2021. URL: https://developers.yubico.com/PGP/Attestation.html (visited on 2021-11-03).

[Zer21] Zertificon Solutions GmbH. *Z1 SecureMail Gateway - Enterprise Email Encryption.* 2021. URL: https://www.zertificon.com/en/solutions/email-encryption-gateway (visited on 2021-11-08).

[Zoh21] Zoho Corporation Pvt. Ltd. *Zoho Sign - Electronic Signature Software | Digital Signature Software for Business Signatories.* 2021. URL: https://www.zoho.com/sign/ (visited on 2021-11-03).

[Zuc21] Simon Zuckerbraun. *ProxyToken: An Authentication Bypass in Microsoft Exchange Server.* CVE-2021-33766. Zero Day Initiative. 2021-08-30. URL: https://www.zerodayinitiative.com/blog/2021/8/30/proxytoken-an-authentication-bypass-in-microsoft-exchange-server (visited on 2021-10-15).