A proposal for the survival of the OpenPGP decentralized trust network

Gunnar Wolf
gwolf@gwolf.org
Instituto de Investigaciones Económicas, UNAM; Facultad de Ingeniería, UNAM
Ciudad de México, Mexico

Jorge Luis Ortega-Arjona
jloa@ciencias.unam.mx
Facultad de Ciencias, UNAM
Ciudad de México, Mexico

ABSTRACT

While the most common transitive trust model for identity validation in use over Internet is the heavily centralized Public Key Infrastructure with Certification Authorities (PKI-CA) model, it is also possible to make use of a fully decentralized model: the Web of Trust (WoT). The best known implementation of this model is OpenPGP, derived from the original PGP software, first released in 1991. In order to be useful for a geographically-dispersed group of people, the WoT requires a *keyserver network* for key lookup and discovery — in a fittingly decentralized way.

However, during the last decade, several high profile vulnerabilities have surfaced for the keyserver network. These vulnerabilities are not on the software that implements it, but on its basic protocols and assumptions, which make them particularly hard to solve. As a consequence, the keyserver network has shrinked, and it is facing an existential crisis.

This paper outlines a work in progress for solving such a situation, acknowledging the need to keep a decentralized solution for transitive trust model viable, and the proposal to do so by modifying the admission criteria for new key certificates.

CCS CONCEPTS

• **Theory of computation** → **Cryptographic protocols**; • **Social and professional topics** → Identity theft; • **Networks** → *Security protocols*; *Network privacy and anonymity*; Online social networks; • **Information systems** → Collaborative and social computing systems and tools; • **Security and privacy** → **Public key (asymmetric) techniques**.

KEYWORDS

OpenPGP, web of trust, transitive trust models, distributed trust, certificate poisoning

ACM Reference Format:
Gunnar Wolf and Jorge Luis Ortega-Arjona. 2022. A proposal for the survival of the OpenPGP decentralized trust network. In *Conference on Information Technology for Social Good (GoodIT'22), September 7–9, 2022, Limassol, Cyprus.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3524458.3548488

GoodIT'22, September 7–9, 2022, Limassol, Cyprus

ACM ISBN 978-1-4503-9284-6/22/09…$15.00
https://doi.org/10.1145/3524458.3548488

1 INTRODUCTION

The OpenPGP standard is best known for being the most widespread encryption technique for e-mail. Nevertheless, in order to do so reliably, it must be paired with the use of the distributed identity authentication capabilities and transitive trust, the *Web of Trust* (WoT, see Section 2), which links cryptographic key pairs to real-world identities. The WoT requires the information to be made available through a loose network of *keyservers*, avoiding centralization by using a *Gossip* protocol to synchronize their respective keyring databases.

The keyserver network, and thus the whole infrastructure for distributed OpenPGP key distribution, are in existential danger due to a series of weaknesses and attacks (particularly the attack named *certificate poisoning*. Several implementations focus on alternatives for key distribution, but in the process, such implementations either fall back to a centralized scheme, or weaken the WoT transitive trust model.

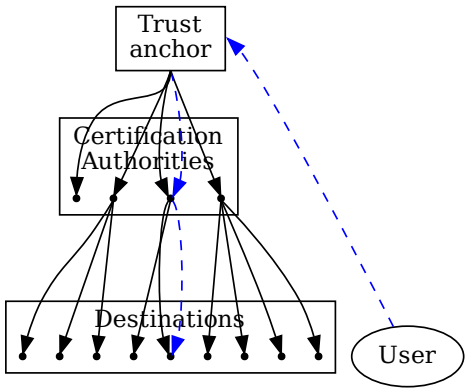
Here, it is presented a protocol for key certifications that tackles certificate poisoning, by requiring all modifications to be *attested* by the target key.

2 TRANSITIVE TRUST MODELS

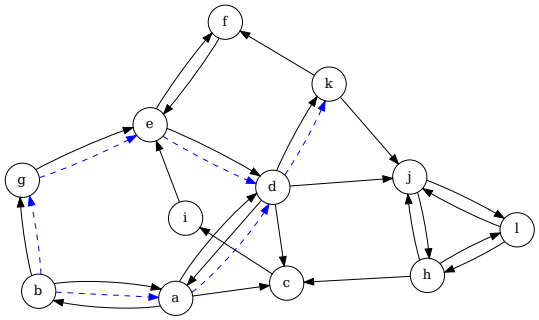
In order to be sure about a given communication peer's *identity*, a *trust model* must be followed. The most common trust models are *transitive*, this is, a host *A directly* trusts the identity of a small set of identities $\{TA_1, TA_2, TA_3, \dots\}$, which are able to *certify* others.

Most of Internet's encrypted traffic is carried out using Transport Layer Security (TLS). The TLS protocol, as well as its predecessor Secure Socket Layer (SSL), is based on the centralized *Public Key Infrastructure – Certification Authority* model (PKI-CA, see Figure 1(a)). In this model, end users fully trust a set of servers known as *trust anchors* [Rescorla 2018, p. 45]. The list of trust anchors is often decided by the operating system or web browser vendor. Trust Anchors usually *delegate* their certifying ability to *Certification Authorities*, so for the example described above, the Certification Authorities (second row) are certified by the Trust Anchors (first row).

There is, however, a different TTM based on a distributed model: the *Web of Trust* (WoT). This model, rather than relying on predefined trust anchors, centers trust roots in the individual node interested in finding a trust path, and conforms the WoT: a mesh-like structure, like the one depicted in Figure 1(b). In this case, if



(a) Centralized model: PKI-CA. User verifies there is a valid path of trust from a prespecified trust anchor to the destination they want to reach.



(b) Distributed model: WoT. User b builds trust paths towards target k along the existing edges of a graph.

Figure 1: TTMs. Black lines denote all trust relationships in the network, and blue dashed lines mean trust paths followed from a user to their target.

user *Bob* (b)[1] wants to communicate with *Karen* (k), he should build a *trust path* so that $b \rightarrow k$ is possible. Acyclically traversing the graph, *Bob* finds the two following paths:

$$b \rightarrow a \rightarrow d \rightarrow k$$

$$b \rightarrow g \rightarrow e \rightarrow d \rightarrow k$$

WoT certifications can also include information other than the identity certification, including expirations and *weights*, allowing for different models for assessing trust levels [Jøsang 1999].

[1]It is common practice in literature about cryptography to refer to the nodes by a familiar, informal name following their initial letters; usually, the first two participants are named *Alice* and *Bob* [Schneier 2015, p. 32].

The best known and widespread implementation of a WoT is OpenPGP [Callas et al. 2007]. The main use of OpenPGP is for *asynchronous* communication: e-mail encryption, document signing (often for file download authentication in the Internet), or document backup. Given that the main use-cases are asynchronous, finding a trust path between b and k needs further infrastructure: no round-trip communication is carried out between them, and b should not be required to send an unencrypted message to bootstrap communication. Hence, a mechanism for *public key distribution* is required. For OpenPGP, specific keyserver software has been developed and later standardized [Horowitz 1997; Shaw 2003]. Independent keyservers synchronize with each other using the *Gossip* large set reconciliation protocol [Minsky and Trachtenberg 2002]. *Gossip* ensures that keys uploaded to any of the keyservers participating in a network quickly reach all other servers.

3 WEAKNESSES AND ATTACKS ON THE WEB OF TRUST

Weaknesses in the WoT have to be presented in different levels: this section addresses not just specific weaknesses in implementation, which lead to identifiable vulnerabilities (that would, in turn, be addressed via specific software fixes, often known as *patches*), but also weaknesses of the base assumptions and protocols upon which the OpenPGP ecosystem is built (much harder to fix, as addressing them is not "simply" patching a software error). The issues are individually described, but a *big picture* mindset should be adopted, as the interaction between the following weaknesses builds up to a much stronger problem than each of them considered separately.

3.1 Lack of model understanding and use

The WoT model assumes OpenPGP users take the necessary steps to get their identities linked to the *strong set*, that is, to the set of public keys that have at least one certificate path to and from the largest bidirectionally interconnected set of keys [Wolf and Gallegos 2017].

Users have historically approached OpenPGP implementations because of the ability to perform strong encryption, seeking the ability to hide the contents of their communications from attackers often identified with communications carriers or national governments. This can be seen in old and new user guides [Hamilton 1998; Petters 2020]; further, this orientation can be found even in corporate guides about the topic [Fortinet 2022]:

> PGP is most commonly used to encrypt email messages. It was initially used by anyone wanting to share sensitive information, such as activists and journalists. But its popularity has increased significantly in the face of organizations and government agencies collecting user data, as people look to keep their personal and sensitive information private.

However, the use of the authentication capabilities conveyed by the WoT are often neglected, or if at all, presented as a feature for "power users". This has led to the vast majority of keys not being at all linked to the WoT: According to Yakubov et al. [2020], at the time of their study, out of 5 217 474 keys carried by the keyservers, over 84% of them exist in isolation, not having any certifications that would allow other users to establish trust in communications

to them; 8.14% of the keys have a single certificate, 2.53% have only two — and only 1.71% of the keys have over 10 certificates. From the keys that are certified by other keys, only 60 000 comprise the *strong set* (the largest bidirectionally interconnected set of keys). This means that, effectively, the WoT is useful only for roughly 1% of the keys uploaded to the keyserver network.

3.2 Immutability of unauthenticated information

The keyserver network is explicitly designed to be resistant to a threat model including national governments' interference or censorship. Data submission to the keyservers is not authenticated (anybody can upload arbitrary OpenPGP data) [Yakamo 2019].

Given that the keyserver network is built on the gossip-based set synchronization protocols, the network is able to accept new keys as well as updates on existing keys, but once information has been recorded and spread, it cannot realistically removed — any server still carrying a packet not found at other servers injects it back into the network; it is effectively decentralized, as there is no central coordination among keyservers [Fiskerstrand 2016].

Having censorship-resistance as a design goal, the protocol provides no provision for removing unwanted information. This issue has carried legal issues for keyserver operators, which are unable to respond to deletion requests based on privacy laws newer than the implementation [Pramberger 2010]. The adoption of the General Data Protection Regulation (GDPR) by the European Union in 2016 makes this issue of prime importance, and has lead several keyserver operators to stop offering their services [Pramberger 2010; Yakamo 2018a].

3.3 Use of the WoT as a *graffiti pad*

The OpenPGP message format standard includes several user-visible fields in an OpenPGP *packet*. One of these fields is the User ID, that "consists of UTF-8 text that is intended to represent the name and email address of the key holder. By convention, it includes an RFC 2822 mail name-addr, but there are no restrictions on its content" [Callas et al. 2007, 48]. Given that there is no enforced format over aid strings, Lee [2014] presented, with the intent of an innocuous and fun *prank* on the protocol, a way to mass-create and upload non-meaningful keys, used then to certify other keys and use the WoT as a *graffiti pad* of sorts, as shown in Figure 2.

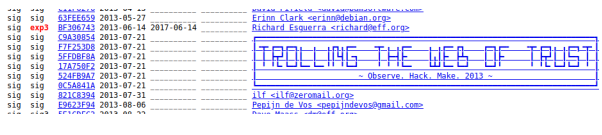


Figure 2: Example of the *Trolling the Web of Trust* attack: When a keyserver displays WoT information for the affected key, each line should include an identity. By creating *throw-away identities* , Lee [2014] uses the WoT as a *graffiti pad*.

Calling this an attack, which could be categorized as a mere *prank*, might seem far fetched, as the impact on the WoT is just adding noise to it. However, this highlights an oversight in OpenPGP's WoT design, that proves to be of a much higher severity.

A derived attack is that of abusing the keyserver network for *arbitrary content distribution*: a program allowing to encode arbitrary information disguised as key and certification material, allowing the abuse of the keyserver network for arbitrary file storage [Yakamo 2018b, 2019]. Given that, as discussed in the previous section, the WoT is effectively a *distributed, anonymous, append-only media, with no content removal facilities*, this abuse makes it effectively illegal to operate keyserver nodes because, due to the GDPR privacy legislation, a government agent can require site operators for given information to be taken offline [Yakamo 2018a].

3.4 Certificate poisoning

The conjunction of some of the above mentioned weaknesses lead directly to a true attack: certificate poisoning.

Let us consider that user *Alice* has key k_A and wants to communicate with user *Bob*, who has key k_B and the certificate chain C_{k_B}. *Alice* connects to the keyserver network and requests for *bob@example.org*. She verifies the results and imports the key into her local keyring. *Alice* checks thoroughly to ensure that k_B is certified by their mutual friends *Charly* (C) and *Diana* (D). OpenPGP keys are also *self-certified*, as the self-signature carries information such as the validity period. She then proceeds to introduce herself to *Bob*. At this point, the certificate chain C_{k_B} is:

$$C_{k_B} = k_B, cert_{k_B \to k_B}, cert_{k_C \to k_B}, cert_{k_D \to k_B}$$

Nevertheless, *Mallory* wants to disrupt the communication between them, so she creates thousands of throwaway keys with no meaningful information; they might not even have a valid identity string, and in no way even lead back to *Mallory*. So *Mallory* controls the keys:

$$k_{M_1}, k_{M_2}, k_{M_3}, \ldots, k_{M_{9\,999}}, k_{M_{10\,000}}$$

Mallory proceeds to certify k_A with all of her throwaway keys and uploads the result to the keyserver network, so that a request for k_A now returns a substantially larger result:

$$C_{k_A} = \begin{cases} k_A, certk_A \to k_A, cert_{k_C \to k_A}, cert_{k_D \to k_A}, \\ cert_{k_{M_1}} \to k_A, cert_{k_{M_2}} \to k_A, cert_{k_{M_3}} \to k_A, \\ \ldots \\ cert_{k_{M_{9\,999}}} \to k_A, cert_{k_{M_{10\,000}}} \to k_A \end{cases}$$

OpenPGP public keys are typically a few kilobytes long; very well connected keys (that is, keys certificated by many other users) can reach the few hundred kilobytes. But after *Mallory*'s attack, k_A measures tens or hundreds of megabytes, and has become *poisoned* — unusable. When *Bob* attempts to get k_A, his OpenPGP client faces orders of magnitude more information to what it is designed to handle. Observed failures from this attack include program freezes and the corruption of *Bob*'s local keystore.

At this point, *Alice*'s key cannot be used anymore, and she needs to migrate to a new k'_A. In order to do so, she also has to meet face to face to cross-verify identities to rebuild trust and become linked again to the WoT. Besides that, once k'_A gained enough signatures to be useful, *Mallory* can repeat her attack.

While few actual such attacks have been reported [Kahn Gillmor 2019b], they are a looming threat to the OpenPGP community as a

whole, and they are part of the reasons the keyserver network is perceived as dying [Lee 2019].

4 PROTOCOL FOR KEY CERTIFICATIONS

Key certification requires interested users to perform an *off-band* identity validation, in which they exchange their *key fingerprints* in order to be able to vouch for each other's identity. This process is depicted in Figure 3: *Bob* is certifying *Alice*'s public key, and although they *should* have met, the keyserver does not enforce this — that means, the keyserver does not require *Alice* to be even aware of any certifications being added to her key. This is precisely the problem that allows certificate poisoning!

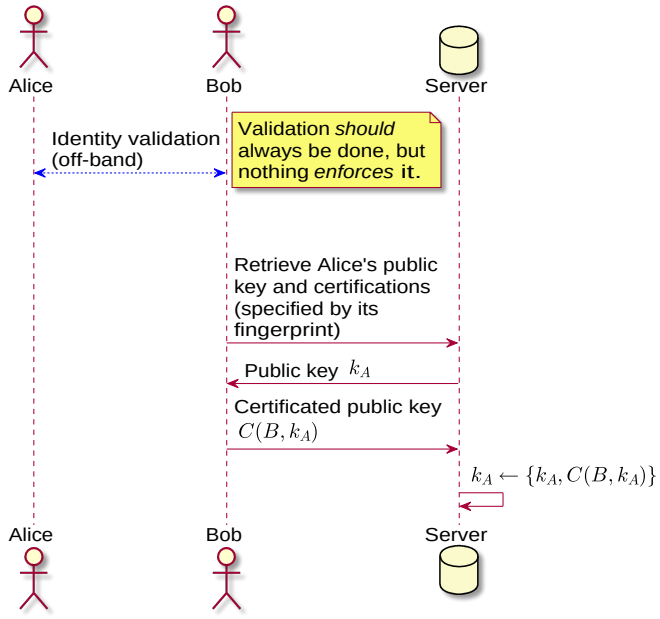


Figure 3: Sequence diagram presenting how *Bob* certifies *Alice*'s key under the preexisting keyserver network: no action is required from *Alice* for new certificates to be appended to her key.

Our proposal modifies the keyserver logic, so that any modification on *Alice*'s key k_A must be approved by *Alice*, as Figure 4 shows. Any packet that modifies k_A has to be *attested* (signed or certified) by Alice's key.

Implementations of proposals similar to this one have been described as *First-party-attested third-party certifications* or *1PA3PC* [Kahn Gillmor 2019a], although no implementations have yet been proposed. It should be easy to understand why 1PA3PC makes certificate poisoning impossible: going back to the example presented in Subsection 3.4, even if *Alice* is not careful and attests some spurious certificates on her key created by *Mallory*'s throwaway accounts, now she does not accept tens of thousands of attestation requests. The keyservers do not distribute most of $cert_{k_{M_1}}$ through $cert_{k_{M_{1\,0\,000}}}$, and *Alice*'s key remains sane.

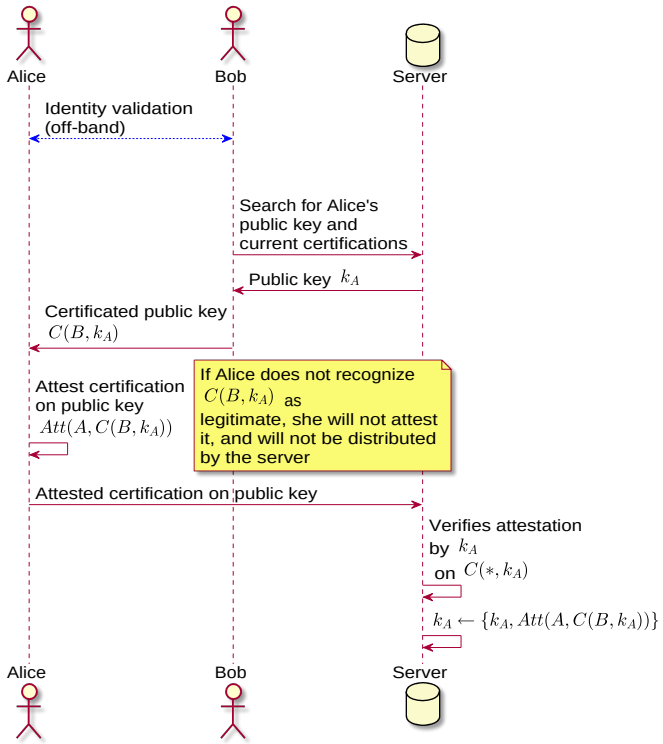


Figure 4: Sequence diagram for the proposed first-party-attested key certification

We must note that 1PA3PC does not prevent the *legitimate* use the WoT as a *graffiti pad*,[2] as described in Subsection 3.3: if *Alice* wants to "decorate" her WoT listing with meaningless throwaway identities, she is able to attest and upload them.

4.1 Threats to applicability

OpenPGP keyservers accept all packets to be appended to existing keys in an unauthenticated way, no matter their provenance. This is partly explained by the issues highlighted in Subsection 3.1: properly using the WoT *takes effort* from casual users; the current keyservers has the lowest possible barrier for participation — and still, participation in the WoT barely reaches 16% of all keys, with barely over 1% belonging to the strong set. However, we do recognize that adding requirements and steps to be taken could likely lead to a lower adoption rate of the TTM.

A keyserver based on the proposed protocol would be unable to participate in the preexisting keyserver network, as the delta of its database and that of other servers would quickly drift apart, as they could present different admissibility politics for key material. An initial keyserver build for a first import can be developed by filtering out packets including non-bidirectional key signatures, but further *Gossip* synchronization runs would probably fail. Thus, a

[2] It could be argued that *graffiti pads* are necessarily abusive, as they use the keysigning network in a way it has not been devised to operate, loading the network with meaningless information. We take the use as *legitimate* because its impact on the network is minimal and affects only the key signed by said throwaway identities.

parallel keyserver network can be started, *seeded* by material fund in the keyserver network, but rejecting non-complying material.

5 RELATED WORK

The OpenPGP user community has not remained idle to the above mentioned challenges. There are several strategies that can be nowadays adopted against them, but most achieve so by compromising on the distributed property of the WoT model. This Section reviews the main proposed strategies.

5.1 Different key discovery mechanisms

The following strategies identify the synchronizing keyserver network as the weak link that enables the attacks, and present different ways for users to discover and assign trust to user's keys.

DNS-based Authentication of Named Entities (DANE) Given that keys are used in connection with their user's mail address, DANE encodes the public key in the answer of a special DNS request on the mail server's domain [Wouters 2016]. DANE requires the mail provider to be willing to present this centralized facility, not only becoming a single point of failure, but also requiring the domain owner to take interest in providing it; nowadays, with the amount of mail users in large-scale mail providers such as Google's GMail, a large amount of e-mail users would be effectively shut off this mechanism.

Web Key Directory (WKD) Koch [2021] presents keys under a specified directory of the mail provider's domain, similarly to DANE. WKD faces less *friction* than DANE, as Web pages are easier to administer than DNS zones and can more easily be delegated to less technical users, but it still requires a highly centralized setup, and leaves the users completely at the whim of the service provider in order to distribute their identities.

Trust On First Use (TOFU) Its proponents reason from the low adoption of the WoT presented in Subsection 3.1, and portray the WoT as *theoretically* strong, but *practically* not useful [Walfield and Koch 2016]. To distribute the public keys, it is included as part of the e-mail headers of every mail the user sends (stripped of any certificates, in order not to bloat total message size), and users willing to engage in a private communication would reply to said mails; TOFU is also known as a *Leap of Faith* trust scheme, as the first communication are vulnerable to impersonation. It is highly unlikely, they reason, that the first communication between two parties would be as important as subsequent ones, and for future communications, trust is to be built from the repeated usage of encryption.

5.2 Different server synchronization mechanisms

Recognizing part of the issue is *Gossip*'s impossibility to remove or even stop serving given keys or certificates, the following works present different synchronization schemes:

BlockPGP Presents an Ethereum-derived blockchain for representing changes in the keyserver data [Yakubov et al. 2020]. This solution can address propagation times of keys to the whole network and better bringing up to date servers that have fallen behind on the synchronization. While the keyserver operation would thus retain its decentralized way, it does introduce an *administrator account* that could remove toxic information from the blockchain; this is often seen as antithetical to the decentralized philosophy of OpenPGP.

Hagrid Sequoia-PGP is a reimplementation of several aspects of OpenPGP. One of the modules it presents is the Hagrid verifying keyserver [Breitmoser et al. 2022]. This keyserver is purposefully not interoperable with the keyserver network, and by discarding all signatures from the keys it serves, makes it impossible for the above mentioned attacks to happen: it allows users to query for keys, but does away with all needed information for establishing a WoT. As the project's website mentions, Hagrid does not publish identity information without the consent of the user, and allows the removal of identity information. It is not currently federable, although federability is on their future roadmap.

5.3 Leaving OpenPGP behind

OpenPGP can be characterized as belonging to a different era, and new protocols have been proposed to replace it, with different starting assumptions and threat models.

Off The Record (OTR) First presented by Borisov et al. [2004], OTR aims at low-latency message-oriented interactions, such as instant messenger. It implements *perfect forward secrecy*, very short-lived session keys and explicit repudiability. The trust model is TOFU, but with the possibility to authenticate a peer's key off-band.

6 CONCLUSION

This work shows a minor modification to the key certification protocol, requiring the owner of a given keypair to acknowledge any modification attempt done to their public key material, It is able to make a crippling attack such as *certificate poisoning* impossible, and can allow the distributed Web-of-Trust transitive trust model to continue to work.

6.1 Further work

This work describes a *work in progress*. While this article sketches out the main idea, its viability is yet to be proven. This requires adapting current keyserver implementations to enforce the presented protocol.

Along with the poisoned certificates issue, being able to remove key material in order to be able to comply with GDPR requests (or requests of comparable laws in different jurisdictions), has to be tackled as well; while it is outside the scope of this work, the preparation of a new keyserver network infrastructure is a great opportunity to work on the GDPR angle.

REFERENCES

Nikita Borisov, Ian Goldberg, and Eric Brewer. 2004. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. Association for Computing Machinery, New York, NY, USA, 77–84. https://doi.org/10.1145/1029179.1029200

Vincent Breitmoser, Justus Winter, Kai Michaelis, and Nora Widdecke. 2022. Hagrid: a verifying keyserver. https://gitlab.com/hagrid-keyserver/hagrid/

Jon Callas, Lutz Donnerhacke, Hal Finney, David Shaw, and Rodney Thayer. 2007. OpenPGP Message Format. *Internet Engineering Task Force (IETF)* 4880 (2007), 89 pages. https://www.rfc-editor.org/info/rfc4880

Kristian Fiskerstrand. 2016. OpenPGP Certificates can not be deleted from key-servers. https://blog.sumptuouscapital.com/2016/03/openpgp-certificates-cannot-be-deleted-from-keyservers/

Fortinet. 2022. PGP Encryption. https://www.fortinet.com/resources/cyberglossary/pgp-encryption

David Hamilton. 1998. PGP for ABSOLUTE Beginners. http://axion.physics.ubc.ca/pgp-begin.html

Mark Horowitz. 1997. *PGP public key server.* Master's thesis. MIT. https://www.mit.edu/afs/net.mit.edu/project/pks/thesis/paper/thesis.html

Audun Jøsang. 1999. An Algebra for Assessing Trust in Certification Chains. In *Proc. Network and Distributed Systems Security Symposium, 1999 (NDSS'99)*. The Internet Society, 10 pages. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.4065

Daniel Kahn Gillmor. 2019a. *Abuse-Resistant OpenPGP Keystores.* Internet-Draft draft-dkg-openpgp-abuse-resistant-keystore-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-abuse-resistant-keystore-04 (Expired draft).

Daniel Kahn Gillmor. 2019b. OpenPGP Certificate Flooding. https://dkg.fifthhorseman.net/blog/openpgp-certificate-flooding.html

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

Micah F. Lee. 2014. Trolling the Web of Trust. https://github.com/micahflee/trollwot

Micah F. Lee. 2019. The Death of SKS PGP Keyservers, and How First Look Media is Handling It. https://code.firstlook.media/the-death-of-sks-pgp-keyservers-and-how-first-look-media-is-handling-it

Yaron Minsky and Ari Trachtenberg. 2002. Practical Set Reconciliation. In *40th Annual Allerton Conference on Communication, Control and Computing*, Vol. 248. Allerton House, Monticello, IL, 16 pages. https://git.gnunet.org/bibliography.git/plain/docs/practical.pdf

Jeff Petters. 2020. What is PGP Encryption and How Does It Work? https://www.varonis.com/blog/pgp-encryption

Peter Pramberger. 2010. Keyserver.pramberger.at terminating. https://lists.nongnu.org/archive/html/sks-devel/2010-09/msg00009.html

Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol, Version 1.3. *Internet Engineering Task Force (IETF)* 8446 (2018), 159 pages. https://www.rfc-editor.org/info/rfc8446

Bruce Schneier. 2015. *Applied Cryptography: Protocols, Algorithms and Source Code in C.* John Wiley & Sons, Hoboken, NJ.

David Shaw. 2003. *The OpenPGP HTTP Keyserver Protocol (HKP).* Internet-Draft draft-shaw-openpgp-hkp-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-shaw-openpgp-hkp-00 Work in Progress.

Neal H. Walfield and Werner Koch. 2016. TOFU for OpenPGP. In *EuroSec'16: Proceedings of the 9th European Workshop on System Security*. Association for Computing Machinery, New York, NY, 1–6. https://doi.org/10.1145/2905760.2905761

Gunnar Wolf and Gina Gallegos. 2017. Strengthening a curated web of trust in a geographically distributed project. *Cryptologia* 41, 5 (2017), 459–475.

Paul Wouters. 2016. DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP. *Internet Engineering Task Force (IETF)* 7929 (2016), 19 pages. https://www.rfc-editor.org/info/rfc7929

K Yakamo. 2018a. Are PGP key-servers Breaking the Law under the GDPR? https://medium.com/@mdrahony/are-pgp-key-servers-breaking-the-law-under-the-gdpr-a81ddd709d3e

K Yakamo. 2018b. Are SKS keyservers safe? Do we need them? https://medium.com/@mdrahony/are-sks-keyservers-safe-do-we-need-them-7056b495101c

K Yakamo. 2019. Using PGP keyservers for decentralised file storage. https://github.com/yakamok/keyserver-fs

Alexander Yakubov, Wazen Shbair, Nida Khan, Radu State, Christophe Medinger, and Jean Hilger. 2020. BlockPGP: A Blockchain-based Framework for PGP Key Servers. *International Journal of Networking and Computing* 10, 1 (2020), 1–24. https://doi.org/10.15803/ijnc.10.1_1