NEXt generation Techno-social Legal Encryption Access and Privacy nextleap.eu

Grant No. 688722. Project started 2016-01-01. Duration 36 months.

DELIVERABLE D2.5

Principles for Decentralised Protocol Design

Jaya Klara Brekke and Marios Isaakidis

University College London

Beneficiaries: **UCL** (lead), INRIA, EPFL (former IMDEA), IRI

Workpackage: **Science of Decentralized Architectures**

Description: An abstract framework and easy-to-use cookbook for principles of decentralized and rights-preserving protocol design that finalizes the solutions found to the hard problems of Internet Science for decentralized systems given in D2.1

Version: Draft

Nature: Report (R)

Dissemination level: Public (P)

Pages: 17

Date: 2019-1-22

Project co-funded by the European Commission within the Horizon 2020 Programme.

Introduction

This document has taken the comprehensive overview and hard problems of decentralized systems of D2.1 and distilled this work into a practical cookbook of solutions for decentralized protocol design. This **Decentralization Cookbook** is aimed at developers who want to contribute to the recent wave of efforts towards common infrastructures and digital spaces that are secure from targeting, surveillance and censorship and so preserve the fundamental rights of users. The regulatory environment established by GDPR provides a powerful motivation and context for the development of privacy aware systems that are secure and safe – for opinion to form, knowledges, cultures and information to be shared. Broadly speaking, the cookbook is therefore intended to contribute towards a distinctly European development pathway for digital infrastructures that prioritises privacy and security of data and communication as an integral part of preserving and expanding fundamental rights and the richness of democratic participation.

The cookbook is structured as follows: **Section 1** contains a discussion of decentralization as a specific strategy. It outlines some of the issues that come about when these specific strategies are generalized into a set of principles, most notably that decentralization becomes confused, at times referred to as a principle, at others as an effect of a given systems design. The lesson is that what is true for a given system design is not necessarily true for those using it, or for its broader effects. **Section 2** first outlines a method for decentralized systems designs to overcome these issues. Here, a distinction is drawn between principles, properties and effects, where the properties of a given protocol defines the design space proper, while principles are general assumptions of how to achieve such properties, and the effects are the ways in which a given system interacts with a given context. Three projects developed as part of the NEXTLEAP project are then examined as case studies, namely AutoCrypt and MLS. A cookbook is established for the four key properties of decentralized design, namely Privacy, Transparency, Security and Availability. The cookbook concludes with **Appendix A**, a unified and concise vocabulary for interdisciplinary decentralized protocol development for internet science, clarifying the more complex and disciplinary vocabulary in D2.1 to reach a common interdisciplinary understanding.

Contents

1 Why Decentralization?

This guidebook is aimed at developers who want to design decentralized protocols in order to build ”real world” decentralized systems to effect change in the world. We distinguish *protocol*, the abstract design used to architect the communication of information in information systems, from the particular *systems* themselves. Decentralized systems were developed largely as strategies to counter the control and censorship of information. Distributed systems have no single point of failure for reasons of availability. Decentralisation is defined as a subset of distributed systems: ”A distributed system in which multiple authorities control different components and no single authority is fully trusted by all others”[TIDH17]. The motivations for decentralised systems go further than distributed systems. and are concerned with censorship resistance, privacy and transparency – properties that are distinct to systems with no authorities. To put another way, decentralization is about the social effects and assumptions of the system: Distributed systems have no single point of failure, but decentralized systems have no single point of *trust*, and trust is an irreducibly social concept. Decentralized protocols are therefore practically employed as a design strategy that would make the targeting and shut down of a given information or transaction system impossible. However, this change from decentralized protocols as specific strategy for circumventing authority, to a generalized techno-social proposition has had some serious implications that require some untangling.

1.1 Decentralization as strategy

Early efforts towards decentralized systems were built in an ad-hoc manner, and largely as a strategic response to particular threats of shut down (see D2.1). For example, the development of filesharing system BitTorrent was in response to the shut down of Napster and a deliberate strategy to avoid legal prosecution for breach of IP in file sharing networks. A system being run by one person or a small group of people implies that system is fully dependent on these, technically as well as legally - which also means it can be shut down by arresting the group and/ or turning off their servers. In contrast, decentralized systems like BitTorrent do not have any central servers and involve many people who do not even know each other, in running the system. If anyone is arrested or has their servers seized, the system continues to run regardless. A similar strategy of decentralization can be traced for Bitcoin in response to the shut down of E-Gold and Digicash. The purpose of a given decentralized design was therefore particular, practical and usually well understood by the communities building, sustaining and using it. Decentralized systems are designed to be resilient to targeting by authorities.

In recent years there has been a broader interest and uptake of decentralization, and in the process these motivations and the attraction of decentralized technologies have become generalized – from a particular strategy vis-à-vis particular companies and regulations, to a general proposal for entirely replacing corporate, legal and state institutions. The context for such a generalization, largely through the emergence of blockchain technology, was the 2008 financial crisis, and shortly after, the Snowden revelations of mass surveillance operations facilitated by telecommunications companies and Silicon Valley for the NSA. These events brought attention and awareness to the surveillance-based business models of and close ties between near-monopoly tech companies and state security agencies. In this context, both decentralization and the notion of authority took on broader meaning and decentralization became a technical, political, economic and social aim in and of itself, reaching outside the “hacker” circles of the early p2p systems. This also meant that the technical and strategic aspects of decentralization, as operationalized in peer-to-peer networks, became associated with a more vague interpretation of the concept, related to social, economic and political hopes and desires. This vagueness has been very productive in its own right, mobilising interest and efforts across otherwise pernicious political, economic or ethical divides, and leading to a remarkable burst of funding in 2017 via the ICO (”Initial Coin Offering”) phenomenon. Yet it has led to confused uses of the term, conflating of meanings and effects, also in the more technical fields. The generalization of the idea of decentralization, via its particular history from peer-to-peer, has made decentralization an aim in and of itself with little understanding of intent or assessment of actual effects. For developers, we hope to be able to clarify their intents and connect them to both technical principles and social effects.

1.2 Decentralization generalized

The provenance of current understandings of decentralization in early peer-to-peer strategies of systems resilience and circumvention of authority has some subtle but important implications for technical, as well as social and political understandings of decentralization. The generalization of decentralization as a strategy into an aim in and of itself has brought with it some important complications for protocol design, three of which are discussed here.

1.2.1 Limits to the design space

A decentralized protocol is specific in that it is resilient to authorities. The social and political reasons for such systems have been justified in the histories of shut downs and legal cases, and expanded to a broader societal concern through mass surveillance used for commercial and military targeting. These events point to an understanding of the concept of authority as external, powerful entities that might target a system to shut it down. But in protocol design for open decentralized systems, authority is conceived in slightly different ways, namely referring to any potential aspect of the system that the system as a whole is dependent on. This, from a security perspective, makes sense. Because if there is any part of the system that it as a whole depends on, this can, in turn, be a target. In the meantime, the generalization of decentralization brought with it an assumption of solving the problem of authority more generally. This entails an ever expanding set of concerns in terms of what might or might not become an authority in a given systems design, in particular open systems. For example, a given protocol might be run across a network of nodes, but if it depends on specialised hardware (such as Intel SGX) that is only produced by a particular company, this would also present a kind of authority. Or, a given protocol might be run in a decentralized manner, but if the reference client is maintained and updated by one or only a handful of people, then these might present a kind of authority in the system. As pointed out in D2.1, even a single team of developers maintaining the protocol could be a source of authority, even a single team of developers maintaining the protocol could be a source of authority. Decentralization, as a solution in and of itself raises the question of what is considered to matter in the design of open decentralized systems, in other words, the limits to the design space.

There are many other ways in which problems with authority can and are addressed. The particular design properties, of having no authorities, make a lot of sense when seeking to avoid prosecution and shut down. But developing systems with no authority can complicate questions of accountability for when things go wrong. For some purposes, explicit authorities with strong accountability structures are more appropriate from the standpoint of the liberal concept of the individual, because responsibilities, and therefore recourse to justice if and when things go wrong, thereby becomes more explicit. Instead, the burden of risk has been shifted onto those using the systems. This needs to change if decentralized systems are to become widespread, meaningful and useful alternatives to existing centralized systems. There is scope to rethink how accountability might operate in decentralized systems, reputation being one of the more commonly explored form, with attempts at mapping social reputation a promising start. It is also worth considering what aspects of accountability are already addressed by other contexts and systems, including a given communities norms and practices or existing regulation. In other situations, accountability on the basis of for example transparency, can be a way to ameliorate authorities that a given systems design cannot eliminate for one reason or another.

1.2.2 Defending user rights

Another subtle and important legacy of the early years of strategic deployment of decentralization has been the conflation of the interests and condition of the system with that of a given person using the system. This can be disentangled by looking closer at BitTorrent for example, as a protocol has proven very effective and resilient towards legal prosecution and shut down by not having a single point of failure. Thbe resilience, however, is of the BitTorrent network itself, not necessarily the individuals using it, who might still face prosecution. Equally, Bitcoin as a system is considered a “trustless” and secure system, but this has not meant that those using

Bitcoin have not had to trust a given exchange for example, had funds stolen and so on. Often times, these contradictions between the interests, survival and well being of a system itself and the interests and well being of those using the systems are dismissed, any fallout being blamed on a given individual user rather than any fault of the systems design. Decentralization might make the overall system resilient to authorities, but it does not necessarily mean those that use it are. In fact, decentralized systems often shift risk from the system itself to the individual using it.

1.2.3 Beyond decentralization as dogma

Finally, technical protocols have been confused with assumed properties, uses and aggregate effects. Decentralized protocols are too readily assumed to bring about decentralized political, social and economic outcomes. Decentralization as determined in and by a networked computer system does not relate in any linear causal manner to social, political and economic notions of decentralization. This conflation means that decentralized protocols have become in and of themselves with little measure and assessment of whether they are achieving the assumed social and political effects. Efforts and improvements are made in the measure of decentralization for systems designs, but there is still far to go in terms of effective methodologies and interdisciplinary efforts towards understanding the effects of decentralized digital infrastructures and applications.

Addressing these issues entail approaching decentralization carefully and with clear definitions. It means starting from a concern and care for what a given system is supposed to achieve for those using it, bearing in mind that, indeed: “The ultimate security bet of decentralized systems is also unknown: Is being vulnerable to a random subset of decentralized authorities better than being vulnerable to one?” (D2.1 p. 28). In this deliverable we might add, the ultimate bet of decentralized systems design is still unknown: can principles for decentralized systems design be generalised without losing sight of who benefits from these? Can we create decentralized systems that benefit the common good?

2 Decentralized Design Cookbook

Decentralization, as defined in D2.1, is a subset of distributed systems in which a given system is not dependent on or controlled by any single trusted authority. As discussed above, decentralization as a systems design principle is often confused with decentralization as an ethical, political or economic ideas and this makes for many varied assumptions as to the properties and effects of decentralized systems. In order to navigate such confusions in terms of protocol design, the specific, desired properties of a given system need to be established first. More general principles can be drawn on as ways of achieving such properties, knowing that they are not necessarily guaranteed by these because so much depends on the contexts and specific designs. In turn, the effects are highly contingent on the context of protocol deployment. If principles cannot be finally established and effects never fully known, what then should be considered the design space and responsibility of protocol designers?

While decentralization seems to resolve issues of security and authority in the protocol design, their impact on the diverse and emerging actors using the systems are not well understood. The main concerns of early decentralized peer-to-peer systems has been to avoid authorities, such that the system is reliable, resilient and can survive attacks and attempted shut down by formidable adversaries. Yet what is true for the system is not necessarily true for those using the system. For this reason, we suggest a distinction between principles, properties and effects.

Principles are the assumptions of what kind of abstract patterns of protocol design tend to have certain properties. These are considered assumptions because the properties that are realized in practice are contingent on both the context of deployment and potential trade-offs with other principles. *Properties* describe the particular technical characteristics of a specific systems design, what that particular design achieves. *Effects* describe the ways in which a system plays out in particular social contexts and environments and for different kinds of actors.

The effects can and often should be measured in order to assess whether a given principle used for a particular design indeed has the intended properties. The suggestion is therefore to understand the decentralized protocol design space as concerned primarily with the task of achieving specific properties, while principles are an aide for achieving these and effects help measure, adjust and correct to achieve the desired properties.

2.1 Principles

Principles are general patterns for protocol design that contain assumptions about what kinds of designs will achieve certain properties and effects. Below, we describe three major principles in decentralized protocol design, followed by a set of questions to assess whether this principle is relevant for you to apply. Examples are given to help understand the differences, using both already deployed systems and blockchain-based designs.

2.1.1 Decentralization (authority)

Decentralization implies a protocol design with multiple authorities were no single authority is trusted by all. The aim of decentralization to ensure resilience against shut-down by authorities, censorship and manipulation. A decentralized authority topology in a system can do this, but conditions external to the network authority topology and protocol proper might affect the desired properties - for example protocol governance; concentration of token holdings and market manipulation in token-based systems; emergent centralizing tendencies and so on. Decentralization can be understood in many ways, and as a principle might be understood or assumed to be related to political equality or fairness. A decentralized authority topology in protocol design might help further such aims, but it does not guarantee it. Here we address decentralization specifically as a network authority topology in relation to trust, i.e. which components are actually trusted.

Do you trust all the components of the protocol?

Yes You don't need decentralization

No Are only some of the components trusted?

- **Yes** Set of authorities for the trusted components
- **No** Anyone can be an authority, the design needs to be trustless and fully decentralized.

Examples: Answering "Yes" to the first question leads you to "Do you trust all the components of the system?": Classical cloud-based services from Silicon Valley, such as Google, Facebook, and Amazon, may be implemented as distributed systems, but they require ultimate trust in the company and trust in all its components, the entire "cloud platform" of the company. Answering "no" but then noting we only need a "set of authorities for the trusted components" leads to the classical federated infrastructure of the early Internet and Web, where anyone can run a website or e-mail service provider, but there is still trust in the domain name system and users tend not run their own infrastructure, but use a trusted server. However, if the answer is "no", then no components are trusted, so each user must be their own authority, as is the case in P2P systems such as Bittorrent and blockchain systems such as Bitcoin.

2.1.2 Openness (operations)

How open a system defines who can join the operational system, for example, in a P2P system who can run a client and become a peer. Operations refers to the actual running of the system, rather than its design. A fully decentralized system implies a level of openness, otherwise those running the system would essentially operate as trusted authorities. However, there can be benefits to having aspects of the system run by trusted authorities to avoid things like churn, to ensure availability and trace accountability in how the network is run (see D2.1). Also, not all systems are designed for anyone to use, and therefore not necessarily anyone should be able to

join as a peer. If you have a defined community that your protocol is serving, having it run by a network of peers from that community and excluding others from joining in order to protect certain properties might make sense.

Does the protocol have a closed group of participants?

Yes Your protocol creates closed, but possibly decentralized, systems. Use techniques such as *consensus protocols* for distributed systems with unreliable nodes, like Paxos [Lam98]. If you want to defend against adversarial participants, see protocols that provide Byzantine Fault Tolerance [LSP82].

No Are there conditions for joining the system?

Yes Your protocol creates an open system, but with requirements. Consider techniques from both *proof of work* and new *proof of stake* systems, in which participants need to either exhibit some "hash power" in mining or use some tokens in order to join the consensus protocol. There are also decentralized forums that one can join only by invitation by an existing participant, as in Briar[1].

No Your protocol creates a totally open system. However, this may lead to attacks. Consider techniques inspired from mechanism design and game theory, like the tit-for-tat incentive in BitTorrent, and reputation systems like the Freenet Web of Trust. Such techniques are needed to prevent Sybil attacks, spammers, and free-riders (see D2.1 p21).

Note that the ability to join the system might be hindered by other things like hardware requirements. A system claiming to be fully open might in fact simply have hidden forms of authority based on non-explicit requirements, as in the widespread use of specialized hardware in Bitcoin mining.

Examples Openness defines not trust per se, but whether or not anyone can join the "operations" of system and what conditions there are on joining the system. Taking the example of messaging systems, classical Silicon Valley systems like WhatsApp, but also applications like Signal, do let anyone get a user account, but you cannot autonomously run your own WhatsApp or Signal server and still connect to other users, and so such systems are closed "wall gardens" just as much as Facebook and other social networking sites. On the other hand, if there are "no conditions" for joining the system, one has the classical problem of e-mail, where spam becomes a problem. The invention of HashCash [B$^+$02], also used in Bitcoin mining, was originally done to put some conditions on joining an email system. In terms of blockchain-based systems, closed "permissioned" blockchain systems such as Hyperledger[2] are not open, but other "permission-less" systems such as Bitcoin or Ethereum let anyone join who can demonstrate commitment via hashing. More complex "proof of membership" can always be used to ensure a given participant can be part of the system. Systems that are truly open are quite rare, but include P2P systems such as BitTorrent, which does not require seeding to run a node, or Freenet[3].

2.1.3 Co-production (governance)

Decentralized systems also tend to be based on principles of co-production, to let the components in the system – including users – produce a commonly desired goal jointly. This follows from the intention of designing systems without authority. If only one person writes and maintains the code for example, this person can become an authority, compromising decentralization. Instead, in many decentralized systems, people can contribute to the code, run a client or even fork and develop a new version of a protocol. In blockchain systems, this extends to verification and protocol level consensus. Note that co-production does not only entail multiple contributors to a given code repository, but also in the development methodology, for example in user-centric design methods or interdisciplinary approaches. Decentralization entails that the power and say over protocol design is brought closer to those who will use or will be affected by these – co-production with and by those affected by a given systems design. The question of governance is further explored in D3.6 with a number of governance case-studies.

[1] https://briarproject.org/
[2] https://www.hyperledger.org/
[3] https://freenetproject.org/

Can anyone contribute to the protocol?

No You have closed-source protocol design and software.

Yes You have open source software with the possibility of forks. Do you have a well-defined process for contributing code?

Yes You should have a formal governance or defined standards process.

No You have a closed formal or informal leadership.

Examples Co-production is more complex that a simple dichotomy between open source and closed source software. Closed source software is when only a particular company owns the code, as exemplified by Microsoft's control over Skype. Open source software examples includes software like the Transmission BitTorrent client or Bitcoin core, which are collaboratively developed over public repositories. However, although anyone can propose a modification, the final decision on whether they get accepted is made by a closed group of trusted maintainers, and so there is closed or informal leadership. In contrast, the W3C and IETF maintain a formal standards process around WebRTC, where anyone can contribute the protocol but there is a well-defined process, and the Bitcoin community has developed a similar process via the BIP (Bitcoin Improvement Proposals) although the final decisions are still made in a closed manner. In the free software, contributors can clone the repository and release their own versions of the software, and in the blockchain space this leads to forking such as Bitcoin Cash or Ethereum space. Note that open protocols can be implemented as closed source systems – e.g. the Blizzard Entertainment update client that is based on a closed source version of the BitTorrent protocol.

2.2 Properties

Properties are what technical goals a given protocol attempts to achieve. They are the desired properties, and are therefore often seen as the reasons for forks or new projects emerging that suggest new and better ways of achieving such properties if and when they turn out to not be working in a satisfactory manner. Networks can have emergent effects, and so the properties might change over time and so need to be assessed to see if the desired property is being achieved for the designed person, group or system. Below, we describe four properties that are commonly desired from decentralized protocol designs, followed by examples of where such properties are achieved for different authority topologies, from a centralized (single trusted authority) to increasingly decentralized (multiple authorities) and eventually completely decentralized (trustless).

2.2.1 Privacy

The technical aspects of a protocol related to the protection of users' related data (identities, actions, etc.). As outlined in D2.1 this protection is usually formalized in terms of privacy properties (anonymity, pseudonymity, unlinkability, unobservability). Privacy is one of the most prevalent properties that are sought in decentralized systems. In the context of surveillance-based monopoly business models underpinning a large part of current day digital platforms and infrastructure, it remains one of the most socially, economically and politically potent engineering tasks (cf. [Rog15]).

Who do you want to hide users' data (identity, actions etc) from in your protocol?

Single authority, trusted You need to hide user data from the trusted authority, usually a central server. One technique is differential privacy, a privacy-friendly statistical technique for aggregation of data (see D2.4 Section 2). Freenet is using a similar mechanism that adds noise to network-related measurements collected from nodes by a central trusted authority [ope].

Set of authorities – semi-trusted You need to hide data from a group of authorities. One technique is secure multi-party computation, a cryptographic method that allows authorities to jointly process the intersection of their private data without revealing them to one another [Gol98], and zero-knowledge proofs, for proving

to others possession of some secret information [GMR89]. Such techniques have been applied to genomic privacy systems and authentication systems [IHD16, KLI+18].

Everyone is an authority – trustless In this case, you should attempt to hide user data from even unknown and powerful third-parties. For concealing metadata about user actions and communications (confidentiality from peers as defined in D2.1) decentralized protocol designers can use either cryptographic or network architectures that mix the user data flows. Zerocoin [MGGR13] and Monero [Noe15] are cryptocurrencies that leverage ring signatures for hiding the sender of a transaction within a set of possible users. Tor and mix-nets can provide unlinkability of user communications by routing messages through various peers, in a way that when peer receive a message they cannot know who was the original sender.

2.2.2 Security

The security aspects of a system, as defined in 2.1, are those that encompass traditional information security properties: confidentiality, integrity, and authentication. These security properties normally involve cryptography, and so depend on the secrecy of key material in modern systems. There are also less traditional security properties such as authorization, plausible deniability or non-equivocation.

Do you need to assure that information in your protocol is accessible only by the intended recipients and has not been altered by third-parties?

Single authority, trusted In these protocols, key material tends to be maintained by a single authority. For example, classical PKI systems like client certificates used in web-browsers depend on centralized X.509 certificates [CDH+05]. Often even privacy-enhanced systems require single trusted authorities to maintain security properties, such as UnlimitID, a privacy enhancement to OAuth, that leverages anonymous attribute-based credentials to prevent Identity Providers from learning which services their users connect to, but a single provider both generates and proves the credential is correct [IHD16].

Set of authorities – semi-trusted For this type of protocol, techniques such as threshold cryptography are often used, so only a subset of signatures need to be validated. One example is Tahoe-LAFS[4], a decentralized storage system that, via the use of threshold cryptography, assures that the authorities that provide the infrastructure cannot by themselves decrypt users' files.

Everyone is an authority – trustless Each end-user maintains their own keys in these protocols. For example, see Simple Distributed Security Infrastructure (SDSI), an earlier competitor to traditional PKI systems [RL96]. Another example is Bitcoin where every single node is verifying the cryptographic signatures to make sure that the protocol has been executed correctly and the control over value is determined only by ownership of the key by the user.

2.2.3 Transparency

Decentralized systems, to some degree, promise a level of transparency, where the functions of the systems are visible to the other components in the system, or even the users of the systems. These aspects of the system can be quite varied – its network flows, its authority topology, as well as its code.

Do you need to hold authorities accountable for their actions by users or other components of the protocol?

Single authority, trusted These protocols tend to use centralized auditable logs. See verifiable systems like Helios Voting, an election system that produces an auditable log of how the central authority is processing users' votes [Adi08] or Certificate Transparency that logs the SSL keys of websites [LLK13].

[4] https://tahoe-lafs.org/trac/tahoe-lafs

Set of authorities – semi-trusted These protocols can use a network of auditable logs that can determine if one authority is attempting to create a false record (equivocate) for some user or other authority, which is difficult to prove otherwise. For example, see CONIKS, an auditable PGP Public Key Infrastructure hosted by email providers [MBB+15]. The threat model of CONIKS is that equivocation by authorities (the email providers) is detectable by users and other authorities. This is actually possible without revealing any information about the users.

Everyone is an authority – trustless These protocols have every member feature a verifable log with P2P gossiping. Bitcoin is a one example. In terms of key management, see ClaimChain, a protocol for sharing own key material and gossiping the keys of others [KLI+18]. In ClaimChain there is no trusted authority, and each user maintains their own verifiable log, but owners have fine-grained access control on who can read their contacts, and others can detect if the owners have gossiped different versions of their ClaimChain to others.

2.2.4 Availability

Protocols can make sure information is accessible to users when they need the information. Although centralized systems typically achieve availability much easier than decentralized systems, decentralized systems can ensure availability in the face of formidable adversaries by making the system near-impossible to shut down in terms of censorship resistance. Decentralized systems that do not have incentives – whether through strong community building, clear shared aims, or economic incentives – might face issues in terms of availability as people lose interest and nodes drop out. Ensuring availability for decentralised protocols entails both resilience towards authorities as well as clear a understanding of the incentives of those taking part.

Do your users need information to be accessible on demand?

Single authority, trusted These systems are exemplified by cloud-based platforms, where users can easily find their information. Highly accessible file storage can be massively replicated via using distributed systems technology, but with a single authority.

Set of authorities – semi-trusted These protocols are federated, and allow users to host their own information or chose where to retrieve their information from. An example would be federated file-showing systems like Nextcloud[5].

Everyone is an authority – trustless The protocols require users to host their own information and allow them to chose to host other users information. One example would be the InterPlanetary File System (IPFS) [6], where a blockchain-based identifiers for content-based networking and gossiping allows, in theory, available decentralized data storage.

2.3 Effects

Effects are the ways that protocol designs play out in specific contexts and for specific participants, including users of a system. Commonly desired effects of decentralized systems include things like *disintermediation* - either in the sense of getting rid of intermediating aspects that might censor or control activities, or in the sense of adding barriers to access; decentralization of power and decision-making; empowerment by being in control of infrastructures that immediately affect our lives; autonomy by having the capacity and possibilities of co-creating the infrastructures that we need when those that already exist no longer serve us. The effects of decentralized systems can never be general, they will always play out differently for different contexts and people, and have emergent tendencies.

[5] https://nextcloud.com/
[6] https://ipfs.io/

Another intended effect could be *autonomy*, both negatively conceived independence from external control as well as positively conceived as the development of capabilities. For example, Bittorrent enabled the bypassing of restrictive copyright and other controls meant to enforce payment by external media companies by end-users, even though it was a generic file-sharing protocol. As Bittorrent also provided users new capabilities in terms of sharing information directly, it increased the autonomy of end-users in unexpected ways. This autonomy both allows short-cutting the traditional models of control of copyrighted media, but created entirely new collective expectations about having low latency access to only the particular media needed, for example, access to a particular song rather than the entire album. This led eventually to new streaming media business models, including both for-pay (iTunes) and advertising-supported (the original Youtube) revenue models. The increased autonomy of users allowed users to make concrete gains in creating a new commons for media, but equally it created new expectations – and thus businesses – that recuperated P2P streaming for both open standards (such as W3C WebRTC [web18], which features a centralized authentication service unlike Bittorrent) and commercial companies such as Skype.

It is therefore important that the intended effects are clearly defined up front, with a clear idea of who and what these effects are intended for, such that they can be assessed and corrected for if necessary. A decentralised system might in fact exacerbate problems of intermediation, becoming another complex layer to navigate with little recourse to justice if things go wrong; it might in fact be disempowering, decision-making entailing unknown risks in complex systems; barriers to entry might prevent newcomers from contributing in any meaningful way, and so on.

3 Case studies

3.1 Autocrypt

Autocrypt, as defined in D2.3 and D5.2, is an enhancement of PGP-encrypted e-mail. We do not use MLS (Message Layer Security) as an example as the protocol is a general purpose protocol for group messaging that can either work in a centralised or decentralised setting. In contrast, the Autocrypt protocol is interesting because it is a partially decentralized system, as it is based on the federated design of SMTP-based e-mail, but also it has some components that are decentralised to the user, such as key management. With ClaimChain, the key management aspects make Autocrypt a very unusual category: A trustless encrypted e-mail protocol. We categorize Autocrypt using ClaimChain (as given in D5.3) using the cookbook:

Decentralisation: Do you trust all the components of the protocol?

No Are only some of the components trusted?

Yes There is a set of authorities for the trusted components. The server is not completely trusted, but the user client is trusted in PGP-encrypted e-mail. There is a set of authorities for the trusted components, namely the server is trusted to transmit and exchange the Autocrypt headers, with other authorities, although they are untrusted to read content of the e-mail. Note that the Autocrypt headers generated by a trusted client e-mail software on the user's client.

Openness: Does the protocol have a closed group of participants?

No Are there conditions for joining the system?

No Your protocol creates a totally open system. The only condition for a user to join an Autocrypt-enabled eco-system is to create an account on an e-mail service provider. This may lead to attacks, so e-mail service providers may wish to vouch for their users. Also, anyone can become an email service provider. This can lead to malicious e-mail provider attacks, so techniques such as a "Web of Trust" given by ClaimChain that holds both other users and service providers accountable for their key material needs to be given.

Co-production: Can anyone contribute to the protocol?

Yes You have open source software with the possibility of forks. Do you have a well-defined process for contributing code?

No You have a closed formal or informal leadership. Indeed, one of the problems with Autocrypt is although it is a community-led project, the editors of the specification still have considerable power, although it is checked by the power of the implementers. Eventually moving to a well-defined standards process, such as via the PGP IETF Working Group, may help if decisions become more difficult over time.

Privacy: Who do you want to hide users' data (identity, actions etc) from in your protocol?

Everyone is an authority – trustless In this case, you should attempt to hide user data from even unknown and powerful third-parties. Autocrypt, although it features service providers as semi-trusted authorities, essentially gives user the power to be an authority via their control of key material. The goal of Autocrypt is also hiding e-mail data from powerful adversaries, including the NSA. Therefore, in the future Autocrypt-enabled e-mail should look to look into more advanced techniques to hide user identity both from semi-trusted e-mail providers as well as techniques such as mix-nets, developed by the PANORAMIX EC project, [7] in order to defend against high-level adversaries.

Security: Do you need to assure that information in your protocol is accessible only by the intended recipients and has not been altered by third-parties?

Everyone is an authority – trustless Each end-user maintains their own keys in these protocols. This is a long-standing tradition in PGP. Although there are weaknesses in Autocrypt enabled PGP (detailed in D2.3) so that it only maintains confidentiality but neither integrity or authentication, future work needs to be done to make the protocol more secure to prevent a malicious semi-trusted e-mail server

Do you need to hold authorities accountable for their actions by users or other components of the protocol?

Everyone is an authority – trustless These protocols have every member feature a verifiable log with P2P gossiping. In terms of key management, Autocrypt uses ClaimChain, a verifiable log protocol for a user sharing their key material and privacy-enhanced gossiping keys via embedding keys in headers [KLI+18].

Do your users need information to be accessible on demand?

Set of authorities – semi-trusted The availability of encrypted e-mail depends on a federation of e-mail providers being available.

3.2 Conclusion

This cookbook, as any level of abstraction, simplifies the underlying complexity of reality of real-world system deployment. Take for example, the division between the properties is often not so clear and the various properties are densely intertwined: The question of privacy is in relation to another commonly desired property of decentralized systems, namely transparency, remains one of the more complex questions pertaining directly to authority topologies and questions of who or what should be transparent and what aspects of the system should preserve privacy. To answer this and design for it in any meaningful way requires an analysis of power, such that transparency can be enforced for the "powerful" – architectures, organisations and people that determine the conditions for others without their say; while privacy is secured for the rest.

This leads us straight from protocol design to unavoidable questions of politics. Another legacy from early peer-to-peer systems is an approach that dictates transparency for the powerful, privacy for the powerless [Hug93, JA12], but this distinction between powerful and powerless becomes less clear within and for decentralized

[7] https://panoramix-project.eu

systems themselves. Who is powerful and who is powerless within a decentralized system? What is considered to matter in terms of power? CPU power, number of bitcoin, social graph, commit access to the reference client, or what about everything else, like dollar wealth, social status, connections to policy makers etc.? Advanced cryptography can begin to resolve some of these issues. For example, attribute based encryption for selectively revealing only the necessary information in order to use a given system; or zero-knowledge proofs as a way to establish trust while retaining secrecy, with potential uses for new accountability structures.

Therefore, this cookbook is not a complete work: The numbers and kinds of principles, patterns, and techniques will change over time. However, as a starting cookbook for developers we hope this cookbook, this cookbook tries to both begin de-mystify the slogan of ”decentralisation” while clearly laying out solutions to the hard problems facing protocol developers.

References

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[B+02] Adam Back et al. Hashcash-a denial of service counter-measure, 2002.

[CDH+05] M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph, and R. Nicholas. Internet x.509 public key infrastructure: Certification path building. RFC 4158, RFC Editor, September 2005. `http://www.rfc-editor.org/rfc/rfc4158.txt`.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[Gol98] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

[Hug93] E. Hughes. A Cypherpunk’s Manifesto, 1993. [Online; accessed 11. Jan. 2019].

[IHD16] Marios Isaakidis, Harry Halpin, and George Danezis. Unlimitid: Privacy-preserving federated identity management using algebraic macs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 139–142. ACM, 2016.

[JA12] Assange Julian and Müller-Maguhn Andy. Cypherpunks: Freedom and the future of the internet. 2012.

[KLI+18] Bogdan Kulynych, Wouter Lueks, Marios Isaakidis, George Danezis, and Carmela Troncoso. Claimchain: Improving the security and privacy of in-band key distribution for messaging. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, pages 86–103. ACM, 2018.

[Lam98] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

[LLK13] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate transparency. Technical report, 2013.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[MBB+15] Marcela S Melara, Aaron Blankstein, Joseph Bonneau, Edward W Felten, and Michael J Freedman. Coniks: Bringing key transparency to end users. In *USENIX Security Symposium*, volume 2015, pages 383–398, 2015.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.

[Noe15] Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptology ePrint Archive*, 2015:1098, 2015.

[ope] operhiem1. Freenet Statistics.

[RL96] Ronald L Rivest and Butler Lampson. Sdsi-a simple distributed security infrastructure. Crypto, 1996.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. *IACR Cryptology ePrint Archive*, 2015:1162, 2015.

[TIDH17] Carmela Troncoso, Marios Isaakidis, George Danezis, and Harry Halpin. Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *Proceedings on Privacy Enhancing Technologies*, 2017(4):404–426, 2017.

[web18] WebRTC 1.0: Real-time Communication Between Browsers, Sep 2018. [Online; accessed 22. Jan. 2019].

Appendices

A Vocabulary for interdisciplinary design

The purpose of this vocabulary is to contribute to efforts started in D2.1 to establish an interdisciplinary vocabulary for decentralized protocol design. The following are additional concepts pertaining mostly to the contributions of this deliverable and concepts to have emerged in decentralized protocol design since Bitcoin.

Autonomy Independence from external influence and control.

Availability Information is accessible to users when they need the information.

Consensus Common agreement on the state of a system.

Consensus protocol The method for getting multiple nodes to agree on a system state.

Co-production The process of collaborative creation through the involvement of different groups.

Decentralization A distributed system with multiple authorities were no authority is trusted by all.

Disruption The effect of a revolutionary concept on a market (or way of doing things).

Effect The ways that protocol designs play out in specific contexts and for specific participants, including users of a system.

Generalization The deduction of general, abstract concepts from particular instances.

Hard fork A fork in the code, chain or project that is not compatible with previous versions.

Mining The process of participating in a blockchain consensus protocol by expending resources.

Soft Fork A fork in the code, chain or project that is compatible with previous versions.

Tokens Measures of account representing a relationship - for example of credit and debt.

Tokenization Implementing tokens into a systems design - for example to visible flows and resources, and award contributions.

Trustless A characterization for systems that do not rely on any trusted authority whatsoever except the end user.

Open When anyone can join an operational system, possibly subject to conditions.

Permissioned Networked systems in which participation is regulated on the basis of access to resources, reputation, or via another access control mechanism.

Permissionless Open participation peer-to-peer systems in which anyone can join.

Principles General patterns for protocol design that contain assumptions about what kinds of designs will achieve certain properties and effects.

Privacy The technical aspects of a protocol related to the protection of users' related data.

Properties The technical goals a given protocol attempts to achieve.

Protocol The abstract design used to communicate information

Security Confidentiality, integrity, and authentication, normally involve cryptography, and so depend on the secrecy of key material in modern systems.

Transparency When the functions of the systems are visible to the other components in the system or the users of the systems.