PADERBORN UNIVERSITY
The University for the Information Society
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group System Security

Bachelor's Thesis

Submitted to the System Security Research Group
in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science

Web Key Directory and other key exchange methods for OpenPGP

Philipp Michael Breuch

Supervisors: Prof. Dr.-Ing. Juraj Somorovsky
Dipl.-Math. Marcus Brinkmann

Paderborn, July 7, 2022

Abstract

The cryptographic protocol OpenPGP exists since 1991 and is used for the encryption and signing of e-mails and data. OpenPGP uses public key cryptography and requires that key authenticity is verified in a secure manner. Historically this is done via key servers and a decentralized trust model, the Web of Trust.

In this thesis, we describe and analyze the OpenPGP key exchange method "Web Key Directory". We provide security definitions for OpenPGP key exchange methods. Based on these definitions, we evaluate whether the Web Key Directory specification and its reference implementation are secure.

We find inconsistencies and specification gaps in the Web Key Directory specification draft. We reveal that the main assumption of the Web Key Directory Update Protocol is too vague. We describe several scenarios and interpretations of the main assumption and analyze them. We can show that the Web Key Directory Update Protocol is vulnerable in multiple scenarios and interpretations.

Furthermore, we find errors in the reference implementation. We could utilize errors to describe an attack on the reference implementation with almost no assumptions. It allows an attacker to illegitimate publish OpenPGP keys for any e-mail address for any domain of a Web Key Directory provider.

Official Declaration

I hereby declare that I prepared this thesis entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

Eidesstattliche Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum	Philipp Michael Breuch

Contents

1 Introduction

The majority of today's communication is exchanged via electronic communication. Initially designed in the 1970s, e-mails are still a ubiquitous medium for this purpose [59]. E-mails hold no necessary provisions to ensure secure communication, such as guarantees about message integrity, authenticity, and confidentiality [64]. Neither the message itself nor the transmission was secured [9, 45]. Although these problems are well-known and improvements have been made, securing e-mails is still challenging; e-mail infrastructure is highly decentralized and can contain many services.

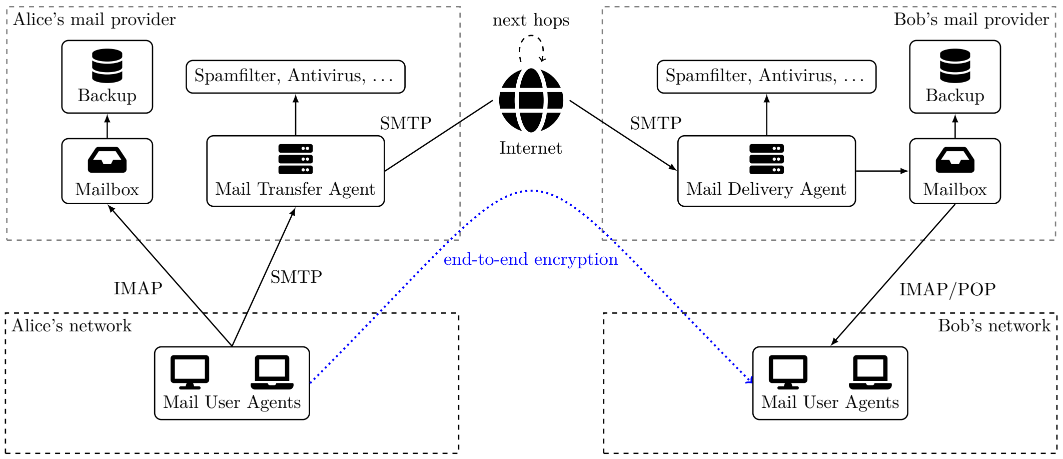


Figure 1.1: System image of components of a typical e-mail infrastructure with two mail providers. (Source: see Section 7.2 (Acknowledgements))

In Figure 1.1, we see an e-mail infrastructure for two users, Alice and Bob, who use different e-mail providers. The black arrows indicate the path an e-mail from Alice to Bob might take. Alice and Bob interact with the e-mail system via their Mail User Agents (MUA), like Mozilla Thunderbird or Microsoft Outlook, where they can access, write, and manage their e-mails. Both have a mailbox wherein their e-mails are organized. Each mailbox is present at the MUA and the provider's server. If both mailboxes should be kept in sync, the Internet Message Access Protocol (IMAP) [8] protocol can be used. If new e-mails should only be downloaded to the local mailbox, the Post Office Protocol (POP) [39] can be used. Both protocols are the most relevant ones for accessing mailboxes. To reduce the risk of data loss, providers might backup mailboxes. In addition, e-mails processed at their systems might also be scanned for spam or viruses. To make communication possible, e-mails must be transmitted from one user to another user via different e-mail providers. If Alice wants to write an e-mail to Bob, her MUA transmits the e-mail to her provider's Mail Transfer Agent (MTA), which is responsible for transmitting e-mails to the next responsible MTA. The e-mail might pass several different MTAs and providers before it arrives at Bob's provider's Mail Delivery Agent (MDA). The MDA delivers the e-mail to Bob's mailbox. The communication between one MTA and another MTA or MDA is done mostly via the Simple Mail Transfer Protocol (SMTP) for mail exchange. Each MUA might check periodically for new arrived e-mails and download them.

One improvement to secure e-mails was the introduction of SSL/TLS to relevant protocols like SMTP, IMAP, and POP [41, 23]. Therefore, the connection between each component in the e-mail system can be secured. As we see, Alice's e-mail is processed at many different components from different entities. The e-mail is present in plain text at every component and can be altered and read. Even if Alice runs her own network, e-mail provider, and knows that everything is properly secured. Once her e-mail is transferred to a different MTA, she simply can not make assumptions about the security of her e-mail. Therefore, securing the transport layer is not sufficient and end-to-end security protocols were introduced.

Phil Zimmermann developed *Pretty Good Privacy* (short: PGP) in 1991. It is a protocol and program to sign, verify, encrypt, and decrypt files including e-mails. Later it was publicly standardized as OpenPGP and expanded in several RFCs [7]. Due to limitations of the early e-mail format, the *Multipurpose Internet Mail Extensions* was introduced. It allows, for example, other text encodings than ASCII and multipart message bodies [19]. With *PGP/MIME*, OpenPGP can be integrated in the MIME structure [15]. Another protocol is the 1996 introduced *Secure/Multipurpose Internet Mail Extensions* (short: S/MIME), with which the MIME structure in e-mails can be signed and encrypted [46].

OpenPGP is based on public key cryptography. Each participant has to generate their own private-public key pair. A user's public key must be distributed to their communication partners. Without proper key validation, no trust could be gained in the distributed key material and no secure communication could be established. An attacker can simply create and distribute keys with a victim's name and e-mail address. Without proper key validation, the attacker's keys might be used for encryption or signature verification instead of the legitimate participant's keys [22].

In this thesis, we will address the end-to-end encryption protocol PGP. Especially, we will be concerned with key exchange methods to distribute and validate PGP keys.

1.1 Key Exchange and Validation Problems

Historically key validation in PGP is done in two ways. First, a public key could be manually validated by its fingerprint. Thereby the fingerprint of the copied key is compared with the fingerprint of the key owner's local key. After that, the key will be signed with the validating person's private key and could be uploaded to a key server to share their key validation with the community.

Second, the validation could be done indirectly via a mechanism called "web of trust" (WoT). Here, the user assigns a trust property to a key, which indicates how much the person identified by the key is trusted to faithfully verify other keys. OpenPGP has six levels of trust: ultimate (reserved for the keyring owner's public keys), full, marginal, untrusted, undefined, and unknown [2]. An unverified key `K` is validated automatically via the web of trust if the following both conditions apply:

1. Key `K` has enough validation signatures[1] by valid keys. This means one of the following must apply:
    - You have signed `K` personally (direct validation)
    - `K` has been signed by one[2] fully trusted and valid key (indirect validation),
    - `K` has been signed by three[2] marginally trusted and valid keys (indirect validation)
2. The length of the path of signed keys from key `K` back to your own key is at most five[2] steps.

We illustrate an example of a web of trust in Figure 1.2. To make the example shorter, we reduce the maximum path length (as shown in enumeration point 2 above) to at most three steps and assume that only two marginal keys as sufficient for indirect validation.

Keys could be transmitted via direct one-to-one communication or via publicly available key servers. These offer the possibility to upload, search, and download other public keys. To make it difficult for censorship to take down public keys, key servers began to automatically synchronize between each other (so called Synchronizing Key Server (SKS)) via the OpenPGP HTTP Keyserver Protocol (HKP). As a consequence, it is nearly impossible to remove an uploaded key [4].

As it turned out, the web of trust and key servers have fundamental problems in their current form. Public (non verifying) key servers do not verify uploaded keys. Everyone can upload keys for any e-mail address, which lead to different attack

[1] A "certification" signature type as per [7, Sec. 5.2.1].

[2] Implementation dependent. Might be lower or higher. The value is the default used by GnuPG.

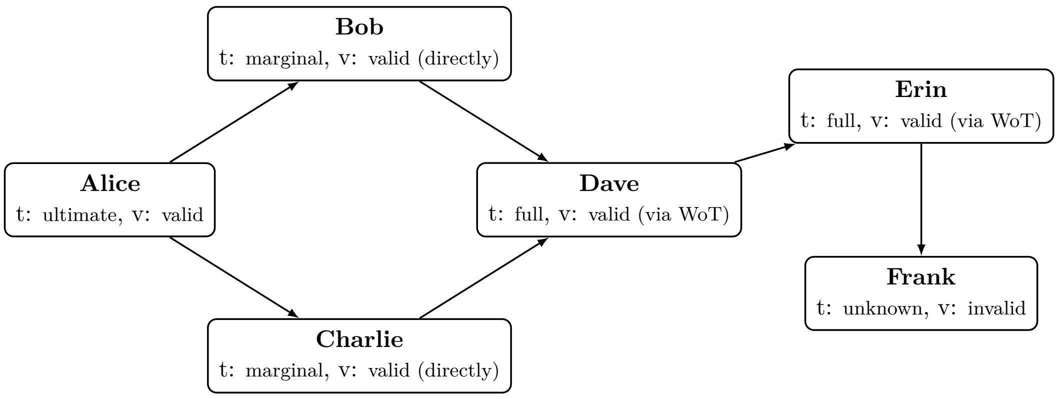

Bob
t: marginal, v: valid (directly)
Erin
t: full, v: valid (via WoT)
Alice
t: ultimate, v: valid
Dave
t: full, v: valid (via WoT)
Frank
t: unknown, v: invalid
Charlie
t: marginal, v: valid (directly)

Figure 1.2: Example of a web of trust. “t:” indicates the trust level as seen from Alice. “v:” indicates whether Alice sees the key as valid. Edges between nodes describe validation signatures. Alice trusts Bob and Charlie marginally. Alice trusts Dave and Erin fully. Alice sees Bob’s and Charlie’s keys as valid because she validated them directly. Alice sees Dave’s and Erin’s keys as valid because of indirect validation through the web of trust. Alice sees Frank’s key as invalid because the length of the path of signed keys from Frank to Alice is four and therefore larger than our (reduced) maximum path length of three.

vectors. The “evil32” attack [28] showed that it is very efficient to generate and upload keys with the same short fingerprint (the last 4 byte of the fingerprint). Users who check only the short fingerprint can not distinguish between the legitimate and the counterfeit key. To make the “web of trust” possible, everyone can sign a public key to express that they have verified it. By massively signing and uploading a victim’s key to a key server, the key size will grow steadily. As a result, a key with too many signatures (the specific amount depends on the used software) can not be processed anymore. Synchronizing key servers are practically only appending key material, which complicates mitigation of such attacks. Not only manipulated keys, but also public keys where users no longer possess the corresponding private key, remain in the system [22]. Ulrich et al. [61] found out that only a fraction of the users of the “web of trust” can profit from it. This is contrasted by the disclosure of personal data and the social graphs of the users, which can be derived from the key signatures [22]. The SKS key server network, which was used as the default key server in GnuPG, is no longer maintained and has been shutdown on 2021-06-21 due to privacy problems and legal problems resulting from the European General Data Protection Regulation [6].

The vast majority of e-mails are not signed or encrypted. Research was therefore also done on the usability of end-to-end encryption including the PGP ecosystem and tools. According to a study from Reuter et al. [52], most users are aware of the problems of unprotected e-mails but do not know or can not use tools to mitigate them. Renaud et al. [47] could not find evidence for non-awareness of basic privacy issues. However, the lack of understanding of e-mail security mechanisms leads to users not even thinking about using end-to-end encryption in the context of e-mails.

1.2 Solution Approaches

Web Key Directory, Autocrypt, and e-mail validating key servers, are relatively new methods. They were introduced to mitigate problems in key exchange, key validation, and usability.

Web Key Directory wants to achieve an easy way to exchange and reach a basic level of key validation of PGP public keys. It is currently in the standardization process. Public keys are therefore provided via a TLS secured HTTP connection to a web server. The responsible web server is found by deriving a well-known path from the e-mail address. It is further possible, but not recommended, to use a foreign web server as a Web Key Directory as a Service (WKDaaS) by indicating it via a DNS entry. With Web Key Service (WKS), a tool and protocol is available to reduce the administrative effort to publish and update keys to the WKD [5, 21, 33].

The Hagrid key server is a free and open source e-mail validating key server running on `keys.openpgp.org`. It tries to solve typical problems which apply to the non-validating key servers like the SKS key server network. The contents of PGP keys are separated into identity information (like the name and e-mail address of its owner) and non-identity information (like technical metadata and the public key material). Identity information is only saved and distributed if the owner explicitly approves that. For this approval process, an e-mail is sent to the key owner's e-mail. Additionally, it is possible to request a deletion. Signatures (like the validation signatures mentioned in the web of trust description above) are also removed from the PGP key before key publication. The Hagrid key server is not part of any SKS key server network and can be used as a Web Key Directory as a Service [43, 40].

Autocrypt is an opportunistic security protocol for e-mail, which tries to simplify key exchange and usability. It is not in a process to become an official standard. PGP public keys are distributed via e-mail. Key management, including key validation is done completely automatic. It uses the cryptographic constructions from PGP but introduces protocol-specific mail headers. The protocol makes no attempt to protect against active attackers [1, 22].

To the best of our knowledge, there is no dedicated scientific security research on Web Key Directory, Autocrypt, and e-mail validating key servers which investigates the above methods in detail. Currently, there is an ongoing bachelor thesis on how usage of WKD can be furthered.[3]

1.3 Contributions

The goal of this thesis is the description and analysis of the Web Key Directory key exchange method. We analyze both the specification and the reference implementation. For the analysis, we define security definitions for OpenPGP key exchange methods and typical operations for those key exchange methods. To the best of our knowledge, this is the first scientific security research on the Web Key Directory key exchange method.

Our results include the finding of inconsistencies and specification gaps in the specification draft. We also analyze Web Key Directory's usage of the SHA-1 hash algorithm but find its usage in Web Key Directory might be secure despite SHA-1 being long deprecated because of its broken collision resistance. Furthermore, the local-part of e-mail addresses is mapped to lowercase by Web Key Directory. According to a small case study, this appears to be relatively unproblematic for bigger providers but could affect specific setups. Furthermore, we see that the Web Key Directory Update Protocol main assumption is to vague and depending on the circumstances an attacker could illegitimately publish keys. The reference implementation has several mistakes and errors. We describe an attack on the reference implementation which utilizes some of those errors. With this attack, an attacker could publish OpenPGP keys for any e-mail address of any domain of a Web Key Directory provider.

1.4 Organization of this Thesis

This thesis is divided into 7 chapters. In Chapter 2, we summarize the background relevant to this thesis. In Chapter 3, we give a brief description of our approach and research method. In Chapter 4, we define security definitions which are applicable to OpenPGP key exchange methods. We will make use of them in the analysis. We then consider the OpenPGP key exchange method Web Key Directory. First, we describe the method's components and functionality in Chapter 5. After that, we analyze Web Key Directory according to our security definitions in Chapter 6. In Chapter 7, we conclude our work and give a brief outlook about research questions which could not be answered in this thesis.

[3]`https://wiki.gnupg.org/WKD/BachelorThesisIncreaseWKDUsage2021`

2 Background

In this chapter, we give a description of OpenPGP, the underlying cryptographic protocol used in the three key exchange and validation methods Web Key Directory, e-mail validating key server, and Autocrypt which are examined in this thesis. Furthermore, we describe the Transport Layer Security protocol briefly since it is used to secure access to Web Key Directory and key servers and is essential for the process of gaining trust in exchanged keys in both methods. The third method, Autocrypt, belongs to the class of opportunistic cryptography protocols and therefore a short description of that class if given, too.

2.1 Transport Layer Security

Transport Layer Security (TLS) [49] is a cryptographic protocol which provides a secure channel between two communication peers independent of the application layer protocol. It relies on a reliable and in-order medium provided by, for example, TCP.[1] Protocols such as HTTP in the World Wide Web context [48] and SMTP, IMAP, and POP in the mail context [37] are secured usually with TLS nowadays. TLS provides data authenticity, confidentiality, and integrity.

The protocol consists mainly of a handshake protocol and a record protocol. For efficiency reasons a combination of asymmetric and symmetric cryptography is used. Asymmetric cryptography is used in the handshake to authenticate peers, negotiate cryptographic modes and parameters, and a shared secret. The application layer data is then divided into records and symmetrically encrypted with this shared secret between the peers.

The level of security gained from TLS secured communication channel is severely affected by the used version, cryptographic modes, and parameters. By using ephemeral key exchange methods, like Ephemeral Diffie-Hellman (DHE) or Ephemeral Elliptic Curve Diffie-Hellman (ECDHE), TLS also provides forward secrecy. TLS versions 1.0 and 1.1 were formally deprecated in 2021, leaving versions 1.2 and 1.3 remaining as Proposed Standards [38].

[1]With DTLS a specification for TLS exists which does not rely on these properties, so that transport protocols like UDP can be supported.

TLS is used mostly in the client-server model context and requires that the server-side peer is authenticated. Authentication of the client peer is optional. However, for proper server authentication, a client must be able to check the validity of the server's public key ownership. Otherwise, an attacker could impersonate as the server without being detected by the client. Therefore, the public key must be signed by a Certificate Authority (CA) through an issued certificate and the client must trust the CA. A CA might be publicly trusted or a private "self-signed" one [56].

A Certificate Authority might issue intermediate certificates (also called sub-CA), which again can be used to issue certificates which proof the server's public key ownership. Therefore, a server's public key could be signed via a certificate which was issued by a sub-CA which was issued by another (sub-)CA. This is called a "chain-of-trust" and the chain must always start with a CA the client trusts. This trust anchor is normally distributed in operating systems or the web browser itself ("trust store") [56].

As a consequence, Certificate Authorities are the centerpiece of this trust model. While there are measures in place to detect CAs issuing illegitimate certificates at least after the fact (e.g. Certificate Transparency Logs [35]), ultimately, TLS clients will trust certificates as long as they are technically valid (e.g. within the validity period time) and signed by a CA in the client's trust store.

2.2 OpenPGP

As briefly addressed in Chapter 1, OpenPGP is a cryptographic protocol which is capable to sign and encrypt data. It is based upon asymmetric cryptography but also uses symmetric cryptography during encryption. OpenPGP provides data authenticity, confidentiality, and integrity. In this section, we provide a description of the OpenPGP basics we need in later chapters based on RFC 4880 [7].

OpenPGP offers probabilistic encryption due to a randomly generated symmetrical key during encryption. Each encryption process of the same plain text results in a non-connectable different encrypted text. OpenPGP is not forward secret because it uses its private-public key pair as long-term keys [7]. There are no session keys like in TLS with ephemeral key exchange methods. Once an attacker gets in possession of the private key, all past (and future) ciphers can be decrypted.

OpenPGP uses a binary format consisting of 8 bit octets. This format can be converted to a Radix-64 ASCII representation for transporting the format through communication channels which are incapable of transmitting 8 bit octets. This representation is nearly identical to the MIME Base64 encoding and also called ASCII Armor. A shell, for example, might interfere with raw binary data as it could be interpreted as a shell control character. Encrypted data, signatures, and keys can be exported in this armored encoding.

The OpenPGP format consists of several records. Record types are identified through specific tags. Records are loosely tied together. Only individual records are signed. As a consequence, records can be omitted but not modified without being noticed. OpenPGP messages (encrypted and/or signed data), keys and keyrings are in this format. Some record types are not included in all of these.

There are the following major records in OpenPGP keys:

User-ID A User-ID consists of a name and an e-mail address. An OpenPGP key may have multiple User-IDs, but exactly one should be primary.

Secret Key Contains the private key. An OpenPGP key may contain multiple keys, but exactly one should be primary. A non-primary key is referred to as a sub-key. Sub-keys are automatically trusted if they are signed by the primary key. Keys may have an intended usage, so different keys can be used for certification, authentication, encryption, and signatures. The primary secret key is used to sign records in the OpenPGP key.

Public Key Contains the (primary) public key.

Secret Sub-Key Contains the private key of a sub-key.

Public Sub-Key Contains the public key of a sub-key.

Signature Packets This record type is used for different signature purposes. It is used for self-signing other records with the primary secret key to ensure integrity. Another important purpose of this packet is to encode signatures from other OpenPGP key owners which express their verification of the OpenPGP key belonging to a user identified by a specific User-ID. The verification statement always refers to a single User-ID.

OpenPGP keys and even sub-keys can be revoked with a revocation signature (a signature packet type) [7, Sec. 5.2.1]. A revocation signature is valid if it is signed by the key being revoked, or by an authorized revocation key [7, Sec. 5.2.1]. The revocation signatures can be used both as stand-alone "revocation certificates" or as part of a OpenPGP keys.

An OpenPGP keyring contains multiple OpenPGP keys of different entities and acts as a local trust store. Each key in the keyring can be assigned a trust record which represents how trustworthy the owner of the key is regarding the verification of other keys, as specified by the keyring's owner.

A more human-readable representation of a public key is the fingerprint which is with 160 bit much smaller compared to a (RSA) public key with 2048 or 4096 bit length. The fingerprint of a OpenPGP key is always generated on its primary public key. OpenPGP keys can be identified by their fingerprints.

There are design decisions in OpenPGP which do not conform with standards of modern cryptography. For example, it uses sign-than-encrypt, message compression, and has extended adherence to backwards compatibility [22].

The trust model of OpenPGP does not rely on centralized authorities to validate the correct connection between a key and an entity. As described in Chapter 1, the trust model is decentralized and each entity must either check the validity of a key directly or indirectly via the “web-of-trust”. This uses both the verification signature packets in OpenPGP keys and the trust records in the local keyring.

2.3 Multipurpose Internet Mail Extensions

Key exchange methods in this thesis make usage of e-mails in the Multipurpose Internet Mail Extensions (MIME) format [19]. Due to the extensive character of the MIME specifications, we will only discuss the most important parts that are required for the description of the addressed key exchange methods.

A MIME message contains multiple headers and a body. The headers specify meta information about how the MIME message or body is to be interpreted. Header types can have parameters. Messages can contain multiple bodies which are separated by specially formed text strings, so-called boundaries, and are called multipart messages. Each of these body parts contain a header and a body themselves. It is thus possible to add different data types in one single message.

Important header fields are:

MIME-Version. Required header field to be compliant with MIME specification. Currently, the only possible parameter is the value 1.0.

Content-Type. The Content-Type field specifies the nature and form of the data in the body. If no Content-Type header is given, the default type is plain text in US-ASCII encoding. The value in this field is the Media Type but it is often denoted as Content Type in practice. Media Types consists of a type and subtype and are in the form `type/subtype`. This makes appropriate handling of such types possible even if an implementation does not know the specific subtype as it can still determine the nature of the given data. For example, there exist the following types: `text` (plain text), `image` (images), `application` (uninterpreted binary data or data which is processed by an application), `multipart` (multipart messages).

In the context of OpenPGP, three additional Media Types are defined [15]: `application/pgp-encrypted` is used for OpenPGP encrypted data, `application/pgp-signature` for OpenPGP signed data, and `application/pgp-keys` for the distribution of OpenPGP public keys.

Content-Transfer-Encoding. E-mails (without MIME) are 7 bit US-ASCII only. Many data formats are transmitted in their natural form and not in this representation. This header field defines which mechanism is used for the encoding in the body. There are seven encodings defined in the RFC: For example, `7bit`, `8bit`, `binary`, and `base64`. If a body is of the Content-Type `multipart` only, the first three encodings above are permitted.

3 Methodology

This chapter provides an overview of our approach for the examination of OpenPGP key exchange methods. We define typical operation classes for key exchange methods in the OpenPGP ecosystem and describe our approach for the security definitions used in the analysis. The operation classes and security definitions are kept broad enough to be used for the examination of Web Key Directory, e-mail validating key servers, and Autocrypt. In this thesis, we address Web Key Directory.

As described in Chapter 1, Web Key Directory, e-mail validating key server, and Autocrypt are fairly new. None of them are officially standardized, with only some being in the process of becoming standards. Therefore, we focus our research for method description and analysis on primary sources which includes drafts, specifications, and source code. The goal of this work is not to find flaws in the underlying cryptographic protocols OpenPGP or TLS used in the mentioned methods.

We define the following key exchange operations with which the functionality of key exchange methods can be modeled:

Key Submission A (public) key is uploaded to a certain key exchange method.

Key Lookup/Delivery A user requests a key for a certain user/e-mail address or fingerprint.

Key Update A submitted key is revoked or updated.

Key Refresh A user checks if certain keys in their keyring are up-to-date and refreshes certain keys which are out-of-date.

Key Deletion A submitted key is removed from the key exchange method.

For the analysis part, we need to know what a secure key exchange method is. Therefore, we need to define security definitions including attacker models and security goals. These are modeled to cover attacker models and security goals which are applicable to key exchange methods in the OpenPGP ecosystem. The defined operation classes and security definitions make a comparison between different key exchange methods possible, even if they take into account approaches to optimize or simplify prior mechanisms.

The provided security goals can be fulfilled (●), partially fulfilled (◐), or not fulfilled (○). A security goal is partially fulfilled if the partial fulfillment criteria defined in the security goal are satisfied. If a goal has none of these criteria it can only be either fulfilled or not fulfilled.

The definition of the security goals follows the principles of confidential requests, authentic keys, and privacy. The content and even the metadata originating from a request might have security implications if leaked as they could indicate communication partners or persons' trust dependencies. A typical expectation from the user's perspective is that a key provided in a key exchange method belongs to the person indicated in the key and that the provided key is identical to the submitted key. Keys might be submitted not only by the key owner and OpenPGP keys might contain personal data. Unauthorized publishing of personal information can lead to severe security implications and even infringement of laws which endanger the availability of the key exchange method.

In this thesis, we describe and examine the Web Key Directory key exchange method in detail. The description contains the components and functionality of Web Key Directory. In the analysis, we are concerned with both the Web Key Directory specification and its reference implementation. We evaluate whether the specification and reference implementation fulfill our security goals, or not. In addition, we evaluate the coverage of the provided key operations.

4 Security Definitions

In this chapter, we define attacker models and security goals (our security definitions). The security definitions are modeled in such a way that they cover the needs for OpenPGP key exchange methods and can be used to analyze and evaluate them. Based on the definitions, we can determine whether a key exchange method is secure, or not. While we will only analyze Web Key Directory in this thesis, the definitions can also be used to compare several key exchange methods.

4.1 Attacker Model

We define four different attacker models. The attacker in none of these models is able to break TLS or OpenPGP.

Malicious User. (A_U) Malicious users are users described by the system of the key exchange method. They may deviate arbitrarily from the key exchange method specification.

These users may have knowledge about the name and e-mail address of other users since these information are not kept confidential and are necessary for participation in the given systems.

Malicious Server. (A_S) This attacker class has full control over services and data provided by the server.

For this class it makes no difference whether the server is successfully compromised by an attacker, a software malfunction, or a misconfiguration. A server could be, in our case, for example, a key server or a mail transfer agent.

Active Network Attacker. (A_{aN}) An Active network attacker has full control over the network and may access, modify, create, or delete all unprotected traffic.

Passive Network Attacker. (A_{pN}) A passive network attacker can read all unprotected traffic of the network.

4.2 Security Goals

The primary purpose of our security goals in this thesis is to ensure the integrity and authenticity of OpenPGP keys and the confidentiality of OpenPGP key exchange method requests. OpenPGP certificates contain personal data which might fall under certain data protection laws; thus, a key exchange method should also handle these data properly to ensure service availability (see SKS key server network shutdown in Chapter 1) and the privacy of their users.

OpenPGP keys contain (possibly multiple) User-IDs which connect the key material with a name and an e-mail address. For this thesis, we define the term "legitimate key owner" (short: "key owner") as the person identified by the e-mail address of the key's User-ID. If a key contains User-IDs of different persons, each person is only the "legitimate key owner" with regard to those User-IDs which identify this person via their e-mail address. For example, if Mallory (mallory@example.org) creates an OpenPGP key with a User-ID "Alice <alice@example.org>", only Alice is considered the legitimate key owner (even though Mallory controls the private key). Thus, only Alice and not Mallory must be allowed to, for example, approve (or refuse to approve) key authenticity for this key as described below.

Having defined the "(legitimate) key owner", we can now define the security goals which an OpenPGP key exchange method should fulfill:

Authentic keys. (G_A) If a user receives a requested key, this key must be the same and most recent confirmed submitted key. A submitted key qualifies as confirmed if the legitimate key owner approves its authenticity and integrity. This confirmation must be done for each of the key User-IDs.

Partial fulfillment: We define this security goal as partially fulfilled if the received key is a subset of the most recent confirmed submitted key, all parts in the received key are identical to those in the submitted key, and the key material (key and subkey parts) of the submitted key is present in the received key. In this case, the confirmation must only be done for those User-IDs which are part of this (partial) key.

Key exchange termination. (G_T) A legitimate key owner must be able to remove and terminate the key exchange for their key from the key exchange method.

Key revocation. (G_R) If a user requests a key for which a submitted valid revocation signature exists, this revocation signature must be delivered, too. A revocation signature is valid if it is signed by the key being revoked, or by an authorized revocation key [7, Sec. 5.2.1]. The revocation signature does not have to be confirmed as described in security goal "authentic keys". The above concerns key revocation signatures and subkey revocation signatures [7, Sec. 5.2.1], both as stand-alone revocation certificates or as part of a larger

OpenPGP key. A later key exchange termination as per G_T supersedes the obligation for revocation signature delivery defined here.

Confidential key requests. (G_C) If a user requests a key for a User-ID or a fingerprint nobody except that user and the server the request is made to must be able to determine information regarding User-ID, fingerprint, or requested key.

Partial fulfillment: We define this security goal as partially fulfilled if the domain parts of the email addresses in the User-IDs are not kept confidential, but all other aspects of confidential key requests are fulfilled.

Privacy considering key exchange. (G_P) Names, e-mail addresses, and images must only be exchanged if the legitimate key owner has approved that.

5 Web Key Directory

In this chapter, we address the key exchange method Web Key Directory (WKD). It provides a method to associate OpenPGP keys to a well-known URI and distribute them over HTTPS. Web Key Directory incorporates security properties of the TLS ecosystem by the use of HTTPS for key exchange. Trusting exchanged keys over this method, therefore, implies trusting in TLS and the (web)server responsible for the publication. We will describe the method and used components and their functionality. In Section 5.1, we will be concerned with the Web Key Directory Discovery Protocol and in Section 5.2 with the Web Key Directory Update Protocol. We also address inconsistencies and specification gaps in the Web Key Directory specification in Appendix A.

The OpenPGP Web Key Directory draft [33] describes two protocols: The main (Web Key Directory) protocol is defined in Section 3.1 and the update protocol is defined in Chapter 4 of the draft specification. In the following, we will name the Web Key Directory Protocol as Web Key Directory Discovery Protocol to distinguish it from the eponymous method. With this protocol, users are able to discover and receive OpenPGP keys published via the Web Key Directory. The Web Key Directory Update Protocol can be used to submit keys in an automated, e-mail driven basis to the Web Key Directory.

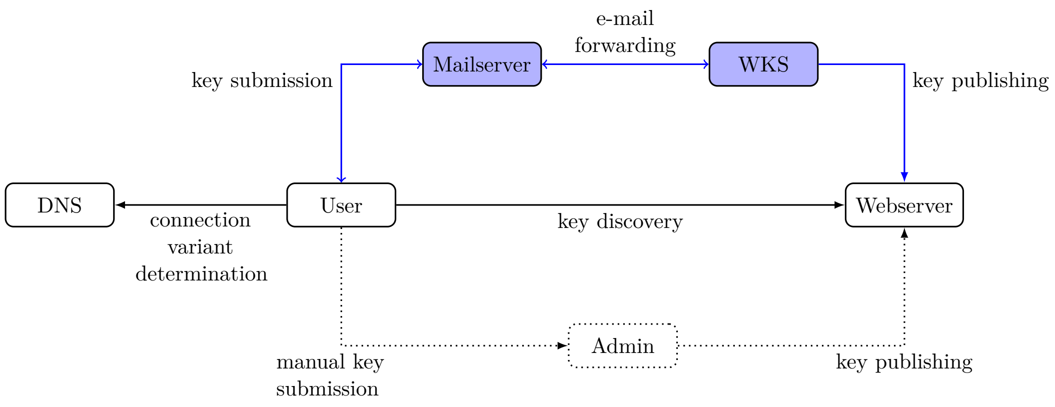


Figure 5.1: System image of components used by the Web Key Directory method. The blue path and components are used in addition if the update protocol is supported.

In Figure 5.1, we see an image of components used by Web Key Directory and their connections. The black components and connections are required for Web Key Directory. In Web Key Directory, the user asks a DNS server to determine the responsible web server for a requested OpenPGP key. The requested key is then discovered via a request to a well-known path. It is possible to manually submit OpenPGP keys to the Web Key Directory via the server administrator without making use of the update protocol. The security properties of this approach are hardly determinable because they depend on various conditions which are not in the scope of the key exchange method. We assume that keys submitted via this approach are securely transmitted and the administrator processing this submission complies with the specification and not consider this approach further. This is not a major assumption since such administrators are already capable of compromising the server and, therefore, the whole Web Key Directory service.

In addition, Web Key Directory provides with the Web Key Directory Update Protocol an automated e-mail based protocol to ease the submission of OpenPGP keys and relieve the administrators. The blue-colored components in Figure 5.1 are also required to support the Web Key Directory Update Protocol. Although defined in the context of Web Key Directory, the update protocol can also be used for publishing OpenPGP keys via the DNS based OpenPGP DANE key exchange approach. We will not consider OpenPGP DANE further since it is only stated as another possible way for publishing the successfully verified keys and not part of the Web Key Directory method which is addressed in this chapter.

In Appendix A, we will describe inconsistencies and specification gaps in both the Web Key Directory Discovery Protocol and the Web Key Directory Update Protocol.

5.1 Web Key Directory Discovery Protocol

The purpose of the Web Key Directory Discovery Protocol is to locate and request OpenPGP keys which are published by the Web Key Directory method. The method associates keys with their e-mail address. Only keys with User-IDs containing e-mail addresses can be published by this method. As OpenPGP keys can contain multiple User-ID packets, the method ensures that published keys must contain only the User-ID used for the association. Other User-IDs and their binding signatures must be removed from the key.

Web Key Directory makes OpenPGP keys available via web servers with the use of HTTPS. Thus, we need to know which domain name is used for the responsible web server. The method provides two variants for this: the direct and the advanced method. Both variants use the domain from the e-mail address of the OpenPGP key. For a key associated with the e-mail address Alice.Wonderland@example.org this would be example.org. For the advanced variant the subdomain “openpgpkey” is prepended to the domain. Only if this subdomain does not exist or consists of an empty TXT Resource Record, the direct method is allowed to be used. The direct method uses only the base domain. By using “openpgp” subdomain of the e-mail domain the advanced variant is more flexible as multiple e-mail domains can be pointed to a single Web Key Directory. This also enables the possibility to provide the Web Key Directory key exchange method for own domains without hosting the Web Key Directory by yourself. This is called Web Key Directory as a Service (WKDaaS). The construction of the URI differs slightly between both variants which we will see in Figures 5.2 and 5.3.

The discovery of published OpenPGP keys is done via well-known URIs. The concept of well-known URIs is standardized and introduces the path prefix `/.well-known/` for HTTP(S) and (Secure) WebSockets [42]. Other protocol schemes are possible, too, but only HTTPS is relevant for Web Key Directory. Therefore, well-known URIs enable a standardized procedure for applications to discover information over the Internet. Applications which want to publish information via such URIs must register the used name in conformance with the RFC by the well-known name registry maintained by IANA. For Web Key Directory, this name is “openpgpkey” which results in the well-known path `/.well-known/openpgpkey/`. The advanced connection variant appends the domain for which keys are published to this path. This is beneficial for systems which serve Web Key Directories for multiple domains within the Web Key Directory well-known path. Within this path, published keys and protocol information are organized. We denote the concatenation of schema, domain, well-known path, and in case of the advanced variant e-mail address domain in the following as base URI or “BASE_URI”. With regard to the discovery protocol, only the directory `hu/` is relevant as it contains all published keys. Nevertheless, the Web Key Directory method requires to serve a policy file which can provide information used for key submission. The policy file can be empty and clients can use it to figure out if Web Key Directory is offered for a given domain. It is not sufficient to check only the existence of that file to determine the availability of a Web Key Directory. Policy flags are defined which disable this publication method. We will address the policy file in the update protocol section (Section 5.2).

In addition to the domain and path, we need to know how the names of these keys are organized to successfully discover OpenPGP keys. We know that Web Key Directory associates keys with their e-mail addresses. The local-part of e-mail addresses (the part before the @) in lower-case is used for the association. All ASCII characters in uppercase are mapped to lower-case. Non-ASCII characters are not changed. This lower-case representation of the local-part is hashed with SHA-1 and then encoded with Z-Base-32 [67, Sec. 5.1.6]. Z-Base-32 is a Base32 [26, Chap. 7] like data encoding but deviates slightly from the specification of Base32. Just like Base32 it maps every 5 bits of the input byte string to one of 32 ASCII letters or digits. This alphabet differs from the one used by Base32 and is chosen in such a way that it is more human-readable. Additionally, Z-Base-32 does not use padding as in Base32 where the encoded output is always a multiple of 8 symbols. In the case of Web Key Directory, the local-part is hashed to 160 bits via SHA-1 and then encoded resulting in a fixed length of 256 bits which can be represented with 32 ASCII characters. The purpose of the encoding is to have a name for a key which does not contain symbols which would prevent the storing in file systems.

Besides the Z-Base-32 encoded SHA-1 hash of the lower-case local-part, the URI contains the query parameter "`l`" which contains the unchanged local-part in percentage escaping.

In Figures 5.2 and 5.3, we see the corresponding key lookup request URIs formed by the advanced and direct connection variant for an OpenPGP key associated with Alice.Wonderland@example.org.

$$\overbrace{\texttt{https://}\overbrace{\texttt{openpgpkey.example.org}}^{\texttt{openpgpkey}\text{ subdomain of e-mail address domain}}\texttt{/.well-known/openpgpkey/}\overbrace{\texttt{example.org/}}^{\text{e-mail address domain}}}^{\text{base URI}} \quad \swarrow$$

$$\hookrightarrow \texttt{hu/}\underbrace{\texttt{5gt5gnaq1zccz7f19kq9whu4ezf1uofq}}_{\text{Z-Base-32 encoded SHA-1 hash of lower-case local-part}}\texttt{?l=}\underbrace{\texttt{Alice.Wonderland}}_{\text{local-part of e-mail address with URI percentage-encoding as needed}}$$

Figure 5.2: Example URI used by the advanced connection variant for requesting Alice.Wonderland@example.org's key. The URI has been wrapped for rendering purposes.

$$\overbrace{\texttt{https://}\overbrace{\texttt{example.org}}^{\text{e-mail address domain}}\texttt{/.well-known/openpgpkey/}}^{\text{base URI}} \quad \swarrow$$

$$\hookrightarrow \texttt{hu/}\underbrace{\texttt{5gt5gnaq1zccz7f19kq9whu4ezf1uofq}}_{\text{Z-Base-32 encoded SHA-1 hash of lower-case local-part}}\texttt{?l=}\underbrace{\texttt{Alice.Wonderland}}_{\text{local-part of e-mail address with URI percentage-encoding as needed}}$$

Figure 5.3: Example URI used by the direct connection variant for requesting Alice.Wonderland@example.org's key. The URI has been wrapped for rendering purposes.

The server hosting the Web Key Directory must provide HTTP HEAD in addition to HTTP GET. With this, clients can check for the existence and whether there are newer keys without downloading the key as payload.

A requested OpenPGP key must be delivered in binary and not in ASCII armored form. It must contain a User ID packet for the requested e-mail address. Delivered keys can be expired or revoked. It is up to the client to handle this properly. A server may deliver revoked keys for a requested e-mail address in addition to the new valid key in a single response. Keys in this response are concatenated. The preferred MIME Content-Type for the response is "application/octet-stream" which indicates generic binary data. Any other Content-Type can also be used and clients should also accept these.

5.2 Web Key Directory Update Protocol

The Web Key Directory Update Protocol is an additional protocol in the Web Key Directory key exchange method. The purpose of this e-mail driven protocol is to automate and ease the publication process of OpenPGP keys to the Web Key Directory. It is based primarily on OpenPGP secured e-mail exchange between the user and their Web Key Directory provider.

This protocol covers only the e-mail driven submission process and not the creation of an OpenPGP key. We assume that a key was successfully created and should now be published. For the submission process, we need information which is served by the web server hosting the Web Key Directory. In the description of the Web Key Directory Discovery Protocol, we considered the determination of the responsible web server based on a given e-mail address in detail. For the description of the update protocol, we make use of the "base URI" to reference the relative position of used files. Furthermore, we don't depict the mail server in Figure 5.1 for simplification of the protocol procedure. With regard to the update protocol, it only processes e-mails from the user and the Web Key Directory Update Protocol implementation and makes e-mail communication between them possible. We denote the implementation of the Web Key Directory Update Protocol as the Web Key Directory Service (WKS).

In Figure 5.4, we see the update protocol as a diagram. We must consider that the protocol procedure can be changed by the usage of policy flags. For better comprehensibility, implications of policy flags are not part of the figure. The policy flag file and their implications will be addressed in detail below. For the submission process the user wants to sent their OpenPGP key via e-mail to their provider. Therefore, they must know the responsible submission e-mail address which is located at the file served under `BASE_URI/submission-address`. This file must contain only the submission e-mail address in one line and is terminated by `LF` or `CRLF` line break. The submission e-mail address can optionally be defined as a policy flag. The actual submission process of the user's key contains three e-mails: The submission e-mail, confirmation request, and confirmation response.

The user has to send their key via an OpenPGP encrypted e-mail to the submission e-mail address. Therefore, the user needs to obtain the OpenPGP key for the provider's submission e-mail address. It can be received via Web Key Directory or

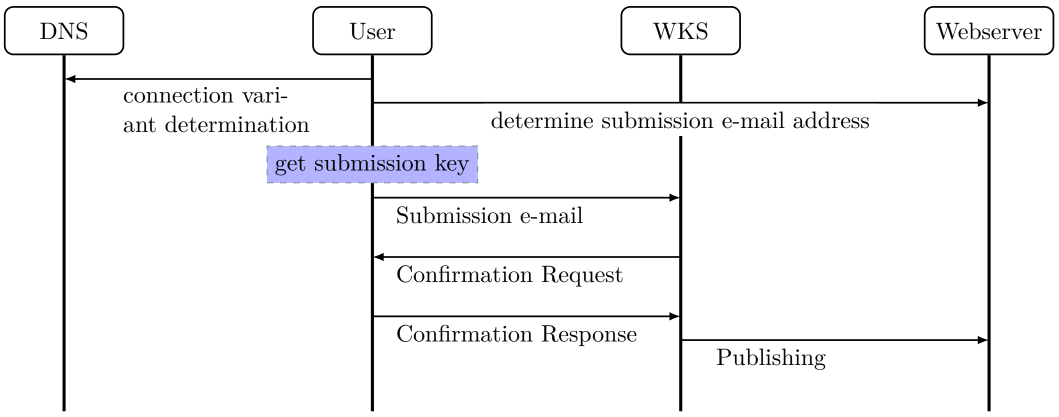


Figure 5.4: Diagram of the Web Key Directory Update Protocol. The procedure can be altered by the usage of policy flags. “get submission key” means the key discovery process (see Sec. 5.1) for the OpenPGP key associated with the submission e-mail address performed by the user. The submission e-mail is encrypted, the confirmation request is signed and contains also an encrypted body-part, and the confirmation response is signed and encrypted with OpenPGP.

DANE. The provider must ensure an OpenPGP key capable of encrypting and signing is published for the designated submission e-mail address.

The user begins the submission process of their key by sending a submission e-mail to their provider. The submission e-mail must be encrypted, but not signed. The reason for the non signing is that the server could not properly validate the signature because the required OpenPGP public key is sent with the submission e-mail and is not yet confirmed to belong to the legitimate e-mail user. The submission e-mail must contain the user’s OpenPGP public key in ASCII armored representation and the Content-Type `application/pgp-keys` as required by the PGP/MIME specification for the distribution of OpenPGP public keys [15, Sec. 7]. In addition, the above MIME part must be encrypted with the OpenPGP key obtained for the submission e-mail address. The detailed structure of PGP/MIME encrypted and/or signed messages is not relevant for the protocol description and therefore not further discussed here. RFC 3156 [15, Sec. 4, 5] provides further information.

The provider checks if the User-ID of the submitted key matches an account name of the provider and if the submission satisfies policies defined in the policy file. Unless defined otherwise in the policy file, the provider must ensure that the submitted OpenPGP key is legitimate. This is done via the e-mail driven confirmation request and response challenge.

For the confirmation request the provider sends an e-mail in response to the above submission e-mail. This response must be sent from the designated submission e-mail address to the e-mail address given in the submitted OpenPGP key. The confirmation request is a PGP/MIME message containing two MIME parts. It is signed by the provider's OpenPGP key associated with the submission e-mail address. The first part is of the Content-Type `text` and its purpose is to explain the confirmation request to the user and what they have to do for a successful confirmation. The second part contains a Web Key Directory specific textual format and must be encrypted to the submitted key but not signed. The used Content-Type differs according to the Web Key Directory draft version the provider supports. `application/vnd.gnupg.wkd` must be used for draft versions as of 5 or higher and `application/vnd.gnupg.wks` for an unknown or less than draft version 5.

The textual format contains name-value pairs with only one of these pairs per line. Lines are terminated by `LF` or `CRLF` and empty lines are ignored. A name is terminated by a colon. The possible name-value pairs of this format are fully characterized by the content description of the confirmation request and response as specified below.

A confirmation request must contain the following name-value pairs in the following order:

type The value must be "confirmation-request".

sender This e-mail address must be equal with the e-mail address from the e-mail's "From" header. Exactly one e-mail address must be given. The purpose of this name-value pair is to indicate the user the e-mail address where the confirmation response is to be sent to.

address The value must be the e-mail address defined in the User-ID of the submitted key. It should match with the recipient's e-mail address used for this confirmation request. As required for OpenPGP User-IDs the value must be UTF-8 encoded.

fingerprint The value must be the fingerprint of the submitted key in upper-case hexadecimal encoding without interleaving spaces.

nonce The nonce must be a string of random ASCII letters and digits with a length between 16 to 64 characters. The purpose of the nonce is to act as a proof for the legitimacy of the key submission. As the nonce is in the encrypted part of the confirmation request, only the legitimate key owner can decrypt it.

The user receives the confirmation request of their provider and verifies the outer message signature. This can be done without the user having to enter their private key password and is intended to reveal spam at an early stage. The user is expected to disregard this confirmation request if they can not verify the signature or if the signing was not conducted by the OpenPGP key associated with the provider's submission e-mail address. Afterward, the encrypted second part in the confirmation request is decrypted and it must be checked if the constraints of the textual format are given. This includes that the fingerprint pair must match the fingerprint of the OpenPGP key the user wants to publish, the address pair must match the e-mail address of a User-ID of such a key, and the user must have a pending publication request. If and only if all of these constraints are given, this confirmation request is legitimate and should be responded via the confirmation response. A suitable Mail User Agent (MUA) should perform above verification and checks automatically and then sent the confirmation response silently if the verification and checks succeeded. If any of the above constraints are not fulfilled, the MUA may notify the user.

After the first successful delivery of the confirmation request, no further delivery attempts should be performed by the provider regardless of the confirmation state of the user's key submission. The provider is expected to discard pending key submissions without confirmation after a suitable time.

The confirmation response must be sent from the e-mail address associated with the User-ID of the key which should be published. The confirmation response must be addressed to the "From" header e-mail address of the confirmation request. It must be signed with the user's key and encrypted to the provider's key and must use the PGP/MIME Combined format described in [15, Sec. 6.2]. The format used for the confirmation response is the same as described above for the confirmation request.

A confirmation response must contain the following name-value pairs in the following order:[1]

type The value must be "confirmation-response".

sender This e-mail address must be equal with the e-mail address from the "sender" parameter of the confirmation request.

address This e-mail address must be equal to the e-mail address in the "address" parameter of the confirmation request.

nonce This must be the value of the "nonce" parameter in the confirmation request.

The provider receives the confirmation response and checks if the extracted nonce matches with the one used in the confirmation request. If they match, the provider assumes that the legitimate key owner wants to publish their key. The key is then published and the user is notified about the successful publication.

As indicated above, the Web Key Directory draft defines a policy file. It may be empty since all entries are optional. The purpose of this file is to provide certain information about the Web Key Directory to the provider's clients in advance of key generation and submission. Based on given policies, the submission process can be modified. This file is located at `BASE_URI/policy` and must be served if Web

[1]The "sender" parameter description was clarified in draft version 14. The "address" parameter was added in draft version 14.

Key Directory is available for the given domain. Clients can check this file to figure out if Web Key Directory is offered. The policy file is made up of keywords which can have additional values. A keyword with value is directly followed by a colon as a separator to the value. Between colon and value adding white space is possible. Each line contains one keyword and is `LF` or `CRLF` terminated. Empty lines and lines starting with a number sign ("`#`") are ignored as comments. Keywords can contain letters, digits, hyphens, and dots. They always start with a letter. Underscores are allowed as name space delimiters. All official keywords are defined in the draft itself. Experimental or provider specific keywords must be prefixed with an underscore and a domain.

The following keywords are currently defined:

mailbox-only The provider does only accept e-mail addresses in the User-ID packets. User-IDs with an additional real name will be rejected and can not be published.

dane-only The provider provides only an OpenPGP DANE service. A Web Key Directory is not provided. OpenPGP keys submitted through the update protocol are published via DNS.

auth-submit The provider uses other mechanisms to check the validity of a key submission. For example, the connection between the client's MUA and the provider's MTA might be authenticated. As a consequence, no confirmation request and confirmation response must be performed and OpenPGP keys are published immediately after their submission.

protocol-version A provider can specify its supported version of the Web Key Directory Update Protocol. Implementations can use this information to handle workarounds for older versions of this protocol.

This keyword must contain the draft revision number as an integer value.

submission-address An alternative way to define the submission e-mail address. This e-mail address must be the same as defined in the `BASE_URI/submission-address` file.

This keyword must contain the submission address as value.

6 Security Analysis of Web Key Directory

In the chapter above, we described the Web Key Directory method in detail. In this chapter, we will analyze this method. We will discuss various design decisions of the specification and examine their security implications. Furthermore, we will address the reference implementation of the Web Key Directory method and examine inconsistencies and implementation errors. In addition, we will introduce an attack on the reference implementation to publish keys for any e-mail address for any domain a Web Key Directory provider offers.

6.1 SHA-1 Hashed Local-parts

In Section 5.1, we described that the URI for the OpenPGP key discovery contains the local-part of the requested e-mail address twice: as a lowercase mapped, SHA-1 hashed and then Z-Base-32 encoded hash and an unaltered GET parameter.

The choice of SHA-1 stands out here because Web Key Directory is a quite recent OpenPGP key exchange method currently still in the status of an RFC draft, while SHA-1 has known weaknesses and has been deprecated for several uses for quite a while [44, 3, 62]. The protocol authors evaluate the usage of SHA-1 as unproblematic because “the use of SHA-1 for the mapping of the local-part to a fixed string is not a security feature but merely used to map the local-part to a fixed-sized string made from a well-defined set of characters” [33, Chap. 5] and that “it is not intended to conceal information about a mail address” [33, Chap. 5].

We will now discuss whether it is possible to attack the key discovery of Web Key Directory by taking advantage of the SHA-1 hashed local-part. We assume that the OpenPGP keys published in the Web Key Directory are served by the Z-Base-32 encoded hash and not by the unaltered GET parameter. This is not a major assumption because it corresponds to the Web Key Directory setup with the Web Key Directory server reference implementation (which is the only known one to us) and a generic web server as described in the GnuPG wiki [21]. If the GET parameter is used instead of the encoded hash, the below attack is not possible. We define the following attack vector:

Alice has a mail provider which offers Web Key Directory and the Web Key Directory Update Protocol. Mallory has access to this mail provider, too, so she is able to register e-mail users with this provider. Mallory wants to publish an OpenPGP key to the Web Key Directory which will be served for key discovery requests for Alice's e-mail address. Mallory, therefore, wants to find and register an e-mail address which results in the same hash as Alice's e-mail address. In other words: Mallory could replace Alice's OpenPGP key if she is able to break the second-preimage resistance of SHA-1. This would target mainly our security goal "Authentic keys" (G_A) but also "Privacy considering key exchange" (G_P), especially if Alice has not already published a key.

As a reminder, the second-preimage resistance property states: For a given hash function $h\colon X \to Y$ and input $x \in X$, it is computationally infeasible to find a second input $x' \in X$ with $x \neq x'$ such that $h(x) = h(x')$ [58].

The complexity to find such a second-preimage for a cryptographic hash function with an image length of n bits is maximally 2^n steps because w.l.o.g. $|X| > 2^n$, $|Y| = 2^n$ and therefore the $2^n + 1$ distinct input must match to one of the previous images. The SHA-1 hash function maps to a 160 bit image and therefore the theoretical resistance would be 160 bit or an attack would need a complexity of 2^{160}.

According to Tchórzewski et al. [58], the resistance for a second-preimage attack on SHA-1 can be lowered depending on the input length. This reduction can be calculated by the following equation:

$$L(M) = \left\lceil \log_2 \frac{\text{len}(M)}{B} \right\rceil \tag{6.1}$$

$\text{len}(M)$ is the length of the input message M in bits. B is the block size of the hash function. SHA-1 uses a block size of 512 bits. Therefore, the second-preimage resistance for SHA-1 can be calculated by:

$$\text{2nd-Preimage-Resistence}_{\text{SHA-1}}(M) = 160 - \left\lceil \log_2 \frac{\text{len}(M)}{512} \right\rceil \tag{6.2}$$

Our input message in this case is the local-parts which can have a length from 1 byte up to (including) 64 bytes (512 bits) [30, Sec. 4.5.3.1.1]. This results in a resistance of 160 bits for an input length of 64 bytes and a resistance of 147 bits for an input length of 500.000 bytes. The function is also plotted in Figure 6.1.

Consequently, the second-preimage resistance of SHA-1 is not broken, even if a mail server implementation does not enforce the local-part limits of the RFC, and we can not attack the Web Key Directory method via this attack vector.

The question arises whether this "strong" second-preimage property of SHA-1 is really necessary or whether we can achieve better results by taking advantage of the "much weaker" collision resistance property.

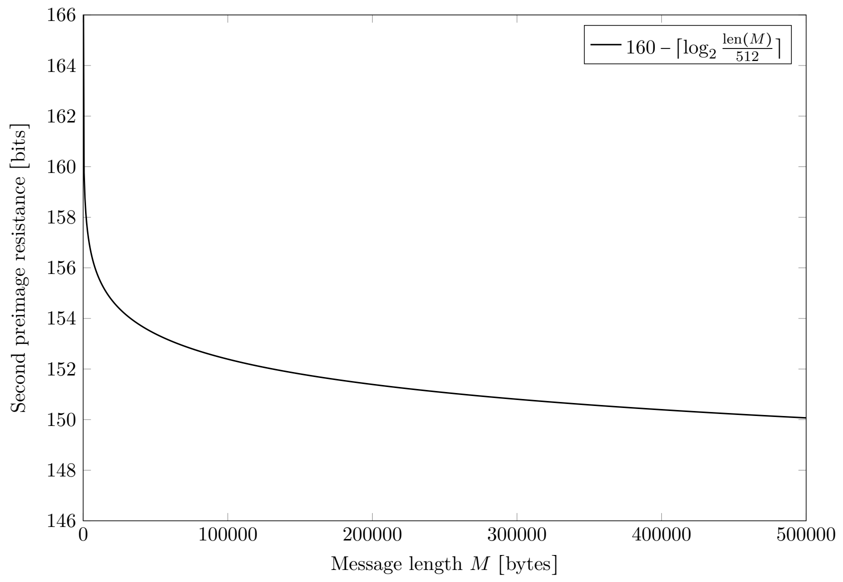


Figure 6.1: This diagram shows the resistance of SHA-1 in bits dependent on the input length.

As a reminder, the collision resistance property states: For a given hash function $h\colon X \to Y$, it is computationally infeasible to find a pair $x, x' \in X$ with $x \neq x'$ such that $h(x) = h(x')$ [58].

For this, we must adapt our attack vector. Mallory (A_U) is now also in a position where she can register e-mail addresses for other users. Alice needs an e-mail address and asks Mallory to register one for her. If Mallory can generate a suitable e-mail address and a second different e-mail address where the local-parts result in the same SHA-1 hash, she might perform the attack on the Web Key Directory method. This would also target the same security goals (mainly G_A but also G_P) as above.

We want to consider two aspects: First, we mentioned the collision resistance as "much weaker" because the complexity to find a collision for a cryptographic hash function with an image length of n bits is up to $2^{n/2}$ because of the birthday paradox [14]. For SHA-1 the complexity for a collision attack would therefore be 2^{80}. The best known collision attacks on SHA-1 have a complexity of $2^{61.2}$ and are practically feasible [36].

Furthermore, local-parts can not be treated as generic binary data with arbitrary length. Even if a mail server implementation technically supports local-parts longer than the specified 64 byte limit, a cryptic local-part containing arbitrary characters is neither intuitive nor practical. Local-parts must consist of UTF-8 characters, but not all bytes represent valid UTF-8 characters, limiting the viable input set further. The implications of such a limited subset $X' \subset X$ on the current SHA-1 collision attacks are not exactly known. Theoretically, this could make these attacks not just unfeasible, but impossible if the reasonable input set X' is smaller than the image of the hash function Y and the hash function is injective for $X' \subset X$ and Y. While we can not give exact numbers on the attack complexity for a limited X', it seems at least intuitive that such constraints on the input set increase the attack complexity.

The described attacker model where Mallory can assign an e-mail address to Alice (instead of Alice picking one) seems rather unrealistic. Even in this case we would assume the attack complexity to be considerably higher than a plain collision attack. Thus, while we can not guarantee this attack to be unfeasible, we will not consider it further.

6.2 Lowercase Mapped Local-parts

In Appendix A (Inconsistencies and Specification Gaps), we mentioned the draft's assumption for the key discovery that "almost all MTAs treat the local-part case-insensitive" [33, Sec. 3.1]. In this section, we will discuss implications that could result from this assumption and conduct a small evaluation about how the case of local-parts is handled by providers.

In Section 6.1, we described an attack vector where Mallory tries to attack the second-preimage resistance of SHA-1 to publish an illegitimate OpenPGP key for Alice. A similar attack would be possible, if Mallory (A_U) could register an e-mail address similar to Alice's which only differs in case and Mallory receives at least e-mails for her registered e-mail address. It is necessary that Mallory at least receives e-mails for her own e-mail address because she must be able to receive the confirmation request to publish keys. For the attack it does not matter, even if this would imply a configuration mistake by the e-mail provider, if Mallory or Alice receives e-mails from each other or if Alice's e-mails are delivered only to Mallory. This attack utilizes the lowercase mapping of local-parts prior to hashing described in [33, Sec. 3.1]. As a consequence, OpenPGP keys for Alice's and Mallory's e-mail addresses would be mapped to the same Z-Base-32 encoded SHA-1 hash and therefore to the same file stored in the file system. Therefore, mainly the security goals "Authentic keys" (G_A) but also "Privacy considering key exchange" (G_P) is targeted. As in Section 6.1 mentioned, we disregard the case where the keys are delivered based on the unaltered GET parameter.

Handling local-parts in a case-sensitive manner is in full compliance with the relevant e-mail specification [30] and historically intended but nowadays probably not realistic since e-mail addresses are treated as case-insensitive by convention (for reasons including convenience) and the possibility to treat them case-sensitive might not be widely known. Nonetheless, providers offering Web Key Directory and the Web Key Directory Update Protocol are consequently prone to this attack if the corresponding e-mail provider allows e-mail addresses which differ only in case. However, it should also be noted that it is likely that such behavior would result in problems not only for Web Key Directory but also other tools and protocols because handling local-parts in a case-insensitive manner might be prevalent nowadays.

To get an impression of the real world situation about how the case of local-parts is handled by e-mail providers, we conducted a small case study. For this evaluation, we chose the e-mail providers used in [55, Table 4] which are based on common e-mail providers and the e-mail providers mentioned in [21, Mail Service Providers offering WKD] as explicitly offering Web Key Directory. We chose 16 random lowercase ASCII letters to build our local-parts. The first local-part ($\texttt{ID}_1$) is the unaltered string: `eavhupphzqtswnxm`. The second local-part ($\texttt{ID}_2$) is the string with the first three letters capitalized: `EAVhupphzqtswnxm`.

The evaluation covers two questions:

1. *Case-sensitive IDs.* Is the registration of accounts / e-mail addresses with the names $\texttt{ID}_1$ and $\texttt{ID}_2$ possible? This means, is it possible to have e-mail addresses which differ only in their case?
2. *Case-sensitive delivery.* Are e-mails sent to $\texttt{ID}_1$ and $\texttt{ID}_2$ delivered to the same e-mail account?

The test was conducted as follows:

1. Register an account with the name of $\texttt{ID}_1$.
2. Try to register an account with the name of $\texttt{ID}_2$.
3. Sent an e-mail to $\texttt{ID}_1$`@domain.tld` and check if it is delivered to the account of $\texttt{ID}_1$ or $\texttt{ID}_2$.
4. Sent an e-mail to $\texttt{ID}_2$`@domain.tld` and check if it is delivered to the account of $\texttt{ID}_2$ or $\texttt{ID}_1$.

For the evaluation we apply the following criteria:

For the question “Case-sensitive IDs”, we denote the test as negative “no” if only $\texttt{ID}_1$ and not $\texttt{ID}_2$ could be registered. We use “(no)” if we could not register any account but the registration form allows only lowercase characters. This could be a characteristic of registration form and could not correspond with restrictions in the underlying mail infrastructure. However, we do not assume that the provider allows the registration of such case-sensitive e-mail addresses in another way, which would result in the same non-support. If we can register both accounts, we consider the test as positive and mark it as “yes”.

For the question “Case-sensitive delivery”, we denote the test as positive “yes” if e-mails sent to $\texttt{ID}_1$ are delivered to $\texttt{ID}_1$ and e-mails sent to $\texttt{ID}_2$ are delivered to $\texttt{ID}_2$. Otherwise, the result is considered to be negative is donated as “no”. Additionally, we add the ID, to which e-mails are delivered, to the test result: “($\texttt{ID}_1$)” if the e-mails were delivered to $\texttt{ID}_1$, "($\texttt{ID}_2$)" if they were delivered to $\texttt{ID}_2$, and "($\texttt{ID}_1$, $\texttt{ID}_2$)" if the e-mails were delivered to both.

The above evaluation is shown in Table 6.2. All tests which could be successfully conducted indicated that local-parts are handled in a case-insensitive manner (i.e. in lowercase) by these e-mail providers. We could not identify any e-mail provider which would be susceptible to this attack in our short survey.

Table 6.2: Evaluation of case sensitivity behavior of e-mail providers. Providers in the upper section are chosen from [55, Table 4]. Providers in the lower section are chosen from [21, Mail Service Providers offering WKD].

	Domain	**Case-sensitive IDs**	**Case-sensitive delivery**
AOL Mail	aol.com	no	no ($\texttt{ID}_1$)
FastMail	fastmail.com	no	no ($\texttt{ID}_1$)
Gmail	gmail.com	no	no ($\texttt{ID}_1$)
GMX Mail	gmx.de	no	no ($\texttt{ID}_1$)
Outlook.com	outlook.com	no	no ($\texttt{ID}_1$)
Yahoo! Mail	yahoo.com	no	no ($\texttt{ID}_1$)
Runbox	runbox.com	no	-[a]
Hushmail[b]	hushmail.com	(no)	-
Mail.ru[b]	mail.ru	(no)	-
iCloud[b]	-	-	-
Zoho Mail[b]	-	-	-
posteo	posteo.de	no	no ($\texttt{ID}_1$)
protonmail	protonmail.com	no	no ($\texttt{ID}_1$)
mailbox	mailbox.org	no	no ($\texttt{ID}_1$)
mail.de	mail.de	no	no ($\texttt{ID}_1$)
mailfence	mailfence.com	no	no ($\texttt{ID}_1$)
systemli.org[b]	-	-	-
netzguerilla.net[b]	-	-	-

- Test could not be conducted.

[a]E-mails to ID_1 and ID_2 could not be delivered. Provider's MTA reporting unroutable address. Outgoing e-mails from ID_1 to other providers were delivered. It seems to be a survey unrelated mail infrastructure error of this provider. Possibly due to a not unlocked test account. We have not been able to narrow down the reason for this error.

[b]Could not register mail accounts for this provider. Reasons given below:
Mail.ru: We did not want to use our private phone number. **iCloud**: We found no way to register an e-mail account. **Hushmail**: Due to pricing reasons. **Zoho Mail**: Needs further configuration (e.g. own DNS zone). **systemli.org**: Invitation codes required for registration. **netzguerilla.net**: No registration form given. Accounts seems to be manually created after request.

At the end, we want to note that this was only a small survey and without quantifiable data about mail provider market shares it is not rational to draw more conclusion from it. A larger and more detailed evaluation could strengthen these indications which might be relevant for not only Web Key Directory.

6.3 Update Protocol Main Assumption

The Web Key Directory Update Protocol is based on the fundamental assumption that "a mail provider can securely deliver mail to the INBOX of a user (e.g. an IMAP folder)" [33, Sec. 4]. Unfortunately, this assumption is quite unspecific and explicable in different ways. In the following, we want to discuss interpretations of this assumption and their implications on the security of the update protocol. Therefore, we will present an attack vector for the Web Key Directory Update Protocol. The above assumption might be intended to prevent these kinds of attacks. We will first describe the attack vector and then different interpretations of the assumption and their implications. For the attack, we do not assume social engineering or misconfigured ACLs (i.e. Eve can not read Alice's mailbox at the file system level).

6.3.1 Scenario Description

In Figure 6.3, we see a diagram containing three scenarios of how the Web Key Directory Update Protocol can be used. We consider the following scenarios:

1. *one provider.* (S_1) Provider A does manage their own Web Key Directory on their own infrastructure.

2. *WKDaaS direct.* (S_2) Provider A does not manage their own Web Key Directory but uses provider B as a Web Key Directory as a Service provider. Provider A and provider B use only their own infrastructure and exchange e-mails directly (i.e. provider A sends e-mails directly to the corresponding mail server of provider B).

3. *WKDaaS indirect.* (S_3) Provider A does not manage their own Web Key Directory but uses provider B as a Web Key Directory as a Service provider. Provider A uses a preceding e-mail provider C for their ingoing and outgoing e-mails. Provider B does not provide an own mail server but uses provider D for handling the e-mail exchange for the submission-address. Consequently, e-mails between provider A and provider B are not exchanged directly but indirectly over provider C and provider D.

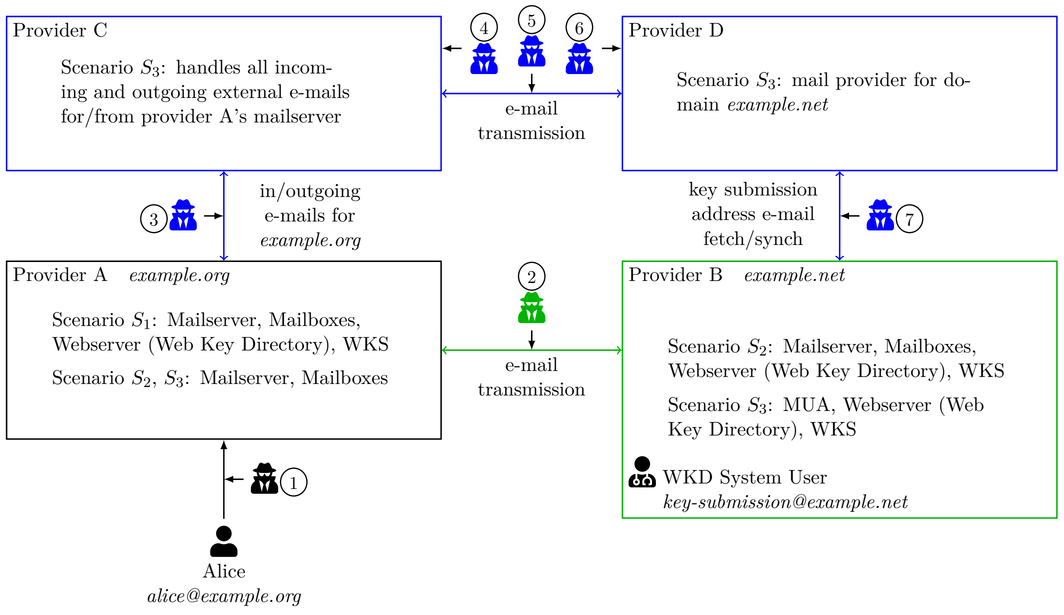

Provider C
Scenario S_3: handles all incoming and outgoing external e-mails for/from provider A's mailserver
Provider D
Scenario S_3: mail provider for domain *example.net*
4
5
6
e-mail transmission
3
in/outgoing e-mails for *example.org*
key submission address e-mail fetch/synch
7
Provider A *example.org*
Scenario S_1: Mailserver, Mailboxes, Webserver (Web Key Directory), WKS
Scenario S_2, S_3: Mailserver, Mailboxes
2
e-mail transmission
Provider B *example.net*
Scenario S_2: Mailserver, Mailboxes, Webserver (Web Key Directory), WKS
Scenario S_3: MUA, Webserver (Web Key Directory), WKS
WKD System User
key-submission@example.net
1
Alice
alice@example.org

Figure 6.3: Three Web Key Directory scenarios with enabled update protocol. Provider A wants to offer Web Key Directory. Provider B can act as a Web Key Directory as a Service provider for provider A (green parts). The Web Key Directory implementation is handled by the WKD System User at provider B's space. Additional blue parts describe a more complex variant how provider A and provider B could be connected. Possible attack points for Eve are marked with Eve (🕵) and arrows on the correspondent path/element. If such a point is only applicable in some scenarios, it is in the correspondent color.

6.3.2 Attack Vector

In the attack vector, Eve wants to publish a self created OpenPGP key with the identity of Alice. Alice is a user of provider A. Eve might or might not require a e-mail address by provider A (like eve@example.org) depending on the setup of the mail servers (e.g. spoof protection measures with SPF [27], DMARC [34]). This attack targets the security goals G_A and G_P and Eve is required to have at least the capabilities of the attacker models A_U and A_{pN}. If she intercepts e-mails or intervenes actively (e.g. downgrade attacks to enforce unencrypted SMTP, IMAP, or POP3 connections), she also needs the active network attacker (A_{aN}) capabilities.

Eve sends a submission message according to the update protocol with a spoofed "From" field impersonating as Alice's e-mail address. According to the update protocol, a confirmation request will be sent back to Alice because the submission seems valid. To succeed, Eve must know the nonce in the confirmation request. Eve could try to read this request as a man-in-the-middle when the e-mail is in transport or when Alice fetches/synchronizes her e-mails from her mailbox at provider A. In case Eve was successful, she can decrypt the request with her created OpenPGP key and can sent a correct confirmation response (with spoofed “From” and “sender” fields impersonating as Alice). Provider B would then accept Eve’s key submission as valid. Consequently, the key would be published, and a success message may be sent to Alice. If Eve is not able to intercept the confirmation request and success message, Alice could take notice of this attack and inform one of the mentioned providers. This however, requires that Alice could interpret and understand both messages and their implications.

6.3.3 Assumption Interpretations

The assumption is explicable in different ways and for the discussion we consider the following interpretations. For the interpretations, we assume that the term “securely deliver” covers at least confidentiality and that the term “provider space” means the entirety of all systems and infrastructure over which a provider has full control.

1. *narrow interpretation.* (I_1) A mail provider can securely deliver e-mails to the INBOX of their own users once the e-mail enters the provider’s space.

2. *closure interpretation.* (I_2) A mail provider can securely deliver e-mails to the INBOX of the recipient’s user. The recipient might be a user of a different provider.

3. *security-conscious providers.* (I_3) The mail provider, who wants to offer a Web Key Directory for their users, and if applicable, their Web Key Directory as a Service provider can guarantee the secure delivery within their own provider spaces. The providers enforce at least confidential connections to and from their provider spaces.

Furthermore, we have two interpretations for the term “securely deliver mail to the INBOX” used in interpretations I_1 and I_2:

a. e-mails are securely delivered to the INBOX of a user on their provider’s mail server.

b. e-mails are securely delivered to the INBOX of a user on their own MUA.

We want to mention that interpretation I_2 might be a very rare and unrealistic scenario in the real world as a provider must ensure security properties of different third party provider.

6.3.4 Implications

We will now discuss the implications of the different assumption interpretations on the described scenarios.

Scenario S_1, Interpretation I_1. In Scenario S_1, provider A manages all necessary mail server, web server, and WKD infrastructure. Web Key Directory Update Protocol related e-mails are sent directly between the provider and their user. Eve could therefore only get in possession of the content of the confirmation request while it is in transfer from Alice's provider to Alice (attack point 1). For this interpretation, it is essential which scope the assumption has regarding the "delivery to the INBOX".

If we apply interpretation $(a.)$, the transport to the INBOX of Alice's MUA is not covered by the assumption. The confirmation request is only delivered securely to Alice's INBOX stored by her provider. We must assume that Eve could get in possession of the confirmation request and could succeed in her attack. Eve might also intercept the delivery of e-mails to Alice to obfuscate her attack as the connection is not secured.

If we apply interpretation $(b.)$, the transport to the INBOX of Alice's MUA is covered by the assumption. This would effectively prevent access by Eve as the confirmation request transport and other e-mails are protected, for example, by Transport Layer Security (e.g. POP3S, IMAPS) between Alice and provider A.

Scenario S_1, Interpretation I_2. Same as for Scenario S_1, Interpretation I_1.

For Scenario S_1, interpretations I_1 and I_2 are the same, as the recipients are always users of provider A.

Scenario S_1, Interpretation I_3. Same implications as for Scenario S_1, Interpretation I_1 $(b.)$.

For Scenario S_1, provider A has no Web Key Directory as a Service provider. The only difference between S_1, I_1 $(b.)$ and S_1, I_3 is that e-mails, which are sent by Alice are securely transmitted to her provider (e.g. SMTPS) (attack point 1), too. (Interpretation $(b.)$ is only concerned about e-mails received by Alice, not sent.) There is no difference for e-mails Alice receives (e.g. confirmation request). In the context of the attack vector, the additional secured connection for Alice's outgoing e-mails is not relevant as Eve does not depend on reading or intercepting Alice's outgoing e-mails.

Scenario S_2, Interpretation I_1. In Scenario S_2, provider A uses provider B as their Web Key Directory as a Service provider and e-mails between provider A and provider B are exchanged directly. Both providers use their own e-mail infrastructure and are therefore considered "a mail provider" for the purpose of the assumption. In this scenario, we have to consider the attack points marked in black or green in Figure 6.3.

The implications for the black attack point 1 are the same as for S_1, I_1.

The green attack point 2 is not covered by the assumption because the transport is between the provider spaces of providers A and B and not within a provider space. Additionally, provider B could not be considered a user of provider A with an INBOX (and vice versa). Therefore, interpretations ($a.$) and ($b.$) are not applicable and do not affect the implications. We must assume that Eve could get in possession of the confirmation request and could succeed with her attack. Eve might also intercept the delivery of e-mails sent from provider B to Alice to obfuscate her attack as the connection is not secured.

Scenario S_2, Interpretation I_2. According to the interpretation, both provider A and provider B can securely deliver e-mails to the users of each provider. This is Alice at provider A and the Web Key Directory User at provider B. Therefore, the transport path between both providers must be secured (e.g. SMTPS). This would effectively prevent access by Eve as the confirmation request transport and other e-mails are protected.

For Eve, only the black attack point 1 remains for which the implications are the same as for S_2, I_1 or S_1, I_1.

Scenario S_2, Interpretation I_3. Provider A connects only to Alice and provider B. Provider B connects only to provider A. Consequently, all of these connections are direct and therefore must have been secured per assumption (attack points 1 and 2).

Eve can not get in possession of the confirmation request.

Scenario S_3, Interpretation I_1. In Scenario S_3, we have a more complex connection between provider A and their Web Key Directory as a Service provider, provider B. Provider A uses provider C as a preceding mail provider (i.e. all ingoing and outgoing e-mails from provider A are handled at provider C before reaching other providers). Provider B does not manage their own mail server anymore, but uses provider D as a mail server as a Service provider for their e-mails. As a consequence, this presents the attack points marked blue in Figure 6.3 to Eve. The green attack point disappears for Eve in this scenario due to the indirect communication between provider A and provider B.

According to this scenario, provider A and provider D are considered mail providers that deliver e-mails to their own users' INBOXes. Consequently, attack point 6 is covered by the assumption and not accessible for Eve.

For attack points 1 and 7, the implications are similar to scenario S_1, I_1. The difference to the mentioned case is that in addition to Alice (a user of provider A) we have the WKD User of provider B that is an e-mail user of provider D.

The assumption interpretation does not include any guarantees regarding provider C's space (attack point 4), or the transmission between provider C and provider D (attack point 5), or between provider A and provider C (attack point 3). We must assume that Eve can access and intercept e-mails including the confirmation request. Therefore, she can succeed with her attack.

Scenario S_3, Interpretation I_2. In this interpretation, the whole e-mail transport between provider A and provider D is secure because these providers are "mail providers" for the purpose of the assumption. Therefore, attack points 3, 4, 5, and 6 are not accessible by Eve.

The implications for the remaining two attack points 1 (between Alice and her provider A) and 7 (between providers B's WKD user and their provider D) are the same as described in case S_3, I_1.

Scenario S_3, Interpretation I_3. Per interpretation provider A and provider B enforce at least confidential connections to and from their spaces in this scenario. Provider A has connections to Alice (attack point 1) and provider C (attack point 3). Provider B has connections to provider D (attack point 7). Consequently, these three attack points are not accessible by Eve.

We have no assumption about the provider spaces of provider C and provider D and the connection between them (attack points 4, 5, 6). Therefore, we must assume that Eve can access and intercept e-mails, including the confirmation request, and she can succeed with her attack.

6.3.5 Summary

Summarizing the above discussion, we showed that the main assumption of the Web Key Directory Update Protocol offers too much room for interpretation. We showed that there exist several attack points in real world relevant scenarios which are not covered in various discussed assumption interpretations. For scenario S_3, only interpretation I_2 (*b.*) completely prevents the attack where an attacker can publish a self created OpenPGP key with the identity of another user (see attack description in Section 6.3.2). For scenarios S_1 and S_2, only about half of the interpretations prevent this attack. As a consequence, an attacker could be able to submit OpenPGP keys for other users in several e-mail infrastructure scenarios. Hence, security goals G_A and G_P are targeted. The attacker needs at least the capabilities of A_U and A_{pN}. If the attacker also intercepts e-mails or intervenes actively (e.g. downgrade attacks to enforce unencrypted SMTP, IMAP, or POP3 connections), the active network attacker (A_{aN}) capabilities are also needed. The results of the discussion are condensed in Table 6.4 below.

Table 6.4: Evaluation of the Web Key Directory Update Protocol main assumption discussion for different scenarios and interpretations. In each cell the attack points (see Fig. 6.3) are recorded which are exploitable by Eve for the given scenario and interpretation.

	S_1	S_2	S_3
I_1 *a.*	1	1, 2	1, 3, 4, 5, 7
I_1 *b.*	-	2	3, 4, 5
I_2 *a.*	1	1	1, 7
I_2 *b.*	-	-	-
I_3	-	-	4, 5, 6

- No attack points accessible by Eve.

The assumption should point out in which context the update protocol is intended to be used securely. Therefore, the meaning of the term “secure delivery” should be defined and the concrete security aspects should be stated. We assumed at least confidentiality must be covered as otherwise an attack could succeed easily. The context of the delivery (i.e. to the user’s INBOX on the mail server or the MUA and which users are actually meant) should be clarified as well, as this has a major impact on the security implications. We want to mention the differences for the implications in Table 6.4 between interpretation (*a.*) and (*b.*). (They are not differentiated for I_3 because I_3 implicitly assumes secure delivery to the user’s INBOX and the next MTA.) The need to consider the case of multiple different providers is well illustrated in the amount of possible attack points for scenario S_3. The assumption should point out more clearly which providers are committed to and covered by the assumption as the term “[a] mail provider” is not sufficient for scenarios with multiple providers. If the specification authors do not intend the update protocol to securely perform in such complex mail infrastructure scenarios as seen in S_3, the assumption should at least be accompanied by a corresponding warning about the possible security implications.

Only interpretation I_2 (*b.*) prevents this attack in all described scenarios. However, as already mentioned in Section 6.3.3 (Assumption Interpretations), we consider only interpretations I_1 and I_3 as realistic in the real world application as interpretation I_2 requires a provider to ensure the secure delivery to another provider’s user’s INBOX. Considering scenarios S_1 and S_2, the evaluation indicates that the Web

Key Directory Update Protocol can securely be used without being susceptible to this attack on key authenticity if e-mails are always securely delivered to the user's INBOX on the MUA and between the e-mail provider and their Web Key Directory as a Service provider.

6.4 Reference Implementation

In this Section, we will address the server and client reference implementations of the Web Key Directory method which are developed by the GnuPG Project. We noticed that the implementation adds additional headers to the e-mails which are not specified and cause the implementations to not behave according to the specification anymore. Furthermore, we will look into errors in the signature creation and processing behavior of the implementations concerning the confirmation response. However, this section is not to be seen as a full review of the reference implementation.

We want to mention that Appendix B (WKD Update Protocol Run) contains all messages emerging from a complete and successful OpenPGP key submission using the Web Key Directory Update Protocol. We have tested the implementation of the current upstream version 2.3.6 and of the current LTS version 2.2.35. Both versions show the same behavior.

We also want to mention that according to the source code both versions implement the Web Key Directory specification in version 3 [20, tools/gpg-wks.h:28]. After examining the differences between draft versions 2, 3, and 13, we can acknowledge that there are no breaking changes in the format of the messages (since beginning of version 2) and they can be considered compatible with client implementations based on at least draft version 2 [54].

6.4.1 Additional Headers

During the examination of the Web Key Directory method we noticed the occurrence of additional header fields which were added by the client and server implementation to all generated e-mails.

Wks-Phase The confirmation request and the optional key publication success e-mail contain the `Wks-Phase` header. The value is either "confirm" for the confirmation request or "done" for the publication success e-mails.

We were curious about the relevance and reason this header was introduced in the reference implementation as it is not specified and not secured by any means (as all e-mail headers). It turned out that the reason was to help the plugin Enigmail for handling OpenPGP in the MUA Thunderbird with the identification of these messages and was added in version 2.1.19 in 2017-03-01 [20, NEWS:1579]. According to comments in the source code, it was not meant to use this header as a secured feature [20, tools/gpg-wks-server.c:1009]. In the worst case an attacker could add, remove, or change this header or its value to affect the proper detection of tools which rely on it.

We want to mention that this header is not interpreted by the reference implementation and apparently not used in Enigmail [16] or in Thunderbird [60] anymore. Therefore, this header must be considered as superfluous and should be removed because its purpose is not relevant anymore – at least with the information we have.

Wks-Draft-Version All e-mails generated by the reference implementation contain the `Wks-Draft-Version` header.

This header was introduced in 2016-09-29 as a preparation for draft version 2 [20, commit: 33800280d]. The idea for the introduction according to the commit message was to support a way to handle a new message format (versions ≥ 2) without breaking existing (version 1) implementations. Old implementations would not add or interpret this header field and newer implementations could decide a format version fallback based on the value and existence of this header. However, the `Wks-Draft-Version` header was not considered in the specification process and is still not specified.

The reference implementation checks if the header was added and that the value is at least 2 [20, tools/wks-receive.c:453/280]. If this does not apply, the messages are considered to be in draft version 1.

A problem emerges from the incompatibility of the confirmation request which was fundamentally changed. In version 1, the e-mail was encrypted and optionally signed. Beginning with version 2, it is signed and contains a text and an encrypted part [53]. As a consequence, the reference implementation does not operate specification-compliant and in a way that would break other implementations which are only based on the specification. However, a submission e-mail compliant to the current specification version (but without the Wks-Draft-Version header) would trigger a confirmation request in the incompatible draft version 1. The format of the submission e-mail and the confirmation response seems to be backward compatible as they enforce aspects ("MUST") which where prior marked as "SHOULD" (see RFC 2119 which defines Notational Conventions which are used in RFCs).

The `Wks-Draft-Version` header is, similar to the Wks-Phase header, not protected and could be altered. An attacker could perform downgrade attacks via this header. Currently, there only exist an "old" message format (prior version 2) and the "new" format added with version 2. An attacker could remove the header or change the value to 0 which would result in the reference implementation considering a new message format as an old one. This would, for example, mean that the signature in the confirmation request is not required anymore. If we consider this header as a mechanism to add versioning to the Web Key Directory Update Protocol messages and the assumption that there might be further security relevant format changes to the protocol, this problem would aggravate.

However, the header is not specified and the specification authors should address this issue either with changes to the reference implementation or by considering this header in the specification process. It should be examined whether it is possible to provide sufficient integrity for the transmitted version number, for example, by binding it to the used key(s) through signatures.

6.4.2 Irregularities in Message Generation and Processing

We performed a full and successful run of the Web Key Directory Update Protocol and compared the generated messages with the specification. To our surprise, the confirmation response message was only encrypted and not encrypted and signed as specified, although it has been considered valid by the server implementation. Due to this finding, we decided to examine the reference implementation further in regard to specification compliance.

confirmation request checking The confirmation request is specified as a signed multipart MIME message containing a text and an encrypted WKD format part since version 2 [53, 31, 33]. This signature is easily accessible as it is a MIME part which is added to both parts mentioned above. We tested the implementation by checking its behavior when processing confirmation requests with a valid, manually invalidated, and removed signature part. Interestingly, all these messages were accepted and confirmation response messages were created. The only difference in behavior was printed warnings.

The reason for this behavior can be directly seen in the function `verify_signature` [20, tools/wks-receive.c:147ff] which is called if the parser collects signature parts. The function actually only calls the `gpg` executable with arguments as seen in Listing 6.5 and checks the return code. The return code handling is defective as in both cases (successful and non-successful exit of gpg) the function only prints messages, frees the argument array, and returns. Consequently, the verification checking is useless and has no impact on the following program flow. This problem is underlined by the “Fixme: Verification result is not used” message which is always printed when the function is called. The problem exists since 2016 [20, commit:

5d6c83dea]. Three months later, the above debug message was added, and the commit message contains an additional note: “There are still a lot of FIXMEs - take care.” [20, commit: 33800280d].

```
gpg --no-options --batch ( | --verbose) --enable-special-filenames --status-fd=2 --always-
    trust --verify -- -&@INEXTRA@ -
```

Listing 6.5: Call signature of the gpg executable by the `verify_signature` (lines 147ff) function in tools/wks-receive.c. (x | y) means argument x or argument y. The argument `-&@INEXTRA@` is not relevant for our considerations.

confirmation response generation The confirmation response message must both be signed by the signing key of the OpenPGP key which should be published and encrypted to the OpenPGP key associated with the provider’s submission e-mail address using the PGP/MIME Combined format [31, 33]. However, the PGP message generated by `gpg-wks-client`, which can be seen in Listing B.6 in the appendix, is only encrypted and not signed. We could narrow down the reason in the source code to the corresponding `encrypt_response` function [20, tools/gpg-wks-client.c:1225ff] which misses signature creation completely. The function actually only calls the `gpg` executable with arguments for encryption and checks the return code. Apparently, it misses the `--sign` argument which leads directly to an encrypted but not signed message. The program call with arguments is given in Listing 6.6. According to the git history, the mistake is present since 2016.

```
gpg --no-options ( | --quiet | --verbose) --batch --status-fd=2 --always-trust --armor -z0
    (--auto-key-locate=clear,local | --auto-key-locate=clear,wkd,dane,local) --recipient
    addrspec --recipient fingerprint --encrypt --
```

Listing 6.6: Call signature of the gpg executable by the `encrypt_response` (lines 1225ff) function in tools/gpg-wks-client.c. (x | y) means argument x or argument y. `addrspec` is the e-mail address of the recipient and the `fingerprint` is the fingerprint of the corresponding recipient’s key.

The confirmation response is sent unencrypted if no encryption key for the provider’s submission e-mail address is found [20, tools/gpg-wks-client.c:1508ff]. This behavior has been present since 2016 too, and interestingly it appears to be intentional due to the comment in the source code. This mistake appears to be a consequence of changing the specification (draft version 1 “`SHOULD` be encrypted”, starting version 2 “`MUST` be encrypted”) but not adapting the implementation.

confirmation response checking Besides the aspects regarding the generation of the confirmation response message, this message is also processed by the server implementation and was accepted although it does not comply with the specification. Confirmation responses which are only encrypted and not signed are accepted. Confirmation responses which are encrypted and signed (as required by the specification) are not accepted. We narrowed down the reason for this behavior.

Incoming messages are parsed regarding their MIME structure and data, which is for example, encrypted or signed, is collected and then processed by defined callback functions [20, tools/mime-parser.c:676ff, tools/wks-receive.c:409ff]. OpenPGP uses the sign-then-encrypt approach. This means the plain text is first signed and then encrypted and obviously the validation must first decrypt the cipher to get at the plain text and signature. Consequently, signed-and-encrypted ciphers look like only encrypted ciphers before decryption and are handled by the same callback function in the implementation [20, tools/wks-receive.c:197ff/72ff]. The function tries to decrypt the collected cipher by calling `gpg` with the arguments given in Listing 6.7. The decrypted cipher (the `stdout` output of the program call) is written into a data stream.

```
gpg --no-options --max-output=0x10000 --batch ( | --verbose) --always-trust --decrypt --
```

Listing 6.7: Call signature of the gpg executable by the `decrypt_data` (lines 73ff) function in tools/wks-receive.c.

A stream is a sequence of data where bytes can be added over time, and a certain amount of bytes are processed once at a time instead of all at once. As an example, we can picture a file descriptor which points to a file and we can thereby read characters from the file. After we read a character, the next read will give us the next character. We can also set the read position to the beginning of the file without the need to close and open the descriptor again. The streams used in the implementation have at least a beginning, an end, and a current position which indicates the last read byte. It is possible to set the current position back to the beginning which is called “rewinding”.

If the called `gpg` program exits with a non-zero exit code (non-success exit), the function prints an error message and jumps to the “leave” flag where the program call arguments are freed and then it returns. If the program exits successfully, the stream is rewinded prior to the arguments freeing and returning.

The observed problem is caused by this improper handling of the exit code and the fact that the cipher is decrypted to the stream even if the signature is not present or could not be properly validated. The stream is not cleared in the case of a non-success exit. The only difference is that the current read position of the stream is set to the beginning if no error occurs. The reason why the confirmation response is not accepted in the error exit case is that the parser, which tries to parse the stream (e.g. to determine contained WKD format data) finds no data due to the read position of the stream. As a side note, this bears the risk that unaware added code to print the contents of the stream, for example, debugging purposes (which rewinds the stream afterwards again), indicates the stream is empty and makes the contents afterwards (through rewinding) accessible again which could directly lead to the acceptance of the message.

Interestingly this improper handling has been present since the function was added in 2016 [20, commit: 5d6c83dea]. This mistake can not be ascribed to a not adequately updated implementation of an older draft version because the confirmation response e-mail must be signed (and should be encrypted) since version 0 (and must be encrypted since version 3) [32, 31].

In addition, this function has a design flaw: The `gpg` program is not called with proper arguments regarding the verification of the signature. This is currently also not possible because the corresponding to be published key is not handed over to the function in which the signature should be verified. Currently, the called `gpg` program checks implicitly against the keys which are imported in the executing user's local keyring. The to be published key is not intended/specified to be present in this keyring. According to the man page of `gpg(1)`, signature verification needs knowledge about the key which a signature should be checked against and only interpreting the return value is not suitable to properly verify signatures.

The behavior of the reference implementation according to the confirmation response checking can be summarized as follows:

If the confirmation response is encrypted to the key of the submission address[1] and not signed (not as specified by the draft), the reference implementation accepts the message. The called `gpg` executable exits with success and the decrypted cipher is written to the stream. The stream is rewinded by the reference implementation.

If the confirmation response is encrypted to the key of the submission address and is signed by the target key (as specified by the draft specification), the reference implementation will not accept the message. The called `gpg` executable will fail with a non-success exit return code while trying to verify the signature because the executable has no knowledge about the target key which is required for proper signature verification. The executable checks implicitly against the imported keys in the executing user's keyring, but it is not specified or intended that target keys are in this keyring. Nevertheless, the decrypted cipher is already written to the stream. Due to the non-success exit return code of the executable call, the stream is not rewinded by the reference implementation.

[1]For the sake of completeness, it is theoretically possible to encrypt the message to any public key where the corresponding secret key is imported in the executing user's keyring (i.e. WKS user's keyring) because the `gpg` executable checks implicitly against these keys.

If the confirmation response is encrypted to the key of the submission address and signed by any private key where the corresponding public key is imported in the executing user's keyring, the reference implementation will accept the message (which is in conflict with the draft specification). The called `gpg` executable will exit successfully because it was able to verify the signature. The verification was not done against the target key but against an imported key in the executing user's keyring. The decrypted cipher is written to the stream. The stream is rewinded by the reference implementation.

Thus, the reference implementation will not accept draft-compliant confirmation responses, but will accept many other confirmation responses with missing or wrong signatures.

6.4.3 Irregularities in the WKD Format Checking

During the examination in the source, we also noticed incomplete implementation of Web Key Directory format key-pair checks. This concerns the `sender` and `address` fields in the confirmation request and confirmation response checking and the `nonce` field in the confirmation response checking.

For the `sender` field, it is only checked for both confirmation request and confirmation response whether it exists and contains a valid e-mail address according to RFC 822 [20, common/mbox-util.c:166ff, tools/gpg-wks-client.c:1485ff, tools/gpg-wks-server.c:1532ff]. It is not checked that the e-mail address corresponds with the one given in the `From:` header as specified.

For the `address` field, it is only checked for both confirmation request and confirmation response whether it exists and contains a RFC 822 compliant e-mail address [20, common/mbox-util.c:166ff, tools/gpg-wks-client.c:1472ff, tools/gpg-wks-server.c:1545ff]. It is not checked that the e-mail address corresponds with the e-mail of the to be published User-ID in the confirmation request and that the e-mail address matches the value of the `address` field of the confirmation request in the confirmation response as specified. In fact, the missing check of the correspondence of the address field value in the confirmation response will make an attack possible to successfully submit any OpenPGP key for a supported domain to a Web Key Directory. We will describe the attack in Section 6.4.4.

Interestingly the `nonce` value is not checked properly as specified to be the same as the nonce of the confirmation request. It is only checked whether it exists and the length is at least 16 bytes long [20, tools/gpg-wks-server.c:1556ff]. Furthermore, a file path constructed from the nonce has to exist [20, tools/gpg-wks-server.c:1382ff]. The file path is constructed basically as a concatenation of strings as shown in Listing 6.8 [20, common/stringhelp.c:607]. The function used for the concatenation is used in several other places in the repository. This might allow path traversals like the one described below to be exploited in other parts of the

GnuPG code, too if the input strings used for the concatenation are not properly checked.

```
file_path := opt.directory / domain / "pending" / nonce

Example path traversal:
/var/lib/gnupg/wks / example.org / pending / ../../../../../../../to/any/file
```

Listing 6.8: Construction of the file path to access the pending OpenPGP key. An example is shown for a simple path traversal. The file pointed to must be a PGP file with a User-ID corresponding to the `address` field value.

It is possible to modify the nonce in such a way that it can point to any file and be at least 16 characters long. To do that, we use the fact that on Unix-like systems (e.g. Linux) a directory entry named "`..`" always points to the parent directory, except for the root (top-most) directory, where "`..`" points to the directory itself. Due to the implementation, the pointed file must be a PGP file with a User-ID which corresponds with the `address` field value[20, tools/gpg-wks-server.c:1389ff]. We found no scope to utilize that path traversal to extract secret data from the server system, only public OpenPGP keys. However, we will utilize this path traversal to generalize the attack we will describe in Section 6.4.4 to additional domains beside the domain of the attacker controlled e-mail address.

All submitted keys are stored in the pending directory for a given domain. This directory is only accessible by the user under which the WKS tool is running. It is part of the same directory tree that is usually served by the web server to enable the Web Key Directory Discovery Protocol. If the web server runs under the same user as the WKS tool, the web server is able to serve these files as well. If additionally the web server has directory indexing not disabled, it is trivial to confirm each of these pending keys because the nonce is the filename, and the e-mail address is in the stored key, and all these files would be easily accessible through the web.

For the sake of completeness, we want to mention that the domain part is wrongly extracted from the address value because the part after the first occurrence of an "`@`" character is interpreted as the domain [20, tools/gpg-wks-server.c:1379ff]. This is not true for a valid e-mail address like `"user@something"@example.org` [51, Sec. 3.4.1]. We have no evidence that this is a security relevant flaw since "`"`" and "`@`" are not allowed in domain names [29, Sec. 2]. It only affects availability for these users.

6.4.4 Attack on the Reference Implementation

In this section, we describe an attack on the reference implementation which allows us to successfully publish OpenPGP keys for any e-mail address of the domains served by a Web Key Directory. We only assume that the reference implementation is used and that an attacker has an e-mail address at the provider which offers the Web Key Directory. We utilize the path traversal described in Section 6.4.3 to extend the attack scope on e-mail addresses where the domain differs from that of the attacker controlled e-mail address. Furthermore, we will show that by performing this attack we can accomplish a denial-of-service for the Web Key Directory Update Protocol by overwriting the published submission-address OpenPGP key necessary for utilizing the protocol.

All generated messages and OpenPGP key material for this attack are given in Appendix C.

Scenario description The scenario on which the attack is based is generic. It only requires the reference implementation to be used. For the attack, we assume Mallory (A_U) is in possession of the e-mail address `mallory@example.org`. She wants to tamper the OpenPGP key for Alice who uses the e-mail address `alice@example.org`.

Furthermore, we assume that the Web Key Directory Provider manages both the domains example.org and example.net. For the attack, it is not necessary that multiple domains are managed, but we will use this to demonstrate the attack expansion on other domains than the domain of Mallory's e-mail address.

We also want to point out that the submission-address keys are treated like any other key in the Web Key Directory by the implementation and can be managed by the Web Key Directory Update Protocol as well.

For better understanding, we picture the directory structure which is used by the reference implementation in Listing 6.9.

```
/var/lib/gnupg/wks/
|-- example.net
|   |-- hu
|   |   `-- 54f6ry7x1qqtpor16txw5gdmdbbh6a73
|   |-- pending
|   |   `-- o3euik3tr3d6rjwrq4i7o1gg77o48mn1
|   |-- policy
|   `-- submission-address
`-- example.org
    |-- hu
    |   `-- 54f6ry7x1qqtpor16txw5gdmdbbh6a73
    |-- pending
    |   `-- 7mpzp4cgkr5uy3bzaw4pqggwga8ksfz9
    |-- policy
    `-- submission-address
```

Listing 6.9: The directory structure for a Web Key Directory managing the example.org and example.net domains. In directory `hu/` the successfully submitted OpenPGP keys are stored named by their WKD hash for publication (in this example only the key for the submission-address). In directory `pending/` the submitted keys which are not yet confirmed are stored named by the nonce.

Attack description To perform the attack, we have to conduct the following steps:

1. Generate an OpenPGP key with two User-IDs. One for a self-controlled e-mail address (e.g. mallory@example.org) and one for the to be attacked user (e.g. alice@example.org). The domains of these e-mail addresses can differ. The only requirement is that the server handles the chosen domains.

2. Mallory generates the submission e-mail for the User-ID mallory@example.org using `gpg-wkd-client`.

3. Mallory replaces the public key in the PGP message with the public key which contains both User-IDs. This means the PGP message must be decrypted with the self generated key, the contained public key is changed, and the message is encrypted again. This is necessary because the `gpg-wkd-client` tool generates a submission e-mail for only the requested User-ID (as specified).

4. Mallory sends the edited submission e-mail to the provider.

5. The server sends one confirmation request e-mail to Alice and one to Mallory (to each User-ID). The second part of these e-mails are encrypted with the key Mallory has submitted. Alice is therefore not able to decrypt it. Nevertheless, Alice can read the header including the `Subject:` header and the informational text in the first part.

6. Mallory generates the confirmation response.

7. Mallory replaces the value of the `address` field which originally contains Mallory's e-mail address `mallory@example.org` with `alice@example.org`. This means the PGP message must be decrypted with the self generated key, the containing field is changed, and the message is encrypted again. In case Mallory wants to target a user with a different domain, Mallory prefixes the relative traversal path to her pending key to the nonce. For example, a nonce prefix could be `../../example.org/pending/`.

8. Mallory sends the edited confirmation response to the provider.

9. The server publishes the submitted public key with the User-ID of Alice. If for the targeted e-mail address a key is already published, the key is replaced. The server sends a confirmation e-mail to `alice@example.org` to notify her about the key publication. The confirmation e-mail is encrypted to Mallory's submitted key and Alice is not able to decrypt it. Nevertheless, Alice can read the header including the `Subject:` header field.

Evaluation The attack targets the security goals "Authentic keys" (G_A) and "Privacy considering key exchange" (G_P). The attack is based on two security flaws:

1. Wrong handling of the confirmation response checking, especially the `address` value and `nonce` value checking.

2. A path traversal attack on the `nonce` value.

For each User-ID, the complete key is stored in the pending directory of the corresponding domain named by a different nonce. The implementation does not check the value of the nonce in the confirmation response against the one send in the confirmation request but decides the correctness based on the existence of a file at the file path generated from the nonce. The decision on which User-ID is finally published is based on the value of the address field which is also not properly checked. Because the complete key (so all User-IDs) is stored for each of the User-IDs we can change the address field value to a different e-mail address and the server will extract exactly this User-ID for the publication and stores it in the `hu/` directory.

The path traversal is necessary in order to attack also User-IDs which have a different domain than the attacker's e-mail address. This is based on the fact that the domain is part of the created file path (see Listing 6.8) and the implementation would otherwise not find a file named after the nonce for this domain because the nonce is different for each User-ID. We can circumvent this problem by prefixing a path to the domain which is used for the attacker controlled e-mail address (like `../../example.org/pending/`). We know that there is a complete key (containing all User-IDs) stored named after the nonce which was sent to the attacker in the confirmation request. By combining the changed `address` and the prefixed `nonce` we can therefore publish also tampered User-IDs for domains which are different to the domain used in the attacker-controlled e-mail address.

Conclusion The implications of the attack are serious as it is possible to tamper each key of a Web Key Directory for all served domains. If a key is already present, it can be overwritten by the attack. The overwriting of the key associated with the submission address leads to a denial-of-service as the Web Key Directory Update Protocol requires this key because the e-mails to the server are secured by it.

This poses a problem to all Web Key Directories managed by the reference implementation. Especially for Web Key Directory as a Service providers, this could be a even more severe problem because they serve Web Key Directories for multiple domains.

As a consequence, it is highly questionable whether trust in these keys can still be assumed. Nevertheless, it is theoretically possible to detect such an attack at user level as the user will receive the success message from the provider for a key submission which was not ordered by themself.

6.4.5 Summary of our Findings in the Reference Implementation

We will now summarize our findings for the reference implementation. For a better overview we also present the results in a concise form in Table 6.10.

We described in Section 6.4.1 that the reference implementation makes usage of two headers in their messages which are not specified in the draft. The first header, `Wks-Phase`, was introduced to help software like Enigmail to handle Web Key Directory e-mails. The header is not relevant anymore as far as we could figure out because the software mentioned in the commit message does not use it anymore.

The second header, `Wks-Draft-Version`, is used by the implementation to decide which protocol draft version is used. We showed that this is problematic in two ways: The header is not protected and an attacker could simply perform downgrade attacks on it to ease potential attacks. Furthermore, the header is not specified in the draft and the reference implementation assumes messages without the header to follow a draft version prior to version 2 which is not compatible anymore. This behavior would therefore break compatibility with other specification-compliant implementations.

In Section 6.4.2, we addressed the generation and processing of messages in the reference implementation. We showed that the implementation does not act as specified in several places and includes several points which are not properly implemented since years.

We pointed out that the signature checking for the confirmation request as implemented is useless and has no impact on the following program flow, thus permitting messages without valid signatures. Furthermore, we showed that the implementation creates only encrypted and not encrypted and signed confirmation response messages. If no OpenPGP key is found to encrypt the message to the implementation even sends the message unencrypted. Both conflicts with specified behavior. In fact, the implementation can not parse properly encrypted and signed confirmation responses.

In Section 6.4.3, we showed that besides the generation and checking of the messages in the Web Key Directory Update Protocol the Web Key Directory format (the key-value pairs) are mostly not checked properly. The `sender` and `address` fields need only to exist and contain valid e-mail addresses for the purpose of RFC 822. It is not checked the values in the confirmation response correspond with the values of the confirmation request.

We showed that improper checking of the `address` field will make it possible to publish OpenPGP keys for User-IDs other than the person who submits them.

The `nonce` field is also not properly checked against the value of the confirmation request. The nonce is used as a part of a string concatenation which leads to a file path where the implementation assumes the pending key to be be stored. The nonce is seen as valid if a key with the User-ID from the `address` field is found at that file path. We pointed out that this behavior is prone to path traversal attacks. The usage of this vulnerability made it possible to generalizes the attack described in Section 6.4.4 on other domains than the one used in the attacker controlled e-mail address.

In Section 6.4.4, we then presented an attack on the Web Key Directory Update Protocol which makes it possible for an attacker to publish OpenPGP keys for any e-mail address of a domain which is handled by a Web Key Directory provider. For the attack, we only assume that the reference implementation is used. Furthermore, the attacker must be in possession of one e-mail address handled by the provider. For the attack, basically only the OpenPGP key with both the own and the targeted OpenPGP User-IDs has to be submitted and the `address` and (if the target User-ID uses a different domain) the `nonce` have to be changed.

This attack is severe in the context of the level of trust which can be gained for the exchanged OpenPGP keys for Web Key Directories provider using the reference implementation.

6.5 Considerations on Further Security Goals

In this section, we will address the remaining security goals as defined in Chapter 4 (Security Definitions). We have already addressed the security goal "Authentic keys" (G_A) extensively in above sections and will not consider it further in this section.

We will evaluate if the specification properly specifies the approach for the underlying operations (like key publication, key revocation, or key exchange termination) of the security goals and if the reference implementation properly implements these, for example as part of the Web Key Directory Update Protocol.

Table 6.10: Current implementation status of the reference implementation (version 2.3.6) based on Web Key Directory specification draft version 14.

	submission	confirmation request	confirmation response
message format			
generation	complete	complete	partial[a][b]
checking	complete	partial[c]	defective $\lightning^1$
key-value check			
`type`	-	checked	checked
`sender`	-	partial[d]	partial[d]
`address`	-	partial[e]	partial[f] $\lightning^2$
`fingerprint`	-	checked	-
`nonce`	-	checked	partial[g] $\lightning^3$

-	Not applicable.
$\lightning^1$	Message parsing fails for specification-compliant confirmation responses.
$\lightning^2$	Utilized in the attack to publish attacker-controlled OpenPGP keys for any e-mail address of the domains a WKD serves.
$\lightning^3$	Path traversal possible. The file pointed to must be a OpenPGP key with a User-ID containing the `address` e-mail address.

[a] Only encrypted and not signed and encrypted.

[b] If no key was found, the confirmation response is send unencrypted.

[c] The signature is not properly verified. The verification implementation is effectively useless.

[d] It is not checked that the “sender” field matches the “From” field e-mail address.

[e] It is not checked that the “address” field matches the e-mail address of the to be published User-ID.

[f] It is not checked that the “address” field matches the “address” field of the confirmation request.

[g] It is not checked that the given nonce corresponds with the one in the confirmation request. The nonce is used as the last part of a string concatenation to generate a file path.

6.5.1 Key Exchange Termination (G_T)

The security goal we address in this section requires that a key owner must be able to remove and therefore terminate the key exchange for their key.

We do not have to deeply consider the Web Key Directory Discovery Protocol for this security goal because key exchange termination is not in the scope of this protocol. It only discovers and requests published keys. If the key is removed, it could not be discovered anymore.

The first sentence in the description of the Web Key Directory Update Protocol describes it as an automated way to put keys into the key directory [33, Sec. 4]. It is not mentioned that the update protocol would be intended to remove such submitted keys. This is also reflected in the protocol flow: The update protocol is initiated by the submission e-mail, followed by the confirmation request, and the confirmation response. After the confirmation response, the server publishes the key. There are neither extra messages nor special key-value pairs in the Web Key Directory format which empowers the protocol to handle such a termination request.

Consequently, we have to conclude that the termination of key publications is not possible with the Web Key Directory Update Protocol. The proceedings for manual interventions of administrators are, however, clear and in our view obvious because a successful key exchange termination is achieved by simply deleting the key from the publication source (e.g. the `hu/` directory of the web server).

Reference Implementation Although the specification does not consider the removal of published keys via the update protocol, the reference implementation could behave differently. Therefore, we also check the reference implementation regarding key exchange termination.

The `gpg-wks-server` implementation contains a command line flag `--remove-key` [20, tools/gpg-wks-server.c:398ff]. It takes a User-ID, computes the file path to the `hu/` directory, and removes the correspondent key [20, tools/wks-util.c:1162ff]. This operations needs direct file system access. It is not an operation which utilizes the update protocol or introduces other mechanism which are not specified.

Interestingly we find the same command line flag with the same behavior in the `gpg-wks-client` implementation as in `gpg-wks-server` [20, tools/gpg-wks-client.c:-391ff]. This is odd because the client implementation is intended to be used at clients and not by the server itself. Clients have no direct access (i.e. file system access) to the Web Key Directory and the operation should always fail because it is dependent on direct access.

The existence of the `--remove-key` command line flag is not in conflict with the specification. However, there is no (automated) way to make use of it according to the specification, except for manual intervention by the administrator.

6.5.2 Key Revocation (G_R)

The security goal “Key revocation” demands that if a user requests a key for which a valid revocation signature has been submitted, this revocation signature must be delivered, too. The purpose is that it must be possible to revoke published keys and that users of Web Key Directory are enabled to receive such revocations.

We do not have to deeply consider the Web Key Directory Discovery Protocol for this security goal because key revocation is not in the scope of this protocol. The purpose of the discovery protocol is to provide a specified and secure way to discover and request OpenPGP keys via a known e-mail address and HTTPS. The discovery protocol mentions that servers may sent revoked keys along with the new key and that clients may receive revoked or expired keys [33, Sec. 3.1]. The consequences for a served key which is for example revoked (regardless whether the served key is actually only a revocation certificate or a key which contains revocation signatures) are, however, not in the scope of the discovery protocol and are explicitly delegated to the client implementations [33, Sec. 3.1].

Similar to the key exchange termination case, the specification of the Web Key Directory Update Protocol does not hold any additional messages or special key-value pairs for key revocation. However, we need to discuss whether key revocation has to be treated separately, or whether it is already implicitly specified by the submission process.

For this, we want to consider the revocation of OpenPGP keys first briefly. It is possible to revoke OpenPGP keys or individual sub-keys via revocation signatures [7, Sec. 5.2.1]. These signatures can be appended to the corresponding key or saved separately as a revocation certificate. Thus, revocation certificates contain only an OpenPGP signature package and no User-ID packets, nor public keys. A revocation signature packet or certificate is valid if it is signed by the key being revoked, or by an authorized revocation key [7, Sec. 5.2.1]. Revoked keys must not be used further [7, Sec. 5.2.1]. In Listing 6.11, we have presented a revocation certificate.

```
# cat alice.rev | gpg --show-key
rvs?        17682DD1D31ACB7B 2022-06-10 [User ID not found]
     reason for revocation: No reason specified

# cat alice.rev | gpg --list-packets
```

```
:signature packet: algo 22, keyid 17682DD1D31ACB7B
       version 4, created 1654871612, md5len 0, sigclass 0x20
       digest algo 8, begin of digest e4 98
       hashed subpkt 33 len 21 (issuer fpr v4 2
           BE93B1EC9125C1FF6C20FE717682DD1D31ACB7B)
       hashed subpkt 2 len 4 (sig created 2022-06-10)
       hashed subpkt 29 len 1 (revocation reason 0x00 ())
       subpkt 16 len 8 (issuer key ID 17682DD1D31ACB7B)
       data: [256 bits]
       data: [252 bits]
```

Listing 6.11: Revocation certificate for the OpenPGP key with the User-ID `Alice <alice@example.org>` identified by the fingerprint `2BE93B1EC9125C1FF6C20FE717682DD1D31ACB7B`. The first output is the interpretation of `gpg` and the second is the view on the packet contained in the certificate. The revocation certificate was issued by the (primary) secret key with key ID `17682DD1D31ACB7B`.

In the following, we consider whether the specification covers revocation via revocation certificate or keys which contain revocation signature packets.

Revocation certificate The protocol is initiated by the submission e-mail, followed by the confirmation request and confirmation response. The submission e-mail must contain “the ASCII-armored transferable Public Key Packets as defined in RFC 4880” [33, Sec. 4.2]. Consequently, signature packets can not be submitted via a protocol-compliant submission e-mail because they are not public key packets [7, Sec. 5.2]. Besides that, a revocation certificate lacks a User-ID and a public key and the server could not send a compliant confirmation request because the second part of it must be encrypted. However, a revocation certificate should not have to be approved by the user as with a normal public key submission. Due to the nature of revocation certificates, their possession implies already the authorization to revoke the corresponding key.

If we assume that the update protocol would accept revocation certificates, the server should verify the revocation certificate prior to publication to ensure to have received a valid certificate. For this validation, a confirmed key must be present at the server. Without a confirmed key, the server can not determine whether the submitted revocation certificate belongs to a key in possession of a provider's user. Without a validity check, an attacker could submit revocation certificates for keys which do not belong to rightful keys at large scale which could affect service maintainability and availability.

Consequently, it becomes apparent that the Web Key Directory Update Protocol needs separate specification to handle revocation certificates. This raises several questions: What should be done if the server receives a revocation certificate for a key which was not already published in the past? What should happen if the server can not verify a revocation certificate? Should the revocation signature packet be appended to an already published key or should it replace the key?

OpenPGP key containing revocation signature packets Besides the revocation certificate, revocation signature packets could also be embedded in a OpenPGP key. Similar to the above case, the revocation signature packets of a key can not be added to a compliant submission e-mail. Besides the conflict with the Web Key Directory specification, a confirmation request would also be in conflict with the OpenPGP specification because a revoked key must not be used further. The second part of the confirmation request must always be encrypted to this (than revoked) key. The server could check the validity of the revocation signature packets against the submitted key, but we do not know for sure if the key is valid and therefore if the revocation is valid. If the server publishes this unverified revoked key, an attacker could submit fake revoked keys at a large scale which could affect service maintainability and availability.

This indicates that the specification also does not cover revocation with embedded revocation signatures in the submitted key.

To summarize, we can conclude that the specification does not allow the revocation of keys. This is not only applicable to the automated update protocol but it also affects manual interventions of administrators as they do not know how to properly handle this case. For example, the revocation certification could be added to the already published user's key or the key could be replaced by the revocation certificate. As a result, key revocation is also not clearly defined by manual actions.

Reference Implementation According to the man page `gpg-wks-server(1)`, "the command `--revoke-key` is not yet functional".

The command line parser of the server implementation implements the above command which takes a User-ID and calls the `command_revoke_key` function [20, tools/-gpg-wks-server.c:404ff]. Interestingly the function does not print any informational message that the function is not yet supported, but it calls `wks_cmd_remove_-key` [20, tools/gpg-wks-server.c:1982ff], which removes the key for a given e-mail address from the Web Key Directory. It is from our point of view unintelligible and counter-intuitive why the authors added such behavior. They are aware of this because before calling the removal function they added the comment: "Remove[2] [sic!] should be different from removing but we have not yet defined a suitable way to do this." [20, tools/gpg-wks-server.c:1986ff].

[2]There is a typo in the source code. It is clear from the context that "revoke" is meant here.

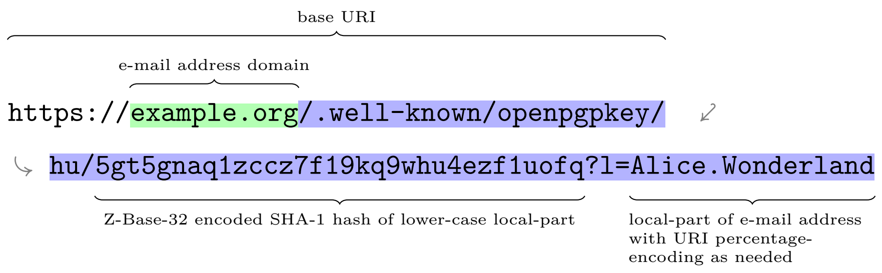
This behavior is odd and unpleasant from a software engineering stance, even though it is not in conflict with the specification.

6.5.3 Confidential Key Requests (G_C)

We defined the security goal “Confidential key requests” as the requirement that nobody except the server and the user who requests a key must be able to determine the User-ID, fingerprint, or requested key. If leaked, these could involuntarily indicate communication partners or other relationships, so they must be kept confidential.

For this security goal we do not have to consider the Web Key Directory Update Protocol because we are only concerned with key requests. The update protocol does not request published keys from the Web Key Directory.

The Web Key Directory Discovery Protocol is based on HTTPS (i.e. HTTP over TLS). As a short reminder, in the discovery protocol a key is discoverable via the e-mail address of a User-ID. A URI is derived from the e-mail address and a HTTP GET request is sent. The server responses with the requested key to this GET request. There exist two kinds of URIs depending on whether the advanced method or the direct method is used. Compared with the URI of the direct method, the URI of the advanced method uses the `openpgpkey` subdomain and the domain part of the requested e-mail address is part of the well-known path. We described the discovery protocol in detail in Section 5.1. We also picture the (direct) URI in Figure 6.12 in this section, again.


base URI
e-mail address domain
Z-Base-32 encoded SHA-1 hash of lower-case local-part
local-part of e-mail address with URI percentage-encoding as needed

```
https://example.org/.well-known/openpgpkey/
hu/5gt5gnaq1zccz7f19kq9whu4ezf1uofq?l=Alice.Wonderland
```

Figure 6.12: Example URI used by the direct connection variant for requesting Alice.Wonderland@example.org’s key. The URI has been wrapped for rendering purposes. The green part is located in the HTTP `Host` header and the blue part in the `GET` header.

As we can see in Figure 6.12, the Web Key Directory Discovery Protocol simply utilizes the HTTPS protocol and the security implications are, with regard to this security goal, only dependent on HTTPS and the steps necessary to establish the HTTPS connection. Both the request from the client, which contains for example the mentioned `Host` and `GET` headers, and the following response with the requested key from the server are only accessible to both parties because the complete HTTP traffic is end-to-end secured by TLS [18, Sec. 4.2.2].

To determine the fulfillment of this security goal, we therefore need to consider TLS as the cryptographic foundation of HTTPS. The exact security implications depend strongly on the used version and configuration of TLS. We want to point out that it is not in the scope of this thesis to find security flaws in cryptographic protocols like TLS. We assume that these protocols can be securely used and hold their security promises.

Besides the information within the HTTP connection there are other locations and situations where information of the request could be leaked that we need to consider. This concerns primarily the used domain which could be used to draw inferences about the requested key because keys are identified by their e-mail address. The resulting security implications (like deniability of a key request) of a domain leakage depend however on additional assumptions. If the key is requested from a domain which only handles e-mail addresses for a single person, it is more likely for the connection to identify that person, than for a domain with thousands of e-mail users.

It is possible that the domain is leaked at the following locations/situations:

- non-confidential DNS resolving
- the destination IP address
- TLS Server Name Indication Extension (SNI) [13]

It is sufficiently well-known that (classical) DNS resolving is not confidential and even protective mechanisms like DNSSEC only ensure integrity. Before a key request can be sent the domain must be resolved. Unless the resolving is performed via a local cache, a passive network attacker could draw connections between the domain resolving and the following TLS protected request to the HTTPS port of the just resolved IP address. There are newer approaches to provide confidential DNS resolving like DNS-over-TLS (DOT) [25] and DNS-over-HTTPS (DOH) [24] but we will not go into these further. Even with confidential DNS requests the TLS connection to the HTTPS port of a specific IP address alone could allow to infer a WKD discovery protocol usage, depending on several factors like whether and what other services are offered via HTTPS on that IP address, the server response size etc.

In addition, there is a TLS extension called “Server Name Indication (SNI)” [13] which is located in the Client Hello message in the TLS handshake and contains the domain. The Client Hello is not encrypted and any passive network attacker could read this information. According to the TLS 1.3 specification, clients should send SNI when applicable [49, Sec. 4.4.2.2]. The TLS 1.2 specification is too old to mention SNI because SNI was not specified then. A brief research indicates that all major web clients (e.g. Internet Explorer, Edge, Firefox, Safari, Chrome) and web servers (e.g. Apache, nginx, Microsoft IIS) support SNI [66].

A mechanism to encrypt the Client Hello (ECH) and therefore protect also the SNI extension is currently in draft status [50]. "ECH is only supported with (D)TLS 1.3 and newer versions of the protocol." [50, Chap. 1].

It is very likely that SNI is used widely and that ECH has not yet become more widely distributed. Therefore, we should assume that the domain part of the e-mail address is leaked by the TLS handshake.

Apart from the e-mail domain the request is confidential. According to the security goal definition we can conclude that the security goal Confidential key requests (G_C) is partially fulfilled.

Reference Implementation We wanted to see whether GnuPG uses Server Name Indication in a WKD key lookup. We performed a key request using `gpg` and examined the TLS Client Hello. The relevant part of the Client Hello is shown in Listing 6.13.

The DNS lookup was performed against the default DNS resolver. The implementation adds the Server Name Indication and does not use the Encrypted Client Hello.

```
TLSv1.3 Record Layer: Handshake Protocol: Client Hello
...
Extension: server_name (len=16)
    Type: server_name (0)
    Length: 16
    Server Name Indication extension
        Server Name list length: 14
        Server Name Type: host_name (0)
        Server Name length: 11
        Server Name: example.org
...
```

Listing 6.13: Extraction of the Server Name Indication Extension of the TLS 1.3 Client Hello.

We can conclude that the domain for the requested e-mail address is leaked by the TLS handshake if `GnuPG` is used.

6.5.4 Privacy Considering Key Exchange (G_P)

The security goal we will now consider demands that personal information, specifically names, e-mail addresses, and images, must only be exchanged if the key owner has approved that.

We do not have to consider the Web Key Directory Discovery Protocol because this is only used to request already published key by the Web Key Directory.

The fulfillment of this security goal is tightly bound to the challenge-response mechanism in the Web Key Directory Update Protocol.

We showed in Section 6.3 (Update Protocol Main Assumption) that the main assumption for the Web Key Directory Update Protocol leaves too much room for interpretation. Additionally, we showed that the used scenario (i.e. the used e-mail infrastructure) is crucial, too. The security implications regarding this security goal are hence dependent on the interpretation and implementation of the main assumption.

Based on the evaluation in Section 6.3, we can make a statement under which interpretation and scenario an attacker could successfully publish a key impersonating as another user. For the security goal it is not important that the key material but the personal information of a user is published. This is also possible with an attacker generated key impersonating another user.

Therefore, we conclude that the security goal is not fulfilled in all cases where an attack vector is available as shown in Table 6.4.

Reference Implementation We presented an attack in Section 6.4.4 (Attack on the Reference Implementation) which allows an attacker to publish keys for any e-mail addresses of any domain a Web Key Directory provider handles. The attack could utilize personal information like names, e-mail addresses, and images of other people to generate keys and publish them. The publication of this information is not approved by the person indicated by the name, e-mail address, or image.

Therefore, the reference implementation does not fulfill the security goal.

6.6 Summary

During the analysis, it became apparent that we had to distinguish between the Web Key Directory specification and the reference implementation in regard to the security implications. Even though we have only knowledge about one Web Key Directory implementation which implements also the Web Key Directory Update Protocol, specifications are designed to make interoperability between several implementations possible, and therefore, we summarize the implications on the security goals separately for the specification and the reference implementation in Table 6.14.

At the beginning of the analysis, we were concerned with the discovery of OpenPGP keys via the Web Key Directory Discovery Protocol. Keys are identified by their e-mail address and the request URI used by the discovery protocol contains the local-part as a SHA-1 hash.

First, we examined in Section 6.1 (SHA-1 Hashed Local-parts) whether it is possible to utilize the – for quite a while deprecated – hash algorithm SHA-1 to target the discovery protocol. We described an idea for an attack where a malicious user Mallory (A_U) wants to find an e-mail address with a local-part which has the same SHA-1 hash as the local-part of the e-mail address of the targeted user Alice. If she succeeds, Mallory could upload her key and potentially overwrite Alice's key. Key requests for Alice's key would then be responded with Mallory's key. This concerns mainly our security goal "Authentic keys" (G_A) but also the goal "Privacy considering key exchange" (G_P), especially if Alice has not already published a key. The attack targets the second-preimage resistance of SHA-1. We found that this property is not broken. Hence, the attack idea is not feasible even for local-parts which are way off the specified local-part length limits.

Second, we examined whether an attacker could succeed if only the much weaker collision resistance is targeted. Therefore, we modified the attack model in such a way that the attacker Mallory assigns users like Alice e-mail addresses. Thereby Mallory can circumvent the strong second-preimage resistance and has only the collision resistance to target because she can freely choose both local-parts instead of only one local-part for the attack. Although there exist practically feasible collision attacks on SHA-1, we could not satisfyingly figure out how the constraints of local-parts impair this feasibility. Due to the constraints (like length and character set), a collision attack could be practically infeasible. At least we assume that the complexity might be considerably higher than a collision attack for SHA-1 hashes over generic data. We did not consider this attack further because the attack model is rather unrealistic.

After the implications of the hashed local-parts, we examined in Section 6.2 (Lowercase Mapped Local-parts) the security implications of the lowercase mapping of the local-parts as it is performed by the Web Key Directory method. We described an attack which targets the same security goal as above (mainly G_A and also G_P) but takes advantage of the lowercase mapping instead of attacks on the security properties of SHA-1. The hash is generated over the lowercase mapped local-part and Web Key Directory makes the assumption that local-parts are treated case-insensitive by almost all providers. An attacker Mallory (A_U) could register an e-mail address similar to the target user Alice's e-mail address which only differs in case. The generated hash would be the same and Mallory could normally submit the key via the Web Key Directory Update Protocol. Consequently, she could successfully publish a key where only the case differs and potentially overwrite Alice's already published key. Key requests for Alice's key would then be responded with Mallory's key.

It is not in contradiction with relevant e-mail specifications that e-mail providers treat their local-parts case-insensitive, but is also not required. To get an impression of the real world situation about how the case of local-parts is handled by e-mail providers, we conducted a small case study. We evaluated 18 providers and all tests (12) which could be successfully conducted indicated that local-parts are handled in a case-insensitive manner by these e-mail providers (see Table 6.2). Thus, we could not identify any e-mail provider which would be susceptible to this attack in our short survey. Without quantifiable data about mail provider market shares, we could not draw more conclusion from it. Based on the survey, it is at least conceivable that the described attack might not have any major practical relevance for generic e-mail providers. However, the lowercase mapping should be considered by administrators at the introduction of Web Key Directory.

Thereafter, we were concerned with the main assumption of the Web Key Directory Update Protocol in Section 6.3. The main assumption is quite unspecific and explicable in different ways. To properly analyze it, we defined scenarios which cover three relevant provider constellations in which Web Key Directory might be used and several interpretations of the main assumption. In addition, we described an attack idea where an attacker Eve (A_U, combined with A_{pN} or A_{aN}) wants to publish a self created OpenPGP key which impersonates her target user Alice. This attack targets the security goals G_A and G_P. We also addressed the implications if Eve has also active network attacker (A_{aN}) abilities. After this, we exhaustively discussed the security implications for all scenario and assumption interpretation pairs.

We showed that there exist several attack points in real world relevant scenarios which are not covered in various discussed assumption interpretations. For one scenario, only one interpretation could prevent the described attack. For the other two scenarios, only half of the interpretations could prevent the described attack. We also gave a detailed summary of our evaluation of the Web Key Directory Update Protocol main assumption in Section 6.3.5.

Besides the considerations in regard to the specification draft, we also addressed the reference implementation of the Web Key Directory key exchange method in Section 6.4. We found out that the implementation behaves in contradiction to the specification draft in several parts, adds non specified behavior, and contains several implementation errors. In addition, we provided an attack to successfully publish OpenPGP keys for any e-mail address for any domain a Web Key Directory provider offers with almost no assumptions.

In Section 6.4.1 (Additional Headers), we described that the reference implementation makes use of two headers in their messages which are not specified in the draft. The `Wks-Phase` header appeared to be not relevant anymore because the reason, to help software to implement the protocol, is not accurate anymore according to the referenced software. More relevant is the `Wks-Draft-Version` header which is added by the implementation to each e-mail. The header is not protected by any means and the reference implementation determines on basis of this header which specification draft version is to be found in the message and which version should be used for e-mail generation. If the above header is not used, the reference implementation assumes the specification draft version prior to version 2 is used which is incompatible with later versions. We pointed out that this is problematic in respect to potential downgrade attacks and to interoperability between different implementations.

In Section 6.4.2 (Irregularities in Message Generation and Processing), we showed that the implementation does not act as specified in several places and include several implementation errors with regard to the e-mail checking and generation. The signature in the confirmation is not checked properly, the confirmation response is not signed and might also be not encrypted if the relevant key is not found. In addition, the reference implementation can not parse (and therefore not accept) properly encrypted and signed confirmation responses due to implementation errors but will accept many others.

In Section 6.4.3 (Irregularities in the WKD Format Checking), we addressed the checking of the Web Key Directory format (the key-value pairs). We found out that they are mostly not checked properly. The `sender` and `address` fields need only to exist and contain valid e-mail addresses for the purpose of RFC 822. It is not checked that the values in the confirmation response correspond with the values of the confirmation request. The `nonce` field is also not properly checked against the value of the confirmation request and is only used as a component in the file path generation to the pending OpenPGP key file. The `nonce` is seen as valid if a key with the User-ID from the `address` field is found at this file path.

We showed that the improper handling of the Web Key Directory format key value pairs, especially the `address` and `nonce`, has severe consequences on the overall security of the Web Key Directory Update Protocol.

We presented in Section 6.4.4 (Attack on the Reference Implementation) an attack on the Web Key Directory Update Protocol which makes it possible for an attacker to publish OpenPGP keys for any e-mail address of any domain which is handled by a Web Key Directory provider. This means a malicious user (A_U + in possession of an e-mail address at the provider) can target security goals G_A and G_P. The attack utilizes the improper handling of the above described key-value pairs. Furthermore, the attack does not rely on strong assumptions because the attacker only needs to be in possession of an e-mail address which is handled by the Web Key Directory provider and the provider must use the reference implementation (with the Web

Key Directory Update Protocol). The attacker has to create an OpenPGP key with two User-IDs (the own and the targeted User-ID). The attacker has to submit this (complete) key. In the confirmation request, the attacker has to change the `address` field from his own e-mail address to the targets' e-mail address. If the target's e-mail address contains a different domain than the own e-mail address, the attacker has to prefix a path traversal string to the `nonce` field, too.

We also gave a detailed summary of Section 6.4 (Reference Implementation) in Section 6.4.5.

Our examination of the Web Key Directory draft specification and reference implementation concerned mainly the security goal "Authentic keys" (G_A) so far. Therefore, we evaluated in Section 6.5 (Considerations on Further Security Goals) if the specification properly specifies the approach for the underlying key exchange method operations (like key publication, key revocation, or key exchange termination) of the security goals and if the reference implementation properly implements these, for example, as part of the Web Key Directory Update Protocol.

With regard to the security goal "Key exchange termination" (G_T) (Section 6.5.1), we found out that the automatic termination of key publications is not possible with the Web Key Directory Update Protocol. However, the termination proceedings are clear and administrators can manually perform these (e.g. by deleting the key from the `hu/` directory of the Web Key Directory). The reference implementation provides a command line argument for this operation.

Key revocations (G_R) (Section 6.5.2) are not possible with the Web Key Directory Update Protocol, as well. Key revocations are more complicated than key removals because manual interventions from administrators are not clear. We examined revocations by revocation certificates and by OpenPGP keys with included revocation signatures and described the open questions which have to be addressed by the specification to make clear interventions possible. For example the revocation certification could be added to the already published user's key or the key could be replaced by the revocation certificate. The reference implementation provides a command line argument for the revocation operation, too. We found out that this operation is not implemented as expected and oddly performs only a key removal.

With regard to the security goal "Confidential key requests" (G_C) (Section 6.5.3), we examined whether User-ID, fingerprint, or requested key of an OpenPGP key request are confidential between server and requester. We described that Web Key Directory utilizes HTTPS for key requests and therefore inherits HTTPS security implications. We found out that above information except the domain part of the e-mail address of the User-ID are confidential. The exception is mainly due to the usage of the TLS Server Name Indication Extension which is part of the unencrypted TLS Client Hello and contains the domain. This extension is also used by the key discoveries performed by `gpg`.

In the end, we addressed the security goal “Privacy considering key exchange” (G_P) (Section 6.5.4) where we described that the fulfillment of this security goal is tightly bound to the challenge-response mechanism in the Web Key Directory Update Protocol. We showed in our discussion of the Web Key Directory Update Protocol main assumption that there is too much room for interpretation and we showed that there are also different applicable scenarios. The security implications for privacy considering key exchanges are therefore dependent on the implications of the main assumption. We concluded that the security goal is not fulfilled in all cases where an attacker could succeed in the attack described in Table 6.4.

We condensed the above examinations regarding the fulfillment of the security goals in Table 6.14 for a better overview.

Table 6.14: Evaluation results regarding the fulfillment of the security goals by the Web Key Directory specification and its reference implementation.

Security Goals	Web Key Directory Protocol	Web Key Directory Reference Implementation
Authentic keys (G_A)	(●/○)[a]	○
Key exchange termination (G_T)	○[b]	○[b]
Key revocation (G_R)	○	○
Confidential key requests (G_C)	◐	◐
Privacy considering key exchange (G_P)	(●/○)[a]	○

●	Security goal is fulfilled.
◐	Security goal is partially fulfilled.
○	Security goal is not fulfilled.

[a]Depending on the main assumption interpretation and scenario.
[b]Key termination is possible if the key owner contacts an administrator and the administrator deletes the key manually.

In Table 6.15, we give an overview about the coverage of Web Key Directory in regard to the OpenPGP key exchange operations as defined in Chapter 3 (Methodology).

Table 6.15: Implementation status of typical operations for OpenPGP key exchange methods as defined in Chapter 3.

Operation	Web Key Directory	
	Protocol	Reference Implementation
Key Submission	yes	yes
Key Lookup	yes (for e-mail address)	yes (for e-mail address)[a]
Key Update	yes (but no revocation)	yes (but no revocation)
Key Refresh	no[b]	no[b]
Key Deletion	no	no (only manual)

[a]Via the `gpg` program in the same software project.

[b]OpenPGP keys can only be clearly identified by their fingerprint. Web Key Directory can only identify keys by their e-mail address. Therefore, it is not possible to clearly check whether a certain key is out-of-date or to request updates for that specific key.

7 Conclusions and Future Work

In this chapter, we will conclude our thesis. We will give an overview about the main results in Section 7.1. Furthermore, we will mention aspects which we could not address in this thesis and could be considered in future work in Section 7.2.

7.1 Conclusions

We described the Web Key Directory key exchange method and analyzed both the method and its reference implementation. To the best of our knowledge there was no previous public research on the Web Key Directory key exchange method. Based on the method description, we found some inconsistencies and specification gaps, some of which we already reported to the specification author and were addressed in version 14 of the Web Key Directory specification draft.

The analysis of the method and reference implementation is summarized in Section 6.6. We will give an overview of our main results below.

Our most important result in regard to the method specification is that the Web Key Directory Update Protocol main assumption is too vague. We showed that there are several interpretations possible, many of which allow an attacker to illegitimately publish keys, thus breaking security goals "authentic keys" (G_A) and "privacy considering key exchange" (G_P). To address this the assumption should be formulated more clearly. It should also be pointed out in which context the update protocol is intended to be securely used.

We found out that the reference implementation is not specification-compliant. The reference implementation is also incompatible to specification-compliant implementations, for example, due to the implicit fallback to older draft version behavior if the unspecified `Wks-Draft-Version` header is missing. In addition, we found further conceptual and implementation errors like missing and incorrect signature generation and verification in the reference implementation. Due to time constraints, we could not report our findings to the program authors yet. For most of these findings we narrowed the problems down to individual code lines which will help the program authors to fix the reference implementation and improve the specification compliance. Furthermore, it would be desirable that the reference implementation is better maintained, since it has many "fixme"'s and almost no updates to the Web Key Directory code since the end of 2016.

Exploiting some of the above errors brought us to our most important result regarding the reference implementation. We found and described an attack on the Web Key Directory Update Protocol implementation which allows an attacker to publish OpenPGP keys for any e-mail address for any domain of a Web Key Directory provider. This attack is based on missing checks of values sent to the server in the confirmation response e-mail, especially the `address` and `nonce` field of the WKD format. The attack relies only on minor assumptions (i.e. the reference implementation is used, and the attacker has an e-mail address by the provider) and is breaking security goals "authentic keys" (G_A) and "privacy considering key exchange" (G_P). Thus, this security issue should be fixed as soon as possible. As above, we could not report this yet due to time constraints.

We described that the Web Key Directory key exchange method supports only a subset of relevant OpenPGP key exchange operations. Especially notable is that key deletion and key revocation is neither specified, nor implemented (except for manual interventions).

In Table 6.14, we condensed the evaluation results with regard to the security goals. We will now conclude the impact on the security goals.

For the Web Key Directory specification, it is dependent on the exact context (which scenario and main assumption interpretation) whether the security goals "authentic keys" (G_A) and "privacy considering key exchange" (G_P) are fulfilled or not because we showed that both security goals are broken in many, but not all of these scenario-main assumption interpretation combinations. Both security goals are broken for the reference implementation because the attack on the Web Key Directory Update Protocol implementation (6.4.4) allows attackers to illegitimately publish keys. It is not evident anymore whether a received key for a key request is legitimate and was published with consent, or not.

The security goals "key exchange termination" (G_T) and "key revocation" (G_R) are not fulfilled because the underlying operations "key deletion" and "key revocation" are not specified. It is possible to delete a published key by manual interventions of administrators. This is only feasible because the manual interventions for the operation key deletion are obvious (unlike key revocation). However, the Web Key Directory key exchange method does not provide specified or automated means for key deletion or revocation..

The security goal "confidential key requests" (G_C) is only partially fulfilled because the domain part of the requested e-mail address is leaked, for example, through the TLS Server Name Indication (SNI) extension. Names, local-parts, fingerprints, and the requested key are secured within the TLS session.

7.2 Future Work

The description and analysis of the Web Key Directory key exchange method took much more effort and time than initially planned. We could not address e-mail validating key servers (by the example of Hagrid), and Autocrypt. Furthermore, it was planned to compare the three key exchange methods Web Key Directory, e-mail validating key server, and Autocrypt in a separate chapter. Due to the much more time-consuming analysis of Web Key Directory, this had to be left out of this thesis.

As part of future work, the remaining OpenPGP key exchange methods e-mail validating key server and Autocrypt could be described and analyzed. These analyses could also be based on our security definitions (see Chapter 4) because they are already designed to be used for OpenPGP key exchange methods and are not only Web Key Directory specific. Based on the security definitions, Web Key Directory, e-mail validating key server, and Autocrypt could then be compared as initially planned for this thesis.

We found problems in the Web Key Directory specification and also in the reference implementation. We will report our findings to the responsible persons. After a suitable time a reevaluation of Web Key Directory would be useful and could point out whether reported findings were properly fixed, or not.

We have the impression that Web Key Directory specification and reference implementation could be better maintained. We found avoidable mistakes and errors which rest in the source code since years. It could be interesting to conduct security analyses on further sub-projects in the GnuPG project. A more dedicated security analysis on the reference implementation of Web Key Directory could also reveal further insights.

Bibliography

[1] Autocrypt team: *Convenient End-to-End Encryption for E-Mail.* `https://autocrypt.org`, visited on July 7, 2022.

[2] A. Barenghi, A. Di Federico, G. Pelosi, and S. Sanfilippo: *Challenging the trustworthiness of pgp: Is the web-of-trust tear-proof?* In G. Pernul, P. Y A Ryan, and E. Weippl (eds.): *Computer Security – ESORICS 2015*, pp. 429–446, Cham, 2015. Springer International Publishing, ISBN 978-3-319-24174-6.

[3] E. Barker and A. Roginsky: *Transitioning the use of cryptographic algorithms and key lengths*, Mar. 2019.

[4] A. von Bidder and N. Weiler: *Key Exchange (KX) - a next generation protocol to synchronise PGP Keyservers.* In *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, pp. 249–254, 2003.

[5] Bundesamt für Sicherheit in der Informationstechnik: *BSI-Projekt "EasyGPG".* `https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Freie-Software/E-Mail-Verschluesselung/EasyGPG/easygpg_node.html`, visited on July 7, 2022.

[6] H. Böck: *SKS: Das Ende der alten PGP-Keyserver.* `https://www.golem.de/news/sks-das-ende-der-alten-pgp-keyserver-2106-157613.html`, visited on July 7, 2022.

[7] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer: *OpenPGP Message Format.* RFC 4880 (Proposed Standard), Nov. 2007. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc4880.txt`, Updated by RFC 5581.

[8] M. Crispin: *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1.* RFC 3501 (Proposed Standard), Mar. 2003. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc3501.txt`, Obsoleted by RFC 9051, updated by RFCs 4466, 4469, 4551, 5032, 5182, 5738, 6186, 6858, 7817, 8314, 8437, 8474, 8996.

[9] D. Crocker: *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES.* RFC 822 (Internet Standard), Aug. 1982. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc822.txt`, Obsoleted by RFC 2822, updated by RFCs 1123, 2156, 1327, 1138, 1148.

[10] Dovecot Authors: *Dovecot manual.* https://doc.dovecot.org/, visited on July 7, 2022.

[11] V. Dukhovni: *Re: Can case-folding for lookup tables be disabled?* https://marc.info/?l=postfix-users&m=163747465219979&w=2, visited on July 7, 2022, thread at the postfix-users mailingslist.

[12] E-Soft Inc.: *Mail (MX) Server Survey. April 1st, 2022.* http://www.securityspace.com/s_survey/data/man.202203/mxsurvey.html, visited on July 7, 2022.

[13] D. Eastlake 3rd: *Transport Layer Security (TLS) Extensions: Extension Definitions.* RFC 6066 (Proposed Standard), Jan. 2011. ISSN 2070-1721. https://www.rfc-editor.org/rfc/rfc6066.txt, Updated by RFCs 8446, 8449.

[14] C. Eckert: *IT-Sicherheit: Konzepte - Verfahren - Protokolle.* De Gruyter Oldenbourg, 2018, ISBN 9783110563900. https://doi.org/10.1515/9783110563900.

[15] M. Elkins, D.D. Torto, R. Levien, and T. Roessler: *MIME Security with OpenPGP.* RFC 3156 (Proposed Standard), Aug. 2001. ISSN 2070-1721. https://www.rfc-editor.org/rfc/rfc3156.txt.

[16] Enigmail Contributors: *enigmail.* https://gitlab.com/enigmail/enigmail, 2022. commit 4f57d6a0e4c030202a07a60bc1bb1ed1544bf679.

[17] Exim Maintainers: *Specification of the Exim Mail Transfer Agent.* Exim project, revision 4.95 ed., Sept. 2021.

[18] R. Fielding (Ed.), M. Nottingham (Ed.), and J. Reschke (Ed.): *HTTP Semantics.* RFC 9110 (Internet Standard), June 2022. ISSN 2070-1721. https://www.rfc-editor.org/rfc/rfc9110.txt.

[19] N. Freed and N. Borenstein: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.* RFC 2045 (Draft Standard), Nov. 1996. ISSN 2070-1721. https://www.rfc-editor.org/rfc/rfc2045.txt, Updated by RFCs 2184, 2231, 5335, 6532.

[20] GnuPG Contributors: *gnupg upstream repository.* https://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git, 2022. commit 3d7d7e8bfd1259e00104b6d52c86d5554fe5b9dd.

[21] GnuPG Project: *WKD.* https://wiki.gnupg.org/WKD, visited on July 7, 2022.

[22] H. Halpin: *SoK: Why Johnny Can't Fix PGP Standardization.* In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery, ISBN 9781450388337. https://doi.org/10.1145/3407023.3407083.

[23] P. Hoffman: *SMTP Service Extension for Secure SMTP over Transport Layer Security.* RFC 3207 (Proposed Standard), Feb. 2002. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc3207.txt`, Updated by RFC 7817.

[24] P. Hoffman and P. McManus: *DNS Queries over HTTPS (DoH).* RFC 8484 (Proposed Standard), Oct. 2018. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc8484.txt`.

[25] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman: *Specification for DNS over Transport Layer Security (TLS).* RFC 7858 (Proposed Standard), May 2016. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc7858.txt`, Updated by RFC 8310.

[26] S. Josefsson: *The Base16, Base32, and Base64 Data Encodings.* RFC 4648 (Proposed Standard), Oct. 2006. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc4648.txt`.

[27] S. Kitterman: *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1.* RFC 7208 (Proposed Standard), Apr. 2014. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc7208.txt`, Updated by RFCs 7372, 8553, 8616.

[28] R. Klafter and E. Swanson: *Evil 32: Check Your GPG Fingerprints.* `https://evil32.com/`, visited on July 7, 2022.

[29] J. Klensin: *Application Techniques for Checking and Transformation of Names.* RFC 3696 (Informational), Feb. 2004. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc3696.txt`.

[30] J. Klensin: *Simple Mail Transfer Protocol.* RFC 5321 (Draft Standard), Oct. 2008. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc5321.txt`, Updated by RFC 7504.

[31] W. Koch: *OpenPGP Web Key Service.* Internet-draft draft-koch-openpgp-webkey-service-03, Internet Engineering Task Force. `https://datatracker.ietf.org/doc/html/draft-koch-openpgp-webkey-service-03`, Work in Progress.

[32] W. Koch: *OpenPGP Web Key Service.* Internet-draft draft-koch-openpgp-webkey-service-00, Internet Engineering Task Force. `https://datatracker.ietf.org/doc/html/draft-koch-openpgp-webkey-service-00`, Work in Progress.

[33] W. Koch: *OpenPGP Web Key Directory.* Internet-draft draft-koch-openpgp-webkey-service-13, Internet Engineering Task Force, Nov. 2021. `https://datatracker.ietf.org/doc/html/draft-koch-openpgp-webkey-service-13`, Work in Progress.

[34] M. Kucherawy (Ed.) and E. Zwicky (Ed.): *Domain-based Message Authentication, Reporting, and Conformance (DMARC).* RFC 7489 (Informational), Mar. 2015. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc7489.txt`, Updated by RFCs 8553, 8616.

[35] B. Laurie, E. Messeri, and R. Stradling: *Certificate Transparency Version 2.0.* RFC 9162 (Experimental), Dec. 2021. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc9162.txt`.

[36] G. Leurent and T. Peyrin: *SHA-1 is a shambles: First Chosen-Prefix collision on SHA-1 and application to the PGP web of trust.* In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1839–1856. USENIX Association, Aug. 2020, ISBN 978-1-939133-17-5. `https://www.usenix.org/conference/usenixsecurity20/presentation/leurent`.

[37] K. Moore and C. Newman: *Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access.* RFC 8314 (Proposed Standard), Jan. 2018. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc8314.txt`, Updated by RFC 8997.

[38] K. Moriarty and S. Farrell: *Deprecating TLS 1.0 and TLS 1.1.* RFC 8996 (Best Current Practice), Mar. 2021. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc8996.txt`.

[39] J. Myers and M. Rose: *Post Office Protocol - Version 3.* RFC 1939 (Internet Standard), May 1996. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc1939.txt`, Updated by RFCs 1957, 2449, 6186, 8314.

[40] Neal: *Hagrid: A New Verifying Key Server Built on Sequoia.* `https://sequoia-pgp.org/blog/2019/06/14/20190614-hagrid/`, visited on July 7, 2022.

[41] C. Newman: *Using TLS with IMAP, POP3 and ACAP.* RFC 2595 (Proposed Standard), June 1999. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc2595.txt`, Updated by RFCs 4616, 7817, 8314.

[42] M. Nottingham: *Well-Known Uniform Resource Identifiers (URIs).* RFC 8615 (Proposed Standard), May 2019. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc8615.txt`.

[43] OpenPGP Alliance: *About keys.openpgp.org.* `https://keys.openpgp.org/about`, visited on July 7, 2022.

[44] T. Polk, L. Chen, S. Turner, and P. Hoffman: *Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms.* RFC 6194 (Informational), Mar. 2011. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc6194.txt`.

[45] J. Postel: *Simple Mail Transfer Protocol.* RFC 821 (Internet Standard), Aug. 1982. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc821.txt`, Obsoleted by RFC 2821.

[46] B. Ramsdell (Ed.): *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification.* RFC 3851 (Proposed Standard), July 2004. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc3851.txt`, Obsoleted by RFC 5751.

[47] K. Renaud, M. Volkamer, and A. Renkema-Padmos: *Why Doesn't Jane Protect Her Privacy?* In E. De Cristofaro and S.J. Murdoch (eds.): *Privacy Enhancing Technologies*, pp. 244–262, Cham, 2014. Springer International Publishing, ISBN 978-3-319-08506-7.

[48] E. Rescorla: *HTTP Over TLS.* RFC 2818 (Informational), May 2000. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc2818.txt`, Obsoleted by RFC 9110, updated by RFCs 5785, 7230.

[49] E. Rescorla: *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446 (Proposed Standard), Aug. 2018. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc8446.txt`.

[50] E. Rescorla, K. Oku, N. Sullivan, and C.A. Wood: *TLS Encrypted Client Hello.* Internet-draft draft-ietf-tls-esni-14, Internet Engineering Task Force, Feb. 2022. `https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-14`, Work in Progress.

[51] P. Resnick (Ed.): *Internet Message Format.* RFC 5322 (Draft Standard), Oct. 2008. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc5322.txt`, Updated by RFC 6854.

[52] A. Reuter, K. Boudaoud, M. Winckler, A. Abdelmaksoud, and W. Lemrazzeq: *Secure Email - A Usability Study.* In M. Bernhard, A. Bracciali, L.J. Camp, S. Matsuo, A. Maurushat, P.B. Rønne, and M. Sala (eds.): *Financial Cryptography and Data Security*, pp. 36–46, Cham, 2020. Springer International Publishing, ISBN 978-3-030-54455-3.

[53] rfcdiff: *rfcdiff of draft-koch-openpgp-webkey-service version 1 and 2.* `https://www.ietf.org/rfcdiff?url1=draft-koch-openpgp-webkey-service-01.txt&url2=draft-koch-openpgp-webkey-service-2.txt`, visited on July 7, 2022.

[54] rfcdiff: *rfcdiff of draft-koch-openpgp-webkey-service version 2 and 13.* `https://www.ietf.org/rfcdiff?url1=draft-koch-openpgp-webkey-service-02.txt&url2=draft-koch-openpgp-webkey-service-13.txt`, visited on July 7, 2022.

[55] J. Schwenk, M. Brinkmann, D. Poddebniak, J. Müller, J. Somorovsky, and S. Schinzel: *Mitigation of Attacks on Email End-to-End Encryption*, p. 1647–1664. Association for Computing Machinery, New York, NY, USA, 2020, ISBN 9781450370899. `https://doi.org/10.1145/3372297.3417878`.

[56] I. Society: *TLS Basics*. `https://www.internetsociety.org/deploy360/tls/basics/`, visited on July 7, 2022.

[57] Statista: *Number of e-mail users worldwide from 2017 to 2025*, 2021. `https://www.statista.com/statistics/255080/number-of-e-mail-users-worldwide/`, visited on July 7, 2022.

[58] J. Tchórzewski and A. Jakóbik: *Theoretical and experimental analysis of cryptographic hash functions*. Journal of Telecommunications and Information Technology, 2019.

[59] The Radicati Group: *Email Statistics Report, 2021-2025*, February 2021. quoted from [57].

[60] Thunderbird Contributors: *comm-central*. `https://hg.mozilla.org/comm-central`, 2022. commit fd96d53b82aca651862860155cc24caa44ca11fd.

[61] A. Ulrich, R. Holz, P. Hauck, and G. Carle: *Investigating the OpenPGP Web of Trust*. In V. Atluri and C. Diaz (eds.): *Computer Security – ESORICS 2011*, pp. 489–507, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg, ISBN 978-3-642-23822-2.

[62] L. Velvindron, K. Moriarty, and A. Ghedini: *Deprecating MD5 and SHA-1 Signature Hashes in TLS 1.2 and DTLS 1.2*. RFC 9155 (Proposed Standard), Dec. 2021. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc9155.txt`.

[63] W. Venema: *local(8) - Postfix local mail delivery*. Exim project, revision 4.95 ed., Sept. 2021.

[64] J. White: *Proposed Mail Protocol*. RFC 524, June 1973. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc524.txt`.

[65] A.M. Wietse Venema: *virtual(8) - Postfix virtual domain mail delivery agent*. Exim project, revision 4.95 ed., Sept. 2021.

[66] Wikipedia contributors: *Server name indication — Wikipedia, the free encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Server_Name_Indication&oldid=1091170563`, 2022. [Online; accessed 15-June-2022].

[67] P. Zimmermann, A. Johnston (Ed.), and J. Callas: *ZRTP: Media Path Key Agreement for Unicast Secure RTP*. RFC 6189 (Informational), Apr. 2011. ISSN 2070-1721. `https://www.rfc-editor.org/rfc/rfc6189.txt`.

List of Figures

Acknowledgements

Figure 1.1 is loosely based on slide 3 of Sebastian Schinzel's talk "Attacking end-to-end email encryption – Efail, other attacks and lessons learned.", accessible via `https://media.ccc.de/v/35c3-9463-attacking_end-to-end_email_encryption`.

Figure 1.1 and Figure 6.3 contain icons from Font Awesome under SIL Open Font License in Version 1.1.

List of Tables

List of Listings

A Inconsistencies and Specification Gaps

In this appendix, we will describe inconsistencies and specification gaps in the Web Key Directory specification. In the first paragraph of this appendix, we address the discovery Protocol and thereafter we will be concerned with the update protocol.

Even though it is only a small mistake, in the assumption that "almost all MTAs treat the local-part case-insensitive" [33, Sec. 3.1] the wording MTA (Mail Transfer Agent) should be replaced by MDA (Mail Delivery Agent) or in general mail server. MTAs are responsible for the e-mail transport between different mail servers and MDAs deliver e-mails to the users' mailboxes. MTAs must treat local-parts case-sensitive and only the host in the domain part of the address must interpret the local-part [30]. Regarding MTAs, Postfix, and Exim have together a market coverage of approximately 93% and preserve the case of local-parts [12, 17, 11]. In case of acting as MDA for an incoming e-mail, both deliver the e-mails in a case-insensitive manner by default. Only Exim offers an option for case-sensitive delivery [17, 63, 65]. For another popular MDA, Dovecot, it is up to the configuration if the delivery is handled in case-sensitive or case-insensitive manner [10]. The potential consequences of this assumption will also be discussed in the analysis below.

In Section 4, third paragraph, the Web Key Directory Update Protocol is described briefly in an enumeration of steps which a user has to perform if they want to submit their OpenPGP key. It is reasonable that such an enumeration does not cover all aspects of the protocol as certain parts are described in detail in sections afterwards, but it must be clear, without contradictions, and must contain all relevant parts. It is not mentioned that the protocol flow can be directly modified by the usage of keywords in the policy file. For example, the "auth-submit" keyword would modify the flow in such a way that the confirmation request and response challenge (steps 4 to 7) are omitted and the OpenPGP key is directly published after its submission. The fact that such modifications are possible through keywords in the policy file is a characterizing property of the update protocol and must be referenced not only in the policy file section. It should be mentioned that future specified keywords may influence the protocol similarly.

Step 6 in the protocol flow enumeration contains a contradiction about the nature of the confirmation response message. It is stated that it is suggested and technically not required to encrypt the confirmation response e-mail. This is in contrast with the definition of the confirmation response in Section 4.4 which defines the response as a signed and encrypted e-mail in the PGP/MIME Combined format.[1]

The specification of the server implementation behavior is not satisfying concerning illegitimate OpenPGP key submission. The confirmation request and response e-mails are only sent in legitimate cases and server implementations should cancel pending key submission after a suitable time. No distinction is made between the nature of illegitimate key submissions. A submission may be illegitimate due to obvious cases like non-responsibility (submission to wrong provider), but also due to violation of policies defined in the policy file. The protocol does not specify that server implementations inform users about violations of policies. If this was a conscious decision, the responsibility of the client implementations should be emphasized to check the policy file prior to key submission. The protocol does not specify the client implementation behavior regarding this file. In the current specification, for example, with the “mailbox-only” keyword set, a legitimate key owner with a key containing a name besides the e-mail address submitting their key without actively checking the policy file would not be notified about the reason why no confirmation request is sent, and the submission is discarded. This behavior seems unnecessarily user-unfriendly.

In Section 4.3, second paragraph, the draft defines the Content-Type of the first MIME multipart part in the Confirmation Request as “text”. According to the MIME Content-Type specification each type must also have a subtype. In this case, the subtype “plain” is missing. Nonetheless, this is only a minor contradiction with the specification as Content-Types are optional and if no type is given “text/plain” is assumed [19].

The specification does not clearly distinguish between the e-mail provider and the Web Key Directory provider. This is relevant in cases where Web Key Directory as a Service is used and both providers are not the same entity. The users’ e-mail provider is then only responsible for e-mail transport/delivery to the Web Key Directory provider and the WKD provider for offering the submission address, submission OpenPGP key, and policy file. As a reminder, Web Key Directory as a Service is covered by the specification as it is a direct implication of the advanced key discovery method (URI with subdomain “openpgpkey”. See Sec. 5.1) which was introduced to be able to flexibly support environments in which several domains can be served by a single Web Key Directory [33]. Therefore, the specification should be more clear on which roles are addressed.

The draft specification contains an appendix with a sample protocol run of the Web Key Directory Update Protocol. This appendix was added to the draft in its second

[1]The contradiction was addressed in draft version 14. It is now stated that the confirmation mail MUST be encrypted to the provider’s public key. Unfortunately it is still not mentioned that the confirmation response must also be signed.

revision and was not modified since then. It must be pointed out that the sample run does not conform with the protocol definition since draft revision 2 (i.e. since the same revision where it was added). Although this non-normative appendix specifies no protocol behavior and is only for implementers' guidance, it would be preferable to update it to prevent confusion and ease the understanding. Prior to the second revision the confirmation request was in the PGP/MIME Combined method whereas thereafter the confirmation request is PGP/MIME signed. Consequently, the Content-Types denoted in the appendix's confirmation request differ from the specification. Based on the nature of a PGP/MIME Combined method message, the Content-Type of the Web Key Directory format (application/vnd.gnupg.wks, -wkd) is within under an encrypted ciphertext and had to be added explicitly. Note that in this case the Content-Transfer-Encoding is also set as "8bit" which is not mentioned in the specification. In addition, the value of the "sender" keyword used in the confirmation response does not conform with the specification as it contains the provider's submission e-mail address instead the e-mail address of the "From" field of this response. Corresponding to the wording of the confirmation response specification, this must be the user's e-mail address because the request is send from the provider to the user and the response from the user to the provider. Furthermore, it contains an "address" key-value pair which is not defined for the use in a confirmation response.[2]

[2]The description of the "sender" keyword was clarified and the "address" keyword was introduced in draft version 14. The usage of the "sender" keyword as given in the appendix was and is correct.

B WKD Update Protocol Run

This appendix contains all messages emerging from a complete and successful OpenPGP key submission using the Web Key Directory Update Protocol. This includes also the used private keys from Alice and the WKS user to ensure reproducibility.

The messages were generated using the latest version available of the server implementation `gpg-wks-server` and client implementation `gpg-wks-client` shipped with `GnuPG` version 2.3.6. The message generation was also conducted with the implementations shipped with the LTS version of `GNUPG` (2.2.35) but did not differ to above messages and were therefore omitted.

Note that these messages are similar to those in the appendix of the draft specification [33], but differ in details.

B.1 Sample Keys

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lFgEYokunhYJKwYBBAHaRw8BAQdAhXuXw4v8AW7IlGoWswHAXLPz8WgtkHvAf18U
6Ae7WfUAAP9czM+rd/qMCIkhxHiQM3HW/Bc3B4zvCGB2SKlI+Lli3RFptBprZXkt
c3VibWlzc2lvbkBleGFtcGxlLm9yZ4iQBBMWCAA4FiEELj/gG15TS9Frp3G44BGQ
dl+MO/4FAmKJLp4CGwMFCwkIBwIGFQoJCAsCBBYCAwECHgECF4AACgkQ4BGQdl+M
O/5OFQD/a2jWq2H17f1Oe7lbFYb5Cvra4wDvSNbf6B/o+OI4uzAA/1TQafJA6Wll
9QR+PfLfWw7LLClXGOyhOWPdmGODGu8OnF0EYokunhIKKwYBBAGXVQEFAQEHQJEB
yUXTeA8fqeGAHrTWLEUFLPwBfp7U5uFLbDKohZgAAwEIBwAA/0cQMOK+WDOCug7j
M3mpil/qkoW1gkTxQOrO8usGD9fIEhKIeAQYFggAIBYhBC4/4BteU0vRa6dxuOAR
kHZfjNP+BQJiiS6eAhsMAAoJEOARkHZfjNP+pPwBAKUq97bYepP8pkYKW8iOmlrS
ARZfLVZgSiXAolJoF6hXAQCFAUyzE8j55dHaPYFNDkt5OC76OtbjAFKvHYo+mesz
BA==
=olUh
-----END PGP PRIVATE KEY BLOCK-----
```

Listing B.1: Private key of the WKS user: `key-submission@example.org`

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lFgEYokuBhYJKwYBBAHaRw8BAQdAvLcXZVOrVheu/CnjlMKbHcZdkeZYBMp7P3RT
bGmS67QAAP98uv7Jgxq9h5RSDohYb+7QDx8kqp7cosMXKe03OQMEZhBbtBlBbGlj
ZSA8YWxpY2VAZXhhbXBsZS5vcmc+iI8EExYIADgWIQQr6TseyRJcH/bCD+cXaC3R
OxrLewUCYokuBgIbAwULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRAXaC3ROxrL
```

```
e2N4APdsmD2IjO5q2Xf3xgQOaDgZANxYvtlD3b4exyGGQDsAAQC4JxXDQJkL1XEJ
2bcHO5HbcqALJkJTeMOh7eS82RpaBZxdBGKJLgYSCisGAQQBl1UBBQEBBOAEctAs
Uv4Ewi/XViIoU63WYjQUttRBNP+dU+2U68dXTgMBCAcAAP9Y9y1NOUikm/b37Oix
3Fckb9oQGKOESdBVUkL2qmhecBCuiHgEGBYIACAWIQQr6TseyRJcH/bCD+cXaC3R
OxrLewUCYokuBgIbDAAKCRAXaC3ROxrLe3RCAPkBc/pcIP+YktIyUerv5tNS+jOU
QaUUff+w8NsbovT5sAEAkDF5DV7wTe+dfDLdPJdwOSEJS56g01QBx7IdmaPVuwQ=
=BOJp
-----END PGP PRIVATE KEY BLOCK-----
```

Listing B.2: Private key of Alice: `alice@example.org`

B.2 Sample Messages

```
From: alice@example.org
To: key-submission@example.org
Subject: Key publishing request
Wks-Draft-Version: 3
MIME-Version: 1.0
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted";
  boundary="=-=01-wan5jhgokpq8wqx3ckjo=-="
Date: Sun, 22 May 2022 13:13:02 +0000


--=-=01-wan5jhgokpq8wqx3ckjo=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-wan5jhgokpq8wqx3ckjo=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DkI3CjsdYJvISAQdAeZmQoYNhy+YSKLTnB6Uug29mw3Zd6pFj2EmTWACL6nUw
9cOhuXgTvmYUHUAWPjV1/sn1MkvcsTGMng2wGZUHYk+hG5CwRCQf8Rz/9+oBpKWl
hF4DVZl1Lb8thr8SAQdAJyIicuzjpwRd5eryl2ylQUliBHCIWdtNexNnSmb6xQww
aVffdwuo3sSPd2Z7qyLvKXIBvn6ORE6gcwQ7NAVM9uCGtw56JdDvcTik3BsOrOuo
OukBDDG1KOr+da9hn/Wbud3bSACRbWHaScfi6S9OI5LbQ/bn+TC1yRCLceuNBMnk
8vgGLrRc1hfE6lzBbm4W5pVjnazce95FPbISYIKm531LxO3fY55sEEmDzzQoNDO9
66/3PZcO1g5IL3lFp6Pb8y1lPXrw88bmgbsOGs6JdgiIZOJ/juFda3hAzcFEAKOo
VP1pMAF6PFi2grO3YYC3oNaI9ZzICGc922JOPqdoKeEYGYF7O3nc+X/xmbOPXaBR
CgBrSRnMIm53N/jOg9V3FnJYcCK3M3+DeUW8LThl2UGnsWkUYXCdaUqrnFHcN9Wl
xefKQ7PKY2ZKI/yNTnUNHQf2t/saEM1e1XgunzbFgtsuEOlNUtjXn9YZ/vXJqICL
ZCGtQOYBEUYzMm/hXKvogX+Omf5kGf4wUZvNiCHV8JFcSm12kQrNq66AcOTe/pw1
8h3lpGQCPVGuTGaVXz2bFcrrKtSNRK574/AHLmuguuOqrOcrvgS6HWC7NSZmswRf
EFgwXokv5aFsSFneGLT71nyOcm4Ac5kr99ymqSQxlbS14fOWfrsNG8N9MTOmgmXJ
9EolkdxMQPXBnbge6PSxOOwIZnvSOsPGFIpheoeUd4HLbCDUN3Mkx8SLg2OGul5y
sDnbQJftvWpDi74blBS59xDj1TOV8/Phi2KVLj8ueanUV8AKdgjIlkdtmSbKRjXF
jebMvqibKrN/AJ4UkE3dcZMOjktCIiFcJ7UHSdrUxvJmcuymVYP9OLY3TEgOKJZ8
t2nOOdKYPDswHWSOg6MwPu/L+mr2KhXoPAjghKWqMts+8PDxE/JR5ODA9cxoR+65
LxmO+d1s5O2mMnxXcPn5uQcsLRfpNphU3aXWzdyQWqHyTHv/9+lSBypCwKfhMZ2E
xKXbz/Clvk2AJKZGOV5xKdJqM9ky6HWSkxk37cClhceBEOMzrzWFbu6iK9sBUA==
=2lUj
-----END PGP MESSAGE-----

--=-=01-wan5jhgokpq8wqx3ckjo=-=--
```

Listing B.3: Submission Mail

```
From: key-submission@example.org
To: alice@example.org
Subject: Confirm your key publication
Wks-Draft-Version: 3
Wks-Phase: confirm
MIME-Version: 1.0
Content-Type: multipart/signed; protocol="application/pgp-signature";
  boundary="=-=01-13m3fowney8irz7kawhy=-="
Date: Sun, 22 May 2022 13:14:00 +0000


--=-=01-13m3fowney8irz7kawhy=-=
Content-Type: multipart/mixed;
  boundary="=-=02-13m3fowney8irz7kawhy=-="


--=-=02-13m3fowney8irz7kawhy=-=
Content-Type: text/plain

This message has been send to confirm your request
to publish your key. If you did not request a key
publication, simply ignore this message.

Most mail software can handle this kind of message
automatically and thus you would not have seen this
message. It seems that your client does not fully
support this service. The web page

       https://gnupg.org/faq/wkd.html

explains how you can process this message anyway in
a few manual steps.

--=-=02-13m3fowney8irz7kawhy=-=
Content-Type: application/vnd.gnupg.wks

-----BEGIN PGP MESSAGE-----

hF4DVZl1Lb8thr8SAQdAqGXorsogA87aqeihSU1Uprmpv61noKU23RmdeNyIRkww
oTQIbNT3isbPwPzZ0HSYUNm7EJICMuwg/Ku9jSPumS1TQIigr/qub9KTHqjbgCip
0sAoAYuLiON239NLp81iBtltsooqfkDa9e6i8sSN1kY9x+XG9w30+fVJmJkfI46S
x+1IdpatMHNQiPG0VjFmZgwZFae9NV6RJFMvqFUuvr9kHxJqfIFGXDCpyLmNwMjN
sfLzPboAC+qUi1k6W6EYi87DS5DQ1x60SWVSHgMeZeM8FsX73S18IVA6d892pqhX
lcUI16emb7jFEkf1BC/G0CIncrt7bBrxBQyn0k6Wozn6TYfE/SjWqL65dPpkq0LB
LdWF26bTx3Rnyd3EBmQr0Ld4gypellQEbyfSQgegKdqAo10ZtLJzy06Cug==
=52fG
-----END PGP MESSAGE-----

--=-=02-13m3fowney8irz7kawhy=-=--

--=-=01-13m3fowney8irz7kawhy=-=
Content-Type: application/pgp-signature

-----BEGIN PGP SIGNATURE-----

iJEEABYIADkWIQQuP+AbXlNL0WuncbjgEZB2X4zT/gUCYoo3GBsca2V5LXN1Ym1p
```

```
c3Npb25AZXhhbXBsZS5vcmcACgkQ4BGQdl+M0/7M9gEA/H1N0etXmdFFbMLdeV89
p108tP2KBgxYAhcbaVd5XawBA0nv0rrwvh7bEcypA7YPeyrcwQivhcZNfRQ9QnJ6
lbkI
=dc90
-----END PGP SIGNATURE-----

--=-=01-13m3fowney8irz7kawhy=-=--
```

Listing B.4: Confirmation Request

```
type: confirmation-request
sender: key-submission@example.org
address: alice@example.org
fingerprint: 2BE93B1EC9125C1FF6C20FE717682DD1D31ACB7B
nonce: 4tkd8z3yxmqkz4gpiybczpf8h8jof37e
```

Listing B.5: Decrypted PGP message in Confirmation Request

```
From: alice@example.org
To: key-submission@example.org
Subject: Key publication confirmation
Wks-Draft-Version: 3
MIME-Version: 1.0
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted";
  boundary="=-=01-sqbtgbja6k8a416ofrso=-="
Date: Sun, 22 May 2022 13:15:14 +0000


--=-=01-sqbtgbja6k8a416ofrso=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-sqbtgbja6k8a416ofrso=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DkI3CjsdYJvISAQdApwcfd8Q8GW2UQXMYloF3n3L+SIioUAXJ0xg501mhlD0w
cUFQfT2rrk+PA9egp6FXlu/uJTnl3yLPL6yreJ4uCsAdmhrtE821ChB31u9VN8ob
hF4DVZl1Lb8thr8SAQdA5UD3gGIU/Ele+uFiV4j1n7kj3mYUpJGLtXWOIFmgKlYw
zCEpV62tdH55gqHUGg6b59QFfHTG5Qct2z1lUOHZ5VhXMdn7xg0tNCjVktYOXwaD
0sA9AXr/6jo1aaJ04ThRZdQTNvABcb1rbIxyQU3tKhNd8/0ZCz+SI8iQ1mlQZ+GS
YvKZVHPFah5nlYbn55pDo9f6d53wkWxTyFQ2mMo17aiNpZBXnovTjXt4v93LH5h4
+hBe32lzgB7wIoGTv4uB6UiDIJhDQ1Ve+4S4Rp5Gthe0TTxiHUF8baCBkrlTdbmk
yeDJ2tIAC+SV2NQ3V0g/Ige0ChO3zr+HVporS+4MLTCR95gPZsNuZFWkUF7uBBdy
Io4DsyoSPpJf4fbt2/WXdk+d3TV8IqUSj4U3tILwDS76Jfsqng2/QWUNizndA4Ts
C1l8YnxEVMp9r5ZF/qfS5g==
=l6c4
-----END PGP MESSAGE-----

--=-=01-sqbtgbja6k8a416ofrso=-=--
```

Listing B.6: Confirmation Response

```
Content-Type: application/vnd.gnupg.wks
```

```
Content-Transfer-Encoding: 8bit

type: confirmation-response
sender: key-submission@example.org
address: alice@example.org
nonce: 4tkd8z3yxmqkz4gpiybczpf8h8jof37e
```

Listing B.7: Decrypted PGP message in Confirmation Response

```
From: key-submission@example.org
To: alice@example.org
Subject: Your key has been published
Wks-Draft-Version: 3
Wks-Phase: done
MIME-Version: 1.0
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted";
  boundary="=-=01-4fr4rcjqmhrtpn3wyoiy=-="
Date: Sun, 22 May 2022 13:15:36 +0000


--=-=01-4fr4rcjqmhrtpn3wyoiy=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-4fr4rcjqmhrtpn3wyoiy=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DVZl1Lb8thr8SAQdAj20ZWqCI4n8NHi+gW0YE3J6wo5WlWtyZwKp7V/6vd2Mw
YKfJR9JQEtEZDi0G8D1320Ue18k/y7CnQl80Q76X4k/u3GmJ5kzS5MIMe14/U1QQ
0sDfAfwsykiiicrnfwVvxLb9nrchyB8ub0j679jl16q8r7XggTHZI34XB0IT+2QH
gpsSzcEY4wZMEeTAtU5EXKDpqIqSTMr5dZ1+LKsWG3CRWxyfZTRPcYCkJC7ytvPq
G8FIlN7D8JiZWr5LElv/5L8hy5aEiyBBlXoDQZuy0SDIkcXRoJhhQK0B5lEa5lY4
lq3c5CPrjLpbjkNk28Q9PqL2FyawyTH41QA5N3fa0hPRj+rPhZ0GLwewgjmLZB87
s0FRhwfz6JWNWMu0KVA3rVKVpczRVq/VyaRL3QNlhZyAgyuy4U8DCx313iUYSsjt
8x0zEhfEu7hmPGEsWydmGUR41MtdTp6FfGH47fjMkA6mAi6hZw36nv9aWMYE0K8q
Zbb5PTmXugFR2n4gv00e5l0dCWthnTkUzdo7R9q8AfYHxQEadY0Nyat+Yp6PP2pD
Z7Ukmv8+2tilTySPXshX8t7hYpNEkfuK370AuiIB2TeRH3So4RB+QFujA7e4e6D5
V82JVYrhKRkMvqGKHovBFTJfvFKKdqUSJBVgJS6Jlz7bVA==
=pp2y
-----END PGP MESSAGE-----

--=-=01-4fr4rcjqmhrtpn3wyoiy=-=--
```

Listing B.8: Publication Message

```
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hello!

The key for your address 'alice@example.org' has been published
and can now be retrieved from the Web Key Directory.

For more information on this system see:

  https://gnupg.org/faq/wkd.html

Best regards

  GnuPG Key Publisher


--
For information on GnuPG see: https://gnupg.org
```

Listing B.9: Decrypted PGP message in the Publication Message

C WKD Attack

This chapter contains all messages and information emerging from the attack on the Web Key Directory Update Protocol described in Section 6.4.4 (Attack on the Reference Implementation).

C.1 Used Keys

The OpenPGP key for the `key-submission@example.org` e-mail address is the same as in Listing B.1.

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lFgEYpkm8BYJKwYBBAHaRw8BAQdACP989Xb+FGGxDHpPH4OVQdeN1tsT5l+xcKKT
GjTjVTEAAQDMSpxAuFYSWZxwdDSVa08JrbBzeTcfqq2IIiDR/N44Cg4ltB1NYWxs
b3J5IDxtYWxsb3J5QGV4YW1wbGUub3JnPoiTBBMWCgA7FiEE/dBXv+Dd/BVJObwU
Cs1/SqVEVN8FAmKZJxICGwMFCwkIBwICIgIGFQoJCAsCBBYCAwECHgcCF4AACgkQ
Cs1/SqVEVN+i3wD7BZhwEVHbUNvvaXGDPqsN2TQ3UdS7UH5YXNcWroavWoOBAJQW
zFYeHbKXTumhLmD35HfLYo8ug1Yf4lhoqGlJrO8OtBlBbGljZSA8YWxpY2VAZXhh
bXBsZS5vcmc+iJYEExYKAD4CGwMFCwkIBwICIgIGFQoJCAsCBBYCAwECHgcCF4AW
IQT9OFe/4N38FUk5vBQKzX9KpURU3wUCYpko6wIZAQAKCRAKzX9KpURU33oSAQD4
t5A0qRx6+djBA9jvXGrcFA4PvoZ45WurQxfhrmi2tAEA4/NvaYkhhMHsduysVTqo
OqyoFW1uoGvrb0aBMuG8wQucXQRimSbwEgorBgEEAZdVAQUBAQdAgpQyhGVWQMHr
PYr7lW8jOi/z7AJx1/pEV8yE/JKmxjADAQgHAAD/S7y1t4rVjQPuG7zciBCw0Dot
y4Uj4iO+SzK/vON01KASBIh4BBgWCgAgFiEE/dBXv+Dd/BVJObwUCs1/SqVEVN8F
AmKZJvACGwwACgkQCs1/SqVEVN8O4QEAshX+Su9QkaeZ6Hj5mIA1GkzkMOVhGMkI
NzWVXfe48cwBAPB8hJs9Wj6qFMn4Lhi9TQkZkS6eqc5Q+2EHy2VdvSME
=PNI2
-----END PGP PRIVATE KEY BLOCK-----
```

Listing C.1: Private key which was created by Mallory (`mallory@example.org`) with an additional (fake) User-ID for Alice (`alice@example.org`)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEYpkm8BYJKwYBBAHaRw8BAQdACP989Xb+FGGxDHpPH4OVQdeN1tsT5l+xcKKT
GjTjVTGOHU1hbGxvcnkgPG1hbGxvcnlAZXhhbXBsZS5vcmc+iJMEExYKADsWIQT9
OFe/4N38FUk5vBQKzX9KpURU3wUCYpknEgIbAwULCQgHAgIiAgYVCgkICwIEFgID
AQIeBwIXgAAKCRAKzX9KpURU36LfAPsFmHARUdtQ2+9pcYM+qw3ZNDdR1LtQflhc
1xauhq9ajQEAlBbMVh4dspdO6aEuYPfkd8tijy6DVh/iWGioaUms7w6OGUFsaWNl
IDxhbGljZUBleGFtcGxlLm9yZz6IlgQTFgoAPgIbAwULCQgHAgIiAgYVCgkICwIE
FgIDAQIeBwIXgBYhBP3QV7/g3fwVSTm8FArNf0qlRFTfBQJimSjrAhkBAAoJEArN
f0qlRFTfehIBAPi3kDSpHHr52MED2O9catwUDg++hnjla6tDF+GuaLa0AQDj829p
iSGEwex27KxVOqg6rKgVbW6ga+tvRoEy4bzBC7g4BGKZJvASCisGAQQBl1UBBQEB
```

```
BOCClDKEZVZAwes9ivuVbyM6L/PsAnHX+kRXzIT8kqbGMAMBCAeIeAQYFgoAIBYh
BP3QV7/g3fwVSTm8FArNf0qlRFTfBQJimSbwAhsMAAoJEArNf0qlRFTfDuEBALIV
/krvUJGnmeh4+ZiANRpM5DNFYRjJCDc1lV33uPHMAQDwfISbPVo+qhTJ+C4YvU0J
GZEunqn0UPthB8tlXb0jBA==
=u4S/
-----END PGP PUBLIC KEY BLOCK-----
```

Listing C.2: Corresponding public key to Listing C.1

C.2 Generated Messages

```
From: mallory@example.org
To: key-submission@example.org
Subject: Key publishing request
Wks-Draft-Version: 3
MIME-Version: 1.0
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted";
  boundary="=-=01-7tioooi47n9iqutjrogy=-="
Date: Thu, 02 Jun 2022 21:12:21 +0000


--=-=01-7tioooi47n9iqutjrogy=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-7tioooi47n9iqutjrogy=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DkI3CjsdYJvISAQdAtnMrI+lV5foFhwcC6Wff1m59pBTCHaA9SRHX+x4/+xAw
KdNePE22gFkK9QId0sKViEYTMXDmZZMgd8HutguypawEAKokPAxWV++aFgqzUO3/
0ukBBDISGgl9LGh1MxIGZ0zfb3ZycjRHx6va+iGgRbRI/Tj00x2jYQuxMAir072L
St0PBprTyjmwilkvApiDJeohmMODIPwWDXYPosnIdS03643lfFjhjxpJOnMHm+Cp
YVkzBE1fHIQC4E02SZ7Zwqtp1GwF29flnoXsoeoVD9PXJD3kNO+SY4FPgIQgC0+Q
F0+TO0Fel79DYbhoUKR9V/LnTXtLUoTyoghRJanUN8OqRgD3RT7PoWWhdh7I2xZt
9ti+693PY+z3F6VVYJMq4WAXx4chLttEht7DVMYJzc0yt0FoS71ehzaMK/iopgmp
Bhe4bf2b7wQL65I/fI4QfCmCVFNuFpwKtdncSkd4YpfgFpm1mt1hIgUc1xnjcR42
+fk8wN7PO2eHwn6AlvbjmDF0uyf03DNgON6dpecJJPDHtMxBa3bEtLTWosfq7bsu
kz1AghYr3qkr8iY0XBrmb4gaIKo+h3ctYiRUjYHwoIPyFJ2jkN4ztjRnAtApYkhW
39Aj7U0C87C8ddF5ZRSW5owcMPN5TR8ip0h9kKlxFM20bfY6S8LpAfzwIkMr/cUW
U1s9q2mlwyH7OH8o/06HzmxqmrvxWIv20S3swAFkZ/LwKEMLn/uaba3oYFU0lpQC
A+Zflc5/V9PoA+mEMVgEcJmMDeNxHJ827EcKQkc0FgkzJ8AafZml0NAGGBw0l2rs
pu4YDlv59BbRM+MI8tnGZaTB0KYmOSo7ahA5xuNRg6+qca3NCwDPDshy5lCu8+Ce
MB0Zc8RzUY7wve9ZStJ+C+8pNhTE8AuBueDoXlLfs4JY1/r6E1e8Uovty+1+1IGX
n0iN8VztYC+ET/9U8ATpf56LEA1i5l3ATbPCz0tseIpg4TqODSfJ/wqNEEl4dJj0
ZNAGH5Gscaygw0pCGBUKRR9lahMySctqTXtddRvdEOFGG5QdAyUBol0Zc7SAKRIc
2hSolznFAtaohth0ptk=
=nclM
-----END PGP MESSAGE-----

--=-=01-7tioooi47n9iqutjrogy=-=--
```

Listing C.3: (Modified) Submission Mail. The public key (with only the User-ID of Mallory) in the PGP message was replaced with the public key with both User-IDs.

```
From: key-submission@example.org
To: alice@example.org
Subject: Confirm your key publication
Wks-Draft-Version: 3
Wks-Phase: confirm
MIME-Version: 1.0
Content-Type: multipart/signed; protocol="application/pgp-signature";
  boundary="=-=01-xqoiptx19dckdwi9deyy=-="
Date: Thu, 09 Jun 2022 19:38:16 +0000


--=-=01-xqoiptx19dckdwi9deyy=-=
Content-Type: multipart/mixed;
  boundary="=-=02-xqoiptx19dckdwi9deyy=-="


--=-=02-xqoiptx19dckdwi9deyy=-=
Content-Type: text/plain

This message has been send to confirm your request
to publish your key. If you did not request a key
publication, simply ignore this message.

Most mail software can handle this kind of message
automatically and thus you would not have seen this
message. It seems that your client does not fully
support this service. The web page

       https://gnupg.org/faq/wkd.html

explains how you can process this message anyway in
a few manual steps.

--=-=02-xqoiptx19dckdwi9deyy=-=
Content-Type: application/vnd.gnupg.wks

-----BEGIN PGP MESSAGE-----

hF4DltbEIrO/3uASAQdAsikRXnx5AWRMZUwZHIftrHdue9X9Ga7w++T4cU3222Iw
bLn6Bf8z4N+37T/OkR2uF1R3yDKmoFXNxNOVur6jgPE6xnAI5lA8G3H8WZdOVSTm
OsAoAVUbslPzUDCQBuda7LB++kEmDg9O3v/fkHQhy3TFjJgR7gi3Zezwtt3YylOB
Fmys9y6C1LfUNJEVgHrQ4Olyhau5wWuw2P6Vt4//W/KjsOsxpBee1DU6XcZcHBgd
OMtgx8jRbR8lOANXvXmyI48J7kGAMlcuOPLaiAB6qc+FL8yvAmKKh7syqwi8KAaC
gJbFjfxrxtyV9DhvvMxKZXYfvnK+vqTU8SpKHm+cLm1AtHvnWgy2uzEsiKGifaO+
6Wsr56i3P9CdbUtyHOiabyI3fMWM7O7+RpTDfYzaSanIkm7ZdRmLUZR4Ug==
=C6K6
-----END PGP MESSAGE-----

--=-=02-xqoiptx19dckdwi9deyy=-=--

--=-=01-xqoiptx19dckdwi9deyy=-=
Content-Type: application/pgp-signature

-----BEGIN PGP SIGNATURE-----
```

```

iJEEABYIADkWIQQuP+AbXlNL0WuncbjgEZB2X4zT/gUCYqJMKBsca2V5LXN1Ym1p
c3Npb25AZXhhbXBsZS5vcmcACgkQ4BGQdl+M0/50+QD8CoP2Ws6283pLhJ1R/6yH
iapIFPIgCzgG1mIN6t2EHGcBAJPTyXWs6gmRJY5U0T5iDz1mZhji2usdtc4vPFf9
/FEB
=hqxp
-----END PGP SIGNATURE-----

--=-=01-xqoiptx19dckdwi9deyy=-=--
```

Listing C.4: Confirmation Request sent to Alice.

```
From: key-submission@example.org
To: mallory@example.org
Subject: Confirm your key publication
Wks-Draft-Version: 3
Wks-Phase: confirm
MIME-Version: 1.0
Content-Type: multipart/signed; protocol="application/pgp-signature";
  boundary="=-=01-cmphm36pdzg41ixqgscy=-="
Date: Thu, 02 Jun 2022 21:19:55 +0000


--=-=01-cmphm36pdzg41ixqgscy=-=
Content-Type: multipart/mixed;
  boundary="=-=02-cmphm36pdzg41ixqgscy=-="


--=-=02-cmphm36pdzg41ixqgscy=-=
Content-Type: text/plain

This message has been send to confirm your request
to publish your key. If you did not request a key
publication, simply ignore this message.

Most mail software can handle this kind of message
automatically and thus you would not have seen this
message. It seems that your client does not fully
support this service. The web page

       https://gnupg.org/faq/wkd.html

explains how you can process this message anyway in
a few manual steps.

--=-=02-cmphm36pdzg41ixqgscy=-=
Content-Type: application/vnd.gnupg.wks

-----BEGIN PGP MESSAGE-----

hF4DltbEIrO/3uASAQdAHhgr7Sxk6BmQ9YcVWHcQi9XqPjCV37SWpGtLWAgP2Www
ekfTHIDkmClYDFKKKjlxcpEefd2wtJrYNblWNFCttdxBl5LFy+nKTtMCrM+lKKqu
OsAqAT19IOW40BYEdMtOGeNxYoy6qFX+6c2+zgB/LJwtEcLh+Aa6CNNbBGPRhl1A
RxW/0/cg+GDG/WxAFEJsOuFZa5FpK81oOl0Zq7iVKpcYfDGrplWUav2cC3ntAIIg
4TBLfvj6PmiDrjerMOmMOPIxgMwmPGj+PG5YUYE5hpjL2V9n+78HjenBioGvR6Nw
i2PIiL4sXlKYwMYknu+11RQH9ION2zsgfiOUHXhTFxqE3j8Lh1KrwaPyTUzfItYg
RXdHU7WCJsDXfby7uYSN4CYzo6fYA5gaFDxbqdsVFtxbU+Sz0poFb6ZBqF/l
=6dtE
-----END PGP MESSAGE-----

```

```
--=-=02-cmphm36pdzg41ixqgscy=-=--

--=-=01-cmphm36pdzg41ixqgscy=-=
Content-Type: application/pgp-signature

-----BEGIN PGP SIGNATURE-----

iJEEABYIADkWIQQuP+AbX1NLOWuncbjgEZB2X4zT/gUCYpkpexsca2V5LXN1Ym1p
c3Npb25AZXhhbXBsZS5vcmcACgkQ4BGQdl+MO/7IXQEAvCZfFlMo8OI5lbb97i7e
rXzngbLDiR5RQU3VrC/ZTj8A/0P1MAYbFtpgoVWsA+xskWnYR7B6OriYrCB9RaB/
wtkF
=Yalb
-----END PGP SIGNATURE-----

--=-=01-cmphm36pdzg41ixqgscy=-=--
```

Listing C.5: Confirmation Request sent to Mallory.

```
type: confirmation-request
sender: key-submission@example.org
address: alice@example.org
fingerprint: FDD057BFE0DDFC154939BC140ACD7F4AA54454DF
nonce: oxbptu91cb4dq3tnss3dm1b9rqbbixj6
```

Listing C.6: Decrypted Confirmation Request (send to Alice).

```
type: confirmation-request
sender: key-submission@example.org
address: mallory@example.org
fingerprint: FDD057BFE0DDFC154939BC140ACD7F4AA54454DF
nonce: 7mpzp4cgkr5uy3bzaw4pqggwga8ksfz9
```

Listing C.7: Decrypted Confirmation Request (send to Mallory).

```
From: mallory@example.org
To: key-submission@example.org
Subject: Key publication confirmation
Wks-Draft-Version: 3
MIME-Version: 1.0
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted";
  boundary="=-=01-7a4ouwa5x1ixx4h3gf3y=-="
Date: Thu, 02 Jun 2022 23:40:10 +0000


--=-=01-7a4ouwa5x1ixx4h3gf3y=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-7a4ouwa5x1ixx4h3gf3y=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DkI3CjsdYJvISAQdAwnH/6pHPzqhNHkyrHi65sOm+jIFZpuR030cVtc5kZnIw
bCcR04Ll8AINvUQGqtwOZ8WxK/zBn19pFF5BvTJjFmTmHkxbp9o/zaXrnGLLbtpS
```

```
0sAhAUrM9ZgEpvLxVwwD680xePkoRcZwcX+h19ENUky0Qkl6wLt4E9GKdGo0FYHI
URwZEgExynoui4i/7HSSVt/tuExz6eknpgIyGYdOoulxr2Tz3V6lWqh87/Qz14yf
UCkWbhgOHS/7AYhumjPxR3L/BhdlWFQr0pGIGtfL+tFG2kRlvINW3cLXHcsLOttq
nEVLStdQLvlLUjRAarpJriSNm2Vx/lwNzc1zB/TIRbv3R6wNI3XAjhLQq+aq7tcW
nt1CHektAoxBSvYM1rqQVo96iMOJ6vitSsJLQA1zV2wlTzn2
=CMUU
-----END PGP MESSAGE-----

--=-=01-7a4ouwa5x1ixx4h3gf3y=-=--
```

Listing C.8: (Modified) Confirmation Response. The address key-value field in the WKS format was changed from mallory@example.org to alice@example.org

```
  Content-Type: application/vnd.gnupg.wks
  Content-Transfer-Encoding: 8bit

  type: confirmation-response
  sender: key-submission@example.org
- address: mallory@example.org
+ address: alice@example.org
  nonce: 7mpzp4cgkr5uy3bzaw4pqggwga8ksfz9
```

Listing C.9: Decrypted Confirmation Response. Shown as a diff.

```
From: key-submission@example.org
To: alice@example.org
Subject: Your key has been published
Wks-Draft-Version: 3
Wks-Phase: done
MIME-Version: 1.0
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted";
  boundary="=-=01-ojz5pgg378htiq7qa1gy=-="
Date: Thu, 02 Jun 2022 23:46:57 +0000


--=-=01-ojz5pgg378htiq7qa1gy=-=
Content-Type: application/pgp-encrypted

Version: 1

--=-=01-ojz5pgg378htiq7qa1gy=-=
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----

hF4DltbEIrO/3uASAQdA1xOZSz8I38SZEw7I6OnNwJRe/RWjpuvAnkjQp+tj+gYw
rn+XXzjOCM1wDIcG8c7PKOlqhQmB+hzE5JLpD8LMiSgwRxzjHOxH+oBn/oK2tYa7
OsDfAR2KnrSL5WNVFrKopmUyVmfcXlWlXlbcKSR6AW1RgZ40tijO134dxlrNvKaO
wq2CjOZdJ38JvTvxJJjFGUIzyYAyR9+PYkEh+xZG9IrIbMVH8MaKrKZi/46s7VWQ
ChJcwncsku0YXLxocbaEOMsKY57DWuSl/Vc7w9WY7pzWNkPY9euCuJPmFNhfAOr/
sMkaCURKb6PtU5nBNkAXBMbFgd/v+pawMQhtkgPOk7OmzA774E22vvuUY/CKQJSq
4nwvYy4POSyvJkakbwAfKLARtX+88M4cyEiZeO/iDO8rB9xy1zq+ZPUwTC6Kwob6
o6CRtlftY1uDm3zxlzl1npVOOO2W/cJJGnGR1vOD+L4YDAabOD3XJ/sYhyyIvaw3
XlPqlX8BJyWSYVj7Q1aRpBt6kQ/FLZhOsEgQxp01B8LpDvFYyIQ+8DvOTKJquxzO
5SgSvz49HSYEhRhhRLrK1ZeKOxDaDYp1TLhXMhHkSRZGuaC62KxD7v+/5yy9Y1yk
2vrTBtAtupXjU1V/5jgQVm5JpaPhSgFZU1EVKQ167FePEw==
=RamN
-----END PGP MESSAGE-----

--=-=01-ojz5pgg378htiq7qa1gy=-=--
```

Listing C.10: Publication Message

```
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hello!

The key for your address 'alice@example.org' has been published
and can now be retrieved from the Web Key Directory.

For more information on this system see:

  https://gnupg.org/faq/wkd.html

Best regards

  GnuPG Key Publisher


--
For information on GnuPG see: https://gnupg.org
```

Listing C.11: Decrypted PGP message in the Publication Message