Deconstructing the Decentralization Trilemma

Harry Halpin[a]
Inria de Paris, 2 Rue Simone Iff, Paris, France

Keywords: Software Architecture, Security, Privacy, Scalability, Federation, Decentralization, Blockchain.

Abstract: The vast majority of applications at this moment rely on centralized servers to relay messages between clients, where these servers are considered trusted third-parties. With the rise of blockchain technologies over the last few years, there has been a move away from both centralized servers and traditional federated models to more decentralized peer-to-peer alternatives. However, there appears to be a trilemma between security, scalability, and decentralization in blockchain-based systems. Deconstructing this trilemma using well-known threat models, we define a typology of centralized, federated, and decentralized architectures. Each of the different architectures has this trilemma play out differently. Facing a possible decentralized future, we outline seven hard problems facing decentralization and theorize that the differences between centralized, federated, and decentralized architectures depend on differing social interpretations of trust.

1 INTRODUCTION

Although there has been a move towards decentralization, projects with decentralized architectures have had fundamental difficulties: Bitcoin and Ethereum seem to be unable to scale to as large a number of transactions as centralized systems such as Visa. On the other hand, centralized projects are increasingly the subject of data leakage and other attacks, calling their security into question. This has been phrased as the "decentralization trilemma" by the co-founder of Ethereum, Vitalik Buterin, and is a widely spread truism in blockchain development that has not been rigorously analyzed and critiqued.[1]

Intuitively, there does seem to be fundamental trade-offs between decentralization, scalability, and security as illustrated in Figure 1: Systems can be less decentralized and more scalable versus more decentralized and less scalable. Indeed, decentralization does seem like a trade-off against scalability, but most large real-world deployments of scalable software, such as Amazon, are actually distributed systems with large trust assumptions and centralized coordination (DeCandia et al., 2007). Also, there are secure centralized systems that use advanced encryption (such as the Signal instant messenger), and decentralized systems that have been found to be insecure, as various attacks on the distributed hash tables used by peer-to-peer file-sharing networks show (Wolchok et al., 2010). Decentralization can be thought of as arising from a separate threat model than traditional security assumptions: A lack of trust in centralized servers. In this paper, we outline the threat model, the *malicious server*, that these decentralized architectures are trying to address in Section 2. Decentralization, which we define in terms of an adversarial approach to distributed systems (Troncoso et al., 2017), is then explored in Section 3 as separate from classical security and scalability requirements. Rather than a binary division, we view decentralization on a spectrum that can be broadly construed as centralized, federated, and decentralized architectures in Section 4, and we provide an analysis in Section 5. In Section 6 we outline six open problems that decentralized systems face in meeting the requirements currently met by centralized architectures. In our conclusion in Section 7, we reassess decentralization and turn to the social hypothesis at the heart of decentralization.

2 THE MALICIOUS SERVER

In distributed systems, all entities are considered to be capable of sending messages (Lamport et al., 1982). In centralized systems, users do not directly receive messages but are mediated by a client (a device, a program such as a mail-reader or a browser, etc.) where the client communicates to a server that stores and for-

[a] https://orcid.org/0000-0003-2143-6965

[1] https://github.com/ethereum/wiki/wiki/Sharding-FAQ

Halpin, H.
Deconstructing the Decentralization Trilemma.
DOI: 10.5220/0009892405050512
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 505-512
ISBN: 978-989-758-446-6

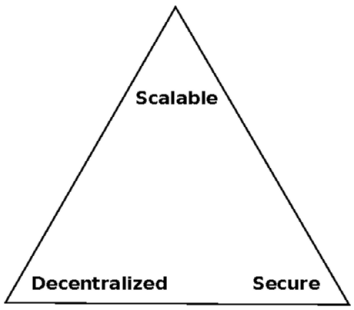

Figure 1: The decentralization trilemma.

wards messages to the client. As exemplified by cloud computing, this server is assumed to be always online. One of the primary advantages of centralized servers seems to be that the deployment and upgrading of any protocol is easier via the usage of a centralized server.

As the server mediates all messages, the server is a *trusted third party*. Centralized systems can be secure and maintain privacy against even powerful adversaries. Secure messaging applications ranging from Signal to WhatsApp, depend on centralized servers (Unger et al., 2015). In the case of secure messaging protocols, even if the message content is encrypted, the server is usually necessary for delivery, especially if the client is offline. As shown by the simple case that messages are assumed not be dropped by the server, the security and privacy properties of centralized servers are dependent on a single root of trust.

Under the adversary model of *malicious security*, trusted components of a larger distributed process may no longer follow the protocol and so no longer maintain their security properties (Yung, 2015). To achieve malicious security, the protocol must work even with a component no longer maintains its security properties. Although originally aimed at cryptographic primitives, malicious security also applies at the level of the entire architecture of a system. In particular, a malicious server is one that no longer securely relays messages according to the protocol.

We theorize that the goal of decentralization is to achieve protocols with malicious security in the face of any component, and so also achieve resistance in terms of the *byzantine failure* of components, where components behavior is unrestricted, and so may be malicious or "unintentionally" faulty (such as being offline) (Lamport et al., 1982). Given the popular deployment of servers in computing protocols, amongst all possible components, the primary threat model of decentralization is a *malicious server*, where the server is untrusted. Malicious security can be achieved in a decentralized protocol simply by not relying on a server at all. Malicious servers are a real-world adversary: As the revelations by Edward Snowden showed that it was trivial to compromise the security of e-mail if they were not end-to-end encrypted by simply retrieving the cleartext from the centralized server. Even if the message content is encrypted, "metadata" such as the time and recipients of the message, often has no legal protection from surveillance (Slobogin, 2014).

The goal of a malicious server in the centralized client-server setting is to read messages and possibly forge the message content of one or more clients. Even if the message is encrypted, the malicious server may be "honest but curious" and so also has the goal of determining the identity of the senders and recipients of messages, i.e. the discovery of the "social graph" of the communication. The server may use techniques like dropping messages, replaying messages, or sending fake messages to identify users. The server is *local*, it does not have the ability to observe the entire network, but can observe every message it relays, and so differs from threat models like the global passive adversary that can observe all messages in an entire network or an adversary that can arbitrarily corrupt any component in the system. The malicious server may be a mobile adversary and so the corruption of the server may be limited in duration (Yung, 2015).

Before cloud computing, servers were generally viewed as trustworthy and message-passing between them governed primarily by standardized protocols such as SMTP and HTTP, leading to federated systems. Although the servers were generally regarded as trusted rather than malicious in early federated internet architectures, issues such as spam and a need for payments led to increased centralization. The pendulum is now swinging in the other direction: It is our fundamental hypothesis that decentralized systems arose in response to the threat model of the malicious server. For example, Bittorrent came into usage after the the centralized index of Napster was eliminated, and Bitcoin became popular after the centralized servers in classical e-cash schemes were viewed as a liability after the "take down" of e-Gold (Troncoso et al., 2017).

3 PROPERTIES: SECURITY AND SCALABILITY

Security and scalability are very broad notions. It is useful to decompose them into more distinct and better known technical properties that can be enforced via cryptography and engineering. The list of properties below is somewhat arbitrary, but reflects classical distinctions within the fields of computer security and

distributed systems.

3.1 Security Properties

A number of traditional of security properties are needed to counter a malicious server. Security properties should be able to hold in terms of messaging even in the presence of a malicious server, which is considered the "adversary" below:

- *Confidentiality*: The adversary cannot read the cleartext of any message.
- *Integrity*: The adversary cannot alter the message without detection.
- *Authenticity*: Only the intended recipient can receive and send the message (in order to prevent replay and impersonation attacks by the server).

Although informally thought of as "security" properties in the decentralization trilemma, we can consider privacy properties separately:

- *Unlinkability*: Each message of a client is unlinkable from any given other message by the client or any other client (Pfitzmann and Hansen, 2005).
- *Unobservability*: A message cannot be distinguished from not sending a message (Pfitzmann and Hansen, 2005).

3.2 Scalability Properties

Additional properties relate to the scalability of the system, and so can be contrasted with security and privacy requirements (Das et al., 2018). A system should be able to scale so it is both available to clients when needed, can support new clients, and deliver messages within a period of time acceptable to the user.

- *Availability*: Messages can be sent or received by the client when requested.
- *Capacity*: New clients may be added at any time while maintaining availability.
- *Latency*: Messages may be sent by one client and received by another client within an acceptable period of time.

Some requirements cannot be fulfilled by traditional cryptographic and engineering requirements, as these related to the freedom of users, particularly when they are under attack by a malicious server.

- *Transparency*: Messages or information about message-passing behavior are recorded for clients to access.

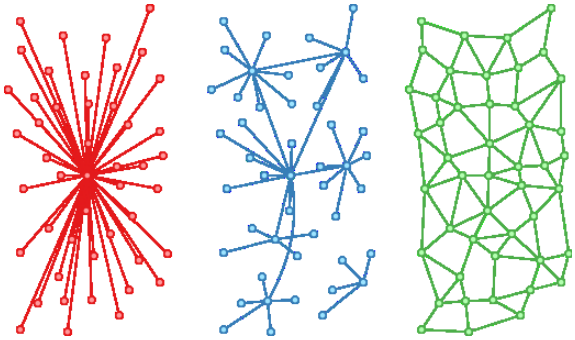

Figure 2: From left to right: (1) Centralized (2) Federated (3) Decentralized.

- *Portability*: Messages may be send and received across all servers and clients, so a user may move their messages to another server or send the message directly themselves in case their server becomes malicious.

These properties are by no means exhaustive, but are meant to deconstruct and ground the terms "security" and "scalability" in the decentralization trilemma in well-accepted technical properties. There are more informal properties that can fit within this framework. Take *usability*, which can be defined as the ability to easily retrieve and send messages by the user.

4 DECENTRALIZED ARCHITECTURES

In order to minimize the damage a malicious server could cause, decentralization is put forward as an alternative architecture to trusted third parties. However, there are multiple variations of architectures claiming to be decentralized: For example, efforts like Autocrypt claim that e-mail, despite involving a SMTP server, is decentralized.[2] So defining and comparing these architectures in terms of messaging passing between clients and servers is necessary. We divide the three primary architectural choices as centralized, federated, and decentralized. These architecture were classically illustrated by Baran in the following Figure 2 (Baran et al., 1964).[3]

[2]https://autocrypt.org/

[3]In his original work, Baran called these (1) Centralized, (2) Decentralized, and (3) Distributed, but terminology has been changed to be consistent with current software architecture practice where "federated" has a clear meaning in terms of a multiple authoritative servers with one or more clients each, while the difference between "decentralized" and "distributed" in informal modern parlance has become vague.

4.1 Centralized

An architecture is strictly *centralized* if there is a single trusted third-party that mediates all messages. Clients are restricted to a server. For example using a messaging service such as WhatsApp, for any two users, all messages must be passed through the WhatsApp server. Slack, Facebook, Signal, Apple Payments, and other popular cloud-based services are examples of centralized architectures.

4.2 Federated

In the *federated* architecture, client may pass messages to different servers, and these servers may communicate to pass messages between clients. This is the classical architecture of the early Internet, where users were often offline and so servers were necessary. Open standards for protocols usually have this architecture, and so clients are not restricted to a single server, as otherwise server-mediated communication between multiple clients on different servers would be impossible. Taking e-mail for example, the user operates a client known as the Mail User Agent (MUA) that sends mail via one or more the Mail Transfer Agents (MTA, i.e. "email servers") until reaching the recipient's server. This server in turn sends the mail to the recipient's MUA. The security of the e-mail ecosystem at least assumes each of these channels is encrypted via TLS (Foster et al., 2015). SMTP services such as Gmail are federated, as would be any service using a standardized messaging protocol such as XMPP+OTR or even "federated" or "permissioned" blockchains.

4.3 Decentralized

In the *decentralized* model – also a "peer to peer" (P2P) model) – clients may directly communicate without a server, and so clients are considered "peers" (Minar and Hedlund, 2001). In a peer-to-peer framework, clients may pass messages on behalf of other clients, although they may also drop messages. Servers are not used. This naturally leads to a default to broadcast messaging in peer-to-peer systems. Earlier systems like Bittorrent as well as newer "permission-less" blockchain-based models fall into this category.

A proper subset of distributed systems that operate in an adversarial model are *decentralized systems*, which are "systems in which multiple authorities control different components and no single authority is fully trusted by all components" (Troncoso et al., 2017) As per malicious security (Yung, 2015), both centralized and decentralized parties may no longer honestly follow the protocol, but may maliciously attempt to subvert the protocol: For example, a peer in a peer-to-peer system may drop traffic or claim falsely to execute part of the protocol while not doing so.

Distributed ledgers, and implementations such as blockchains, allow components of a system to record their activity transparently so that other peers can verify if components are following the protocol and malicious components detected (Halpin and Piekarska, 2017). Distributed consensus protocols are then required to when the components need to have a consistent global view of some part of the system (Lamport et al., 1982), such as a distributed ledger of transactions. Distributed systems that do not require the quorum be set in advance but allow any peer to join are permissionless, and so decentralized. In contrast, if a quorum of existing members or other manual procedure is required to allow a new peer to enter the system, then the manual procedure is a centralized root of trust or the quorum of "trusted" peers equivalent to servers in the federated model in practice.

5 ANALYSIS

The decentralization trilemma[4] states that a system can only have two of three in terms of security, scalability, or decentralization but not all three. On one hand, a system may be both secure and scalable, but not decentralized. Google and other cloud-based services fit in this requirement. In contrast, a system may be decentralized and scalable, but not secure. Systems such as Bittorrent fall in this category (Wolchok et al., 2010). Finally, a system may be decentralized and secure, but not scalable. Blockchains such as Ethereum and Bittorrent would be part of this category.

Federated systems attempts to balance the trilemma by having multiple servers, but in reality this simply spreads the trust from one server to multiple servers. Therefore, federated systems are not "trustless" and so not decentralized, even if they use a blockchain. Some of the examples are subtle: Napster did use a peer-to-peer protocol, but had a centralized search directory. So Napster would count as federated, as the centralized directory server was required for the function of searching torrents, and each peer needs to connect to that server in order to function properly.

By tying the decentralization trilemma to concrete technical properties, we can determine if it actually holds. Every architecture makes certain design

[4]https://github.com/ethereum/wiki/wiki/Sharding-FAQ

choices that privilege one property over another. Although more work is necessary in order to formally characterize each of these architectures, history suggests that these architectures are structurally biased toward certain properties and against others. Table 1 provides a comparison of the choices made by the architectures outlined above.

5.1 Security Analysis

5.1.1 Confidentiality, Integrity, and Authenticity

In general, all architectures are able to implement classical security properties using cryptography. However, authentication is a problem also for all architectures, albeit in different ways. Centralized architectures have low authenticity insofar as a user can only communicate to other users via a server and a server could serve false keys or impersonate its users, as is possible even with Signal (Kobeissi et al., 2017). This critique also applies to federated servers, although out-of-band finger-print verification may help, despite users often being unable to use fingerprint verification to detect a man-in-the-middle attack by a malicious server (Schröder et al., 2016). Peer to peer systems suffer from similar issues, as key discovery can be easily manipulated by a malicious peer. However, decentralization at least allows direct communication of key material and non-custodial key ownership, and so decentralized systems have an advantage as a (possibly malicious) server is not required to mediate key material. Also, decentralization requiring each client to serve as a peer leads to an inability to cover certain use-cases, as security properties need to be enforced over groups, but each peer is meant to be an individual. Security operations over groups are more well-defined but much less commonly deployed (Chaum and van Heyst, 1991). For example, Bitcoin has issues with large group signatures and secure group messaging is still considered a hard problem, even in centralized environments (Goldberg et al., 2009).

5.1.2 Privacy: Unlinkability and Unobservability

Without advanced techniques like differential privacy or secure multi-party computation, centralization has low unlinkability insofar as the malicious server can link any of its users to messages and can always map the network of a user via observing their messages. Likewise for authenticity issues in centralized systems, and the same is possible in federated settings. Decentralized systems by design can offer possibly more privacy as it is less trivial for a central server to monitor all transactions, although not without cost. For example, even Tor is vulnerable to traffic analysis attacks on the exit and entrance nodes. (Dingledine et al., 2004). Some decentralized designs allow a high amount of unlinkability, using techniques like zero-knowledge proofs such as ZCash (Ben-Sasson et al., 2014) and unobservability via the use of cover traffic (Clarke et al., 2000), although these designs are not widely deployed. The problem is cover traffic and other techniques to increase anonymity come at the cost of reducing the capacity of the network, as given by the well-known "anonymity trilemma" between capacity, latency, and anonymity (Das et al., 2018).

5.2 Scalability Analysis

5.2.1 Availability, Capacity, and Latency

Scalability concerns the reliable operations of the system in the face of growing demand. Due to their use of robust distributed (but not decentralized) systems, centralized architectures are highly available and can easily scale to having a high capacity with low latency (DeCandia et al., 2007). Federated systems have as a bottle-neck the server with the least capacity (similar to decentralized systems). In most decentralized architectures, availability is low as peers may always be offline. Although federation may be a disadvantage in terms of unlinkability in comparison to decentralized systems, federated systems typically have much higher availability than decentralized designs as the server is normally online while peer-to-peer systems suffer from network churn. This tends to translate into lower latency in comparison to decentralized systems. In centralized systems, it is much easier to upgrade capacity via simply upgrading servers and connection capacity. In decentralized systems, the problem of the "weakest" link of capacity can be profound, particularly if there is no way to route around low capacity peers. Therefore, real-world decentralized systems like Bittorrent tend to evolve "super-nodes" that have high capacity and are online. These "super nodes" in effect emerge as de-facto "servers" in the decentralized system (Wang and Kangasharju, 2013), undermining the security and privacy properties of decentralized systems and making them de-facto federated systems.

5.2.2 Transparency and Portability

Centralized and federated servers whose code and operations are hidden from their users on a server are difficult for third parties to audit by their users, and so transparency is low. Decentralized systems offer less ability for an adversary to tamper with incoming

Table 1: Properties compared to architectures (* assumes a distributed ledger).

Property	*Centralized*	*Federated*	*Decentralized*
Security and Privacy Properties			
Confidentiality	✓	✓	✓
Integrity	✓	✓	✓
Authenticity	✗	✗	✓
Unlinkability	✗	✗	✓
Unobservability	✗	✗	✗
Scalability Properties			
Availability	✓	✓	✗
Capacity	✓	✗	✗
Latency	✓	✓	✗
Transparency	✗	✗	✓*
Portability	✗	✓	✗
Examples:	Apple Pay, Signal	Visa, E-mail (PGP)	Bitcoin, Bittorrent

messages as the malicious server has been eliminated insofar as messages may be directly sent from user to user, although in practice a relaying peer may tamper with the messages. For example, Bittorrent has no transparent log, and so nodes dropping packets cannot be easily detected.

Although decentralization gets rid of a malicious server, clients may join and act maliciously in a *sybil* attack (Douceur, 2002). Therefore, blockchain technology requires enforcing transparency in a decentralized setting so attacks and failures by peers can be discovered. In terms of portability, decentralized systems suffer from a dearth of interoperable standards, while federated systems are highly standardized (e-mail, XMPP, etc.). The usability of most peer-to-peer systems tend to be poor, while federated systems such as email have been able to compete in terms of usability with centralized systems.

6 OPEN PROBLEMS

Our analysis leads to a number of open problems exist that prevent the further development of decentralized alternatives. Some of the disadvantages of decentralization can be solved by standardization in order to bring the portability of decentralized architectures in competition with well-known federated protocols. Better usability is desperately needed to drive more users from centralized to decentralized systems. Some of the characteristics of architectures could be changed by technical advances: For example, increased usage of cover traffic could help address unobservability when combined with mix networking (Danezis et al., 2010). Yet some problems are more fundamental. In surveys of secure communication (Unger et al., 2015), a pattern starts to emerge: Any serious attempt to build a decentralized alternative that is resistant to a malicious server eventually comes up against similar hard problems.

It is possible to ignore many of these problems if one rules out one of these properties, but users have grown accustomed to contemporary methods of online communication based on silos (Facebook, Twitter, etc.) that are easy to use and have high availability, as well as assume a level of authenticity that is normally not justified by the difficulties of trusted key discovery in centralized systems. So, decentralized architectures that are designed to defend against malicious servers should find solutions to the "seven hard problems" as follows:[5]

1. **Public Key Discovery Problem:** Public key discovery and validation is very difficult for users to manage in a decentralized system, but without it a message cannot have authenticity.
2. **Key Availability Problem:** Users would like communicate seamlessly and securely across different devices, as well as restore data if a device or key is lost.
3. **Group Problem:** Users work in social groups, yet public key cryptography does not have a consistent manner for dealing with groups in decentralized settings. This effects confidentiality, integrity, and authenticity.
4. **Metadata Problem:** Messages are vulnerable to traffic analysis, unless cover traffic is used, which is not common in decentralized systems. This derives from issues around unlinkability and unobservability.

[5] An earlier version of this list was originally defined at LEAP: https://leap.se/en/docs/tech/hard-problems

5. **Sybil Problem:** Without servers, any client may join a decentralized network and act maliciously in a sybil attack (Douceur, 2002).
6. **Update Problem:** Software updates need to be securely delivered across multiple clients in a decentralized system, which is easy to do with a centralized server.
7. **Resource Problem:** It is an open problem on how to let users securely share a resource for real-time collaboration in a decentralized architecture.

These problems cause real-world impact on interest in decentralized architectures both by users and researchers. New standards like IETF Message Layer Security seem to be allowing encrypted user communication to tackle the *group problem*, but it is not clear how to apply to decentralized systems.[6] The *update problem* is partially addressed in terms of security by reproducible builds. The *resource problem* in part is due to the group problem being unsolved in tandem with scalability issues, where decentralized designs have a lack of availability and latency that make sharing a single resource between a group difficult.

7 CONCLUSIONS

The decentralization trilemma is a simplification that can be deconstructed into traditional security and scalability properties. Nonetheless, decentralization solves a real problem: Decentralization prevents a single centralized malicious server from compromising the security of users. There is a kernel of truth in the decentralization trilemma: In broad strokes, decentralization does offer better security at the cost of scalability if end-users manage their keys and can offer higher scalability if sacrificing security is acceptable. Nonetheless, decentralized systems naturally have scaling issues as nodes are more frequently off-line or faulty. Worse, communication can require more hops than in centralized systems, increasing the chance of failure.

The real breakthrough in blockchain technology, in comparison to earlier peer-to-peer decentralized systems, is the use of distributed ledgers to prevent malicious behavior by peers via transparency. This naturally leads to loss of latency due to the nature of distributed consensus protocols (Lamport, 1978). Blockchain technology, while not solving the fundamental issues around availability that plague decentralized systems, does allow malicious servers to be removed and malicious peers to be detected, and could lessen the security problems caused by "super nodes." If peers could be ubiquitously online, they should be more available, which may increase capacity and decrease latency. Yet to do so securely would mean the network would reach widespread usage and users would be highly security-conscious, which appears delusional as most users are not systems administrators.

The low availability of decentralized architectures likely necessitates learning lessons about the evolution of "super-nodes" (high capacity nodes that relay the majority of the traffic) in peer-to-peer architectures, which can act as high availability relays and so bring the benefits of federated designs to decentralized systems. In order to secure these "super-nodes," it does seem to make sense that a specialized class of system administrators would arise in federated systems.

Thus, the technical problem of decentralization is revealed at its core to be a political problem: Centralization to a large extent arises due to the monopoly of technical knowledge by a class of system administrators and programmers. The goal of decentralized systems seems to be to spread this technical knowledge so that all users can autonomously operate and govern their own infrastructure. In this way, federated and centralized models represent the inherent trust assumptions of non-technical users. Historically, humans typically trust friends and associates, as well as deploy a specialization of labor. So a user would likely trust a highly-skilled individual they know in some fashion, perhaps a friend or affiliate at a human-scale institution like a university, to manage technical infrastructure on their behalf. The rise of a non-profit federated systems at the dawn of the Internet maps to pre-existing human trust relationships. As the technical complexity of the Internet scaled and users needed more convenience, this model shifted as users trusted Google to handle the sheer technical complexity in return for profit off of personal data or for a certain cost. The lack of trust in these centralized servers and the humans behind them is what led to the rise of peer-to-peer and blockchain: Decentralization is a political ideology masquerading as a technical architecture.

The next steps to explore this thesis would require formalizing properties such as availability, latency, and capacity in a more rigorous manner than presented here, similar as what has been done in research on privacy (Das et al., 2018). Then architectures could be compared with the same rigor as security properties within a formal message-passing framework as done by the π-calculus (Kobeissi et al., 2017). Another promising avenue is using network-theoretic approaches to study the evolution of decen-

[6] https://datatracker.ietf.org/wg/mls/

tralized networks into super-nodes. Further socio-technical work in needed in understanding the motivations, usability, and political stakes of decentralization. The ideology of decentralization has given us very real advances in technology for ordinary users, but much more is needed for liberation.

REFERENCES

Baran, P. et al. (1964). On distributed communications. *Volumes I-XI, RAND Corporation Research Documents, August*.

Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. (2014). Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE Symposium on Security and Privacy*.

Chaum, D. and van Heyst, E. (1991). Group signatures. In *Advances in Cryptology*.

Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. (2000). Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability*.

Danezis, G., Díaz, C., Troncoso, C., and Laurie, B. (2010). Drac: An architecture for anonymous low-volume communications. In *10th Privacy Enhancing Technologies Symposium*.

Das, D., Meiser, S., Mohammadi, E., and Kate, A. (2018). Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 108–126. IEEE.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. In *ACM Symposium on Operating Systems Principles: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, volume 14, pages 205–220.

Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The second-generation onion router. *Proceedings of the 13th USENIX Security Symposium*, 2.

Douceur, J. R. (2002). The sybil attack. In *1st International Worksop on Peer-to-Peer Systems*.

Foster, I. D., Larson, J., Masich, M., Snoeren, A. C., Savage, S., and Levchenko, K. (2015). Security by any other name: On the effectiveness of provider based email security. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 450–464. ACM.

Goldberg, I., Ustaoglu, B., Gundy, M. V., and Chen, H. (2009). Multi-party off-the-record messaging. In *16th ACM Conference on Computer and Communications Security*.

Halpin, H. and Piekarska, M. (2017). Introduction to security and privacy on the blockchain. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 1–3. IEEE.

Kobeissi, N., Bhargavan, K., and Blanchet, B. (2017). Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 435–450. IEEE.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7).

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401.

Minar, N. and Hedlund, P. (2001). A network of peers – peer-to-peer models through the history of the internet. In Oram, A., editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, pages 9–20. O'Reilly, Sebastopol, CA.

Pfitzmann, A. and Hansen, M. (2005). Anonymity, unlinkability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. Technical report.

Schröder, S., Huber, M., Wind, D., and Rottermanner, C. (2016). When signal hits the fan: on the usability and security of state-of-the-art secure mobile messaging. In *European Workshop on Usable Security. IEEE*.

Slobogin, C. (2014). Cause to believe what?: The importance of defining a search's object–Or, how the ABA would analyze the NSA metadata surveillance program. *Oklahoma Law Review*, pages 14–7.

Troncoso, C., Isaakidis, M., Danezis, G., and Halpin, H. (2017). Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *Proceedings on Privacy Enhancing Technologies*, 2017(4):404–426.

Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., and Smith, M. (2015). Sok: Secure messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249. IEEE.

Wang, L. and Kangasharju, J. (2013). Measuring large-scale distributed systems: case of BitTorrent mainline DHT. In *13th IEEE International Conference on Peer-to-Peer Computing*.

Wolchok, S., Hofmann, O. S., Heninger, N., Felten, E. W., Halderman, J. A., Rossbach, C. J., Waters, B., and Witchel, E. (2010). Defeating vanish with low-cost sybil attacks against large dhts. In *Network and Distributed System Security Symposium*.

Yung, M. (2015). The mobile adversary paradigm in distributed computation and systems. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 171–172. ACM.