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

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

DELIVERABLE D5.1

Identifying Communities, Use cases, Development flows

Holger Krekel (Merlinux), Max Wiehle (Merlinux)

Beneficiaries:	**Merlinux** (lead)
Workpackage:	Identifying Communities, Use cases, Development Flows
Description:	This deliverable will select the projects (including but not limited to other CAPS projects) where we aim to integrate the decentralized protocols, describe accordingly their various use cases for these protocols, and outline development guidelines and roles within the development and validation processes.
Version:	1.0
Nature:	Report (R)
Dissemination level:	Public (PU)
Pages:	22
Date:	2016-06-30

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

Table of Contents

1 Introduction

1.1 Perspectives on decentralized messaging

The topic of “decentralization” has receiv ed increased interest since the Snowden revelations in 2013, but what it actually means, and what follows, depends on who is talking and from which perspective. Many new efforts to “re-decentralize the internet” have sprung up, among them the NEXTLEAP project itself which we are part of and which aims to research and develop decentralized messaging in interdisciplinary ways.

Instantiating networks of programs which then communicate with each other without requiring an all-knowing, all-powerful central mediation point are a recurring topic in computer science, a sort of holy grail in distributed computing. However, programmers often focus just on algorithmic decentralization, i.e. only considering how programs communicate with each other. This leaves out questions of who models, creates the programs and who operates, runs them and in which political-economic contexts. After all, internet “giants” like Google or Facebook extensively use and publish Open Source software and apply algorithmic decentralization internally, all for use within their operational realms and for their own profits.

Re-decentralizing digital communications involves dealing with the techno-socio-economic federation of creating and operating software platforms. Otherwise, new distributed algorithms can just be incorporated by current or future major centralized players like they have been so successfully in the past. This is so because central intermediaries have the means to wrap Open Source technologies into convenient end-user facing products and services because they can, often behind the curtains, monetize their mediation role in massive ways, in turn providing them the means to repeat the incorporation cycle. In other words, centralized operations are probably less a technical necessity and more an effect of economic currents. So what do these predispositions mean for our programming efforts on “decentralized messaging”?

1.2 Community agency for operating infrastructures

We see developments of messaging protocols as part of a wider effort to enable communities and organisations to operate and control their own communication infrastructure. Socially, this means for us engaging with a wide variety of developers and users in the decentralization space. Messaging is, after all, one of the fundamental concepts in computing and has been discussed intensely since around 1950 by cyberneticians Norbert Wiener ("Communication and control in the animal and the machine") and in particular by Stafford Beer, who discussed decentralization and human agency vis-a-vis machine communication and management. However, programming communities are haunted by "only the new is exciting, interesting" filters and few are aware of those historic discussions which influenced today's *imaginaire,* our very imaginations of how we talk about and model decentralization.

We wish to develop and help develop messaging libraries which bring agency to community operators to run their own communication infrastructures in a secure manner. It is a vision which goes beyond a mere focus on algorithmic decentralization, striving for social meaning. We meet and befriend developers from promising decentralizing projects who share visions and practices of empowering users and communities. This allows us to develop common language, perspectives and agreements on partial goals, and lastly, to interact at the level of software libraries and services that we co-create. We consider such wider involvements crucial because there are already hundreds of experimental and a few advanced decentralization programs which, however, are often "island" solutions, not interoperating with each other and duplicating efforts, repeating mistakes and not learning from each other. Mapping community landscapes is a major focus of the D3.1 NEXTLEAP report.

Supporting communities and organizations to run their own messaging infrastructure is as much a matter of well researched crypto-protocols as it is of social operations and considering convenience of usage. How easy is it to set up and maintain secure infrastructure? If more intelligence and agency moves to the software end points how can users deal with damage or loss of their devices? How can trusted delegates of communities customize, influence and drive development of their infrastructure? During the course of this report we present more details on current thoughts and approaches related to these questions.

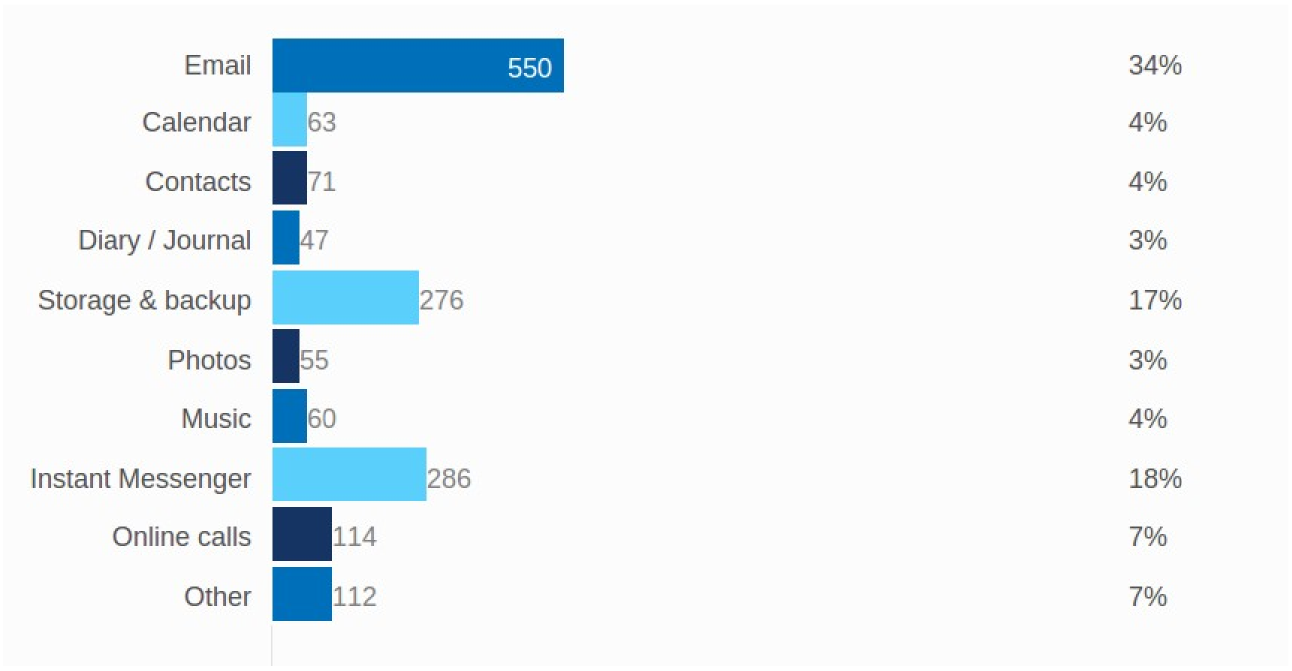
1.3 Focusing on Secure Email operations

In terms of achieving impact, we intend to prioritize our research and open source work towards the email, high-latency messaging space because email represents both the largest federated social network and probably the least privacy preserving communication platform today. Even Facebook's Whatsapp messenger has announced integration of transparent end-to-end encryption. The privacy situation with high-latency email messaging and end-to-end encryption is rather bleak in comparison. In fact, many messaging projects start out by citing the bad privacy situation around email as a reason to invent a new system. Take, for example, the introduction of the BitMessage[1] white paper[2], a new anonymous messaging system using blockchain technology:

> *"Email is ubiquitous but not secure. The ability to send encrypted messages is necessary but current solutions are too difficult for people to use: users must exchange both an email address and an encryption key through a trusted channel (like in person or by phone). Even users who do know how to use tools like PGP/GPG usually do not put forth the effort to do so unless they are particularly concerned about the message content. Novice users have a difficult time learning how to use the software because the relationship between public and private key pairs, and their uses, are foreign concepts..."*

We mostly agree on the stated problems but disagree on the subsequent strategy to replace email with a whole new system, breaking compatibility with the existing system. Encryption tools with PGP keys are already well established through the GnuPG[3] project, but are not used on a massive scale because of user-side complexities of dealing with keys and trust. We join others in wanting to improve, in a backward compatible way, the email situation and to realize by-default end-to-end encryption for users.

An informal poll within the redecentralize[4] community among ~1500 people on the question "Which app would you love to see decentralised (private, resilient, innovative)?" resulted in "email" taking the clear lead with more than double the votes received for the next runners up "Storage and Backup" and "Instant messenger".



1 http://bitmessage.org
2 https://bitmessage.org/bitmessage.pdf
3 https://gnupg.org/
4 https://redecentralize.org

In fact, many have tried to develop “the email successor” and email has been declared dead multiple times, but it obviously refuses to die. Instead, it is a popular “identity backbone” as most web services and platforms ultimately anchor identification to email addresses. It is high time to undertake pragmatic steps towards introducing end-to-end security and re-decentralizing email deployments and we intend our open source work to help it.

1.4 Email providers are key in the leap to more secure communications

The network of federated email providers form the largest open social identity system worldwide. The mobile phone network has more users but it is far from open: even resourceful entities cannot easily set-up their own infrastructure and create new mobile numbers. Setting up email infrastructure to let others create their own addresses can be done comparatively easily and with existing Open Source software.

We suggest that email providers provide common APIs for associating key material with email addresses, in an accountable manner. This would put automated end-to-end encryption for email within easy reach because programs could make use of these APIs without requiring user interaction. Moreover, if you consider that “key material” could also be addresses into other realms, we could have email addresses bootstrap into other decentralized privacy-preserving systems. Take, for example, Zcash[5] which is a privacy-preserving cryptocurrency enabling the transfer of value between Zcash addresses while only revealing metadata of the transaction to participant-selected third parties. If programs could automatically look up Zcash addresses for an email address, we would leap from email to privacy-preserving, decentralized payment systems.

Quite obviously, this puts email providers into a powerful and thus vulnerable position. **When we say “in an accountable manner” we mean to algorithmically, cryptographically bind providers in such a way that they cannot equivocate about associated key material.** They are, after all, the perfect man in the middle attacker for end-to-end encrypted email messages because they relay every single message by design. Removing the ability to lie, or making it very costly and identifiable via non-repudiable cryptographic evidence, means providers are less interesting targets for nazguls and thus more trustworthy towards their users.

We think that extending the open and federated email social ID system this way is technologically and practically feasible and we observe supporting conversations and research in various circles. We’d love to see it happen and help it through our open source efforts in the NEXTLEAP project.

5 https://z.cash

1.5 Initial Involvements with developers and communities

In order to get in contact with a wide variety of developers and users in currently thriving “decentralization” projects we recently participated in the following events and conferences which we briefly highlight here and which also motivated our use-cases later in this report.

- **32C3 Conference, Dec 27th-30th 2015 in Hamburg**: The yearly meetup of more than 12,000 people from the "Chaos Computer Club" and many other programming, decentralizing, culture, art, research and philosophy contexts. Holger Krekel attended the event and gave a talk on Hacking EU Funding for a decentralizing project[6]. Holger also joined sessions on setting up secure email providers (LEAP Project) and on the related PIXELATED web user interface, establishing contacts with respective developers which we describe later on in detail.
- **Internet Freedom Festival (IFF), Feb 29th - March 6th 2016 in Valencia:**: We went to this annual convention of activists and programmers (formerly known as "Circumvention Tech Festival") and led a workshop on how to fight vandalism in Wikis and other collaborative contexts. A particularly interesting discussion came about with Debian developer and cryptographer Daniel Kahn Gillmor on how a provider/server can prove that a client initiated a key change to deflect false “I didn’t submit this key” accusations. The related research questions are currently being further discussed. The IFF also allowed us to get a better picture of end-user use cases. It hosted communities from all over the world enthused by privacy, anonymity and decentralization. It included activists, researchers, trainers, journalists and social scientists. This diversity made it very clear that the use cases depend heavily on the community in question. For example, the Dalit from India can be identified as 'lower caste' based on their last name. Facebook’s real name policy threatens them since Hindu nationalists use it to harass them. This, along with many other examples, shows that use cases for anonymity and privacy are hard to generalize and are always tied to particular contexts. Maybe anonymity and privacy are most relevant when it comes to protecting what makes us stand out.
- **MirageOS Spring 2016 Hackathon, March 11th- 16th 2016 in Marrakech**: We attended the hackathon and investigated how this minimal operating systems (MirageOS) can be used to make the privacy-

6 https://media.ccc.de/v/32c3-7300-hacking_eu_funding_for_a_decentralizing_foss_project

preserving distributed presence system (DP5) more robust. The minimal trusted computing base of MirageOS and its implementation in a functional programming style make it well suited for high security applications and formal proofs. The security gains of DP5 are partly based on distributing data between servers hosted by different entities. Only if all servers conspire against the users can they learn the users' social graphs. Adding a new server implementation would harden this architecture against attacks on the current trusted computing base. What seemed like a good task for a one week hackathon turned out to be more challenging because the private information retrieval library that DP5 builds upon is implemented in C++ and MirageOS has no C++ runtime. What is mostly needed for DP5 to become useful is a client integration according to discussions with Ian Goldberg at the Internet Freedom Festival.

- **Squatconf and "decentralizing sessions", April 29th - May 3rd 2016 in Berlin**: Holger Krekel attended and co-organized sessions focused on current promising decentralizing technologies, among them the OpenSource WebTorrent[7] project which combines WebRTC and Bittorrent-like protocols and ZeroTier[8] which brings peer-to-peer end-to-end encrypted communications to the networking layer of the internet. With the two respective lead developers, Feross Aboukhadijeh and Adam Ierymenko, discussions took place around the notion of "*community defined networking*", where friends and groups operate their own communication and application infrastructures and where the social relation topology maps down to the networking layer: the devices of the involved community members constitute the network topology where the role of "intermediaries" is minimized or only used for bootstrapping. Another evolving conversation took place with users and developers of the Patchwork/SSB community which currently runs and communicates through a semi-stealthy "decentralized twitter" platform with a couple of hundred users empowering end-to-end encrypted decentralized communications between individuals or (temporary) groups of people. Patchwork[9] and its SSB protocol[10] is a well developed example of current "synchronization" architectures, as opposed to online client-server architectures. Synchronization architectures allow for high latency networking through efficient and cryptographically secured replication mechanisms. Besides, with Patchwork, the social topology also determines the network topology.
- **LEAP/Pixelated developer meeting, April 21st till early May 2016 in Sao Paulo, Brazil**: Early in 2016 we began to contribute directly to the LEAP platform, in order to gather ideas on how future NEXTLEAP protocols could be made to work within the LEAP Open Source project. It is one of our main integration targets described in more detail in the corresponding section below. Pixelated is a webmail platform built on top of LEAP that enables PGP encryption with automatic key lookup. We joined

7 https://webtorrent.io/
8 https://zerotier.com/
9 https://github.com/ssbc/patchwork
10 https://scuttlebot.io/

the meetup to discuss in-depth with developers about current and planned key management protocols and discuss challenges and tasks. In addition, the overall architecture of the LEAP platform and bitmask client were reviewed for security issues. Discussions also took place with South American email providers and activists who are looking at using LEAP and automatic key management. These discussions will continue and will inform our prototyping and coding activities. After the main meeting, We spent another week with the developers of Bitmask - the LEAP client. They discussed API changes needed to improve security aspects identified during the meeting. We are planning further in-person meetings to follow-up with LEAP and Pixelated developers.

- **CAPS community meeting, May 18th 2016 in Berlin**: Holger Krekel attended this gathering of CAPS-related EU projects and interested parties. He gave a quick impromptu overview on aspects of NEXTLEAP and subsequently enjoyed good discussions with Stravroula Maglavera from the MAZI[11] project (offline communication infrastructures for physical communities) and with Renato Lo Cigno from the NETCOMMONS[12] project (network infrastructure as commons). We gathered interest in arranging another meetup to exchange research, development and community insights and possibly also tools.

Participating in such wider discussions and developments of decentralized and privacy-preserving communications and architectures will remain part of our activities over the next two years. This allows us to embed our open source work on identity and messaging protocols within wider contexts, to better determine use cases and to inform our API design and algorithms. Similar to how academic worlds are “disciplined”, most programmers also stay and work within a certain community, often tied to programming languages and particular projects. There is no single go-to developer conference on “decentralization”, however -- and maybe this is a good thing -- it means we expose ourselves to all kinds of perspectives and projects when trying to talk about the holy grail of decentralization.

2 Integration communities and activities

We are seeking integration into and usage of our protocol’s open source code from diverse communities and other open source efforts. This is both for validation purposes and because we want to contribute to real-life changes for users and communities. Naturally, we cannot be totally sure which integration efforts will come to fruition in due time. Open source collaborations are hard to predict over the course of years due to social dynamics and technical complexities. Finding communities to collaborate with is thus an ongoing task. In

11 http://www.mazizone.eu/
12 http://netcommons.eu/

any case, we aim to provide high-quality reference implementations of NEXTLEAP protocols and motivate and help others to port them to other languages and use them in their projects.

2.1 Email providers and operators

At the core of the email end-to-end encryption problem lies the “federated identity” question of how to associate email addresses with key material in a secure and privacy-preserving manner. For new secure messaging protocols to have impact in the email space, the basic problems of secure key registration, lookup and verification must be tackled first. We work from the pragmatic conviction that extending the role of already federated email providers to offer secure and privacy-preserving key lookup services is crucial for making progress towards massively deployed end-to-end email encryption.

Our current priorities therefore are sketching API *stubs* for key management, “federated identity” protocols and discussing with email providers and server operators early on. We are seeking contact with diverse email providing organizations such as gmail.com, web.de and the more privacy-oriented mailbox.org and posteo.de, in order to minimize the risks of just thinking up “island” solutions for single platforms which are ignored or rejected by the others. Most of these providers currently pursue their own systems for implementing end-to-end encryption, which makes it even more important that we relate and integrate our activities within respective efforts of others, not all of which are public at this point.

2.2 LEAP & Pixelated

Our current primary integration community is LEAP[13], which focuses on providing secure communication platforms, with a particular focus on email. LEAP aims to provide the means to semi-automatically set up email services which provide automatic key management and encrypted backups of user data. It also provides client-side software to ensure that email is only seen in cleartext on an end-user's computer. Through encrypted backups to a provider, a user can recover loss or damage of a device. Recoverability is a core feature of cloud platforms such as Gmail and necessary to achieve wider adoption these days.

LEAP’s scope touches many aspects of our work. Right now they are focussing on Federated Identity (key management) and asynchronous secure messaging (email). LEAP is somewhat unique in the space, as they are trying to develop tools for setting up providers as well as end user clients. This allows us to integrate our open source modules into both client and provider site programs and thus to validate our approaches. Later on, LEAP developers also plan to tackle the “synchronous” messaging space, aka “chats”, which may fit well with our planned implementations of secure messaging protocols.

13 https://leap.se/

Currently there are half a dozen email providers either running LEAP or in the process of setting it up. In particular we met people from Calyx[14], Riseup[15] and Codigosur[16] at the Internet Freedom Festival (IFF, see above). Most existing deployments currently also route general internet traffic through a virtual private network (VPN). Email services are about to be deployed in the coming year. This gives us the opportunity to integrate our prototypes into systems in use without having to acquire large user bases ourselves.

Moreover, we aim to work with the Pixelated[17] project which implements a modern web interface and itself collaborates and integrates with LEAP in order to provide transparent encryption in the email space.

One issue we see with the LEAP code base is that it currently follows a rather “monolithic” design and its basic key management code cannot be reused easily. We thus aim to factor out the identity and messaging protocol relevant parts into a NEXTLEAP repository which then holds small reusable modules which we can further develop, review and release also for integration in other projects. We later aim to build upon key management to provide a prototype for perfect forward secrecy for async messaging - likely also to be used within the LEAP context.

2.3 Mailman, Schleuder, Sovereign

Mailman[18] is a software system written in Python which helps people to setup and operate mailing lists for groups of people. It is widely used software and has recently seen a code modernization by long-term maintainer Barry Warsaw and his colleagues. We started discussions with Barry, who has indicated his interest in integrating PGP key management. While there have been attempts and prototypes through Summer of Code projects, mailman still does not offer encrypted mailing lists. A major obstacle lies, as with many other email-based software, in the unsatisfying situation around obtaining and verifying encryption keys. It is thus a good means of validation to see if our protocol code can remedy the key management issues in a real life product which is used by millions of people. We would like to eventually have the NEXTLEAP project mailing list use such encryption extensions, in true “dogfooding” spirit.

We also plan to communicate with the Schleuder[19] project which already implements encryption for group email. With Schleuder, a sender encrypts towards the intermediary who unpacks the message and sends it encrypted to all members. The intermediary machine knows about all encryption keys of all members. While Schleuder already implements the basic machinery, it is still experimental and not widely used.

14 https://calyx.net/
15 https://riseup.net/
16 http://www.codigosur.org/
17 https://github.com/pixelated/pixelated-user-agent
18 http://list.org/
19 https://schleuder2.nadir.org/

Sovereign[20] is a collaborative "devops" effort which provides automated setups of best-practice email servers and groupware such as OwnCloud and OpenVPN and it meant as a fully functioning setup for small communities. Trying to fit our key management facilities into such projects will be useful for gathering feedback and comments.

2.4 Synchronous/low-latency messaging

Contrary to the high-latency email space described above, there are many projects in the low-latency, synchronous messaging space. The prospective innovations to be contributed by NEXTLEAP's secure messaging protocol work are not yet sketched out in detail enough to allow us to name with confidence the appropriate integration communities.

However, when email providers begin offering APIs for key and identity management, these can probably be used to bootstrap other decentralized low-latency/real-time chat mechanisms. Using email providers as backbones for key and potentially other "contact" data opens up the possibility for an upgrade path from email/async messaging to synchronous messaging. While using the email federated identity system may sound like an anti-pattern to achieve anonymity, it actually depends on a) how anonymously one can use an email address and b) how easy it is to set up temporary email addresses with associated key material.

As we are already engaged with the LEAP project, which may in turn go for low-latency chat and resource sharing mechanisms, we will certainly consider integrating messaging protocols with them as well.

In the same vein, we will also continue discussions with Dominic Tarr and his fellow developers of the Secure Scuttlebutt[21] protocol, a secure Peer-to-Peer Logstore used in the decentralized Twitter platform Patchwork[22], which is interesting in that it is driven by cross-disciplinary discussions involving anthropology, sociology and philosophy topics. It uses cryptographically secured replication where "following" others means replicating data from their computers.

Within NEXTLEAP we will also look into collaborating with Nadim Kobeissi (INRIA) on CryptoCat,[23] a promising decentralized web-based project for sharing messages and resources in a cryptographically secure way.

In order to intensify collaboration with the formal proof parts of the NEXTLEAP proposal we will also consider the MirageOS and OCAML communities for integration. In particular, Jackline - a minimalistic, security focussed XMPP client - is interesting. Jackline is based on clean-slate implementations of widely spread protocols and standards. Combined with the OCaml language and the functional programming style, it provides a good environment for formal proofs of new protocols, such as a DP5 client integration.

20 https://github.com/sovereign/sovereign/
21 https://scuttlebot.io/
22 https://github.com/ssbc/patchwork
23 https://github.com/cryptocat/cryptocat

2.5 Mazi, NetCommons, Panoramix

As to integrating with other EU research projects, we are looking forward to more discussions with Stravroula Maglavera from MAZI (Greek for “together”) which researches offline, community-operated communication infrastructure in physical neighborhoods. We have also entered discussions with Renato Lo Cigno from the NETCOMMONS EU project which includes GUIFI[24], a large mesh network operator in Spain aiming to empower communities to operate their own communication infrastructure. Moreover, we intend to have a joint meeting with participants of the EU project Panoramix[25] which deals with building a general-purpose anonymity network for messaging and email in Europe.

24 https://guifi.net/en
25 https://panoramix-project.eu/

3 Use cases

3.1 Email / Federated identity

We will describe and discuss here core initial use cases related to federated identity protocols, using the simplified terminology of **mail programs** and **providers** which collaborate to realize email communication today. In email standards one will otherwise find vocabulary such as MUAs (mail user agent) and MTA/MDAs (mail transport agents, mail delivery agents) or mail servers. So let us begin with the basic key management use cases:

> **Transparent Key Registration**: As a user, I want my mail program to automatically generate and register encryption key material to the mail provider that receives and delivers email to me.
>
> **Transparent Key Lookup**: As a user, when sending a message, I want my mail program to automatically lookup key material for each destination address of my message. When receiving messages, I want my mail program to automatically lookup key material so as to verify message signatures from the email provider which serves the sender's email address.

Simple as it may seem, none of the major email providers today provide these two basic functions, instead delegating the implementation and handling of keys on the email programs. In that model, users are to manage keys themselves (export, import, verify). It is a model that even programmers who understand the math and tools behind public key cryptography struggle with, as a lot of recent research shows. By comparison, a provider can offer registration of a public key through established authenticated interfaces (https+login, 2FA) and it is natural to be able to ask a provider then to provide this key for a given email address.

3.1.1 Techno-social provider accountability

If email providers mediate public keys between users then there is a big issue of trust as they also mediate all mail to a user. With just the above use cases, an email provider can trivially launch a "man in the middle" attack, i.e. give a false key out, decrypt mail to that key itself and then re-encrypt with the original user key. By doing this the provider could obtain all mail in clear text without the user knowing. This could be countered by users verifying each other's keys through out-of-band means: e.g. a mail program could show an OCR code which can be scanned by another mail program instance to verify the association of email address and key material. However, for key management to remain more transparent, we want providers to be more directly accountable:

> **Provider Accountability:** As a user, after I have my mail program generate and register my key material, I want certainty that my own email provider correctly serves the key material to everybody else and that I can hold him accountable.

Apart from relieving the user from the need to semi-manually verify keys, provider accountability allows for frequent changes of key material and thus is probably fundamental for providing Perfect Forward Secrecy in the email space. When we say **“**I want certainty**”** there are two interesting interpretations:

> **Social provider accountability:** As a user, I might get certainty by knowing who operates the provider’s platform and in which way. Is it run by very trustworthy people? Do I see the operators having funding, the means to operate their business without corruption and interventions from third parties? Are there ways to penalize the provider if it cheats?

> **Technical provider accountability:** As a user, I want my mail program to obtain cryptographically irrefutable evidence that my provider does not lie with respect to presenting my key material to others. Are the algorithms secure in this respect? Is the implementation of the algorithms correct?

These two interpretations in conjunction are core to our techno-socially informed approach in bringing end-to-end encryption for socially federated communication platforms. This means that some idealizations like “we can demand that clients and servers all change their software in a particular way” cannot be left unquestioned. In Web/Google/Facebook/Signal cases this might be acceptable, but not in the federated email landscape.

3.1.2 Dealing with loss of devices and key material

Further, to relieve users from having devices which can safely store private key material for long times and to keep key material in sync between multiple devices, we sketch another use case which targets data availability and integrity in the event of device damage, theft or accidental erasure.

> **Key availability and backup:** As a user, I want my local device to synchronize key material to the respective provider (or third parties) in an encrypted manner so that only the user can unlock/decrypt it from another mail program instance.

This key availability use case is not unconditionally necessary if e.g. an end-user already uses a device or setup which performs encrypted backups e.g. to cloud services like Tarsnap.[26]

3.1.3 Community Agency

We also consider it crucial to state the use case relating to “community agency” and operating infrastructures which we discuss in several places in this report:

> **Community run infrastructure:** As community-delegated sysadmins, we want to be able to setup and maintain best-practice email servers in an automated manner using secure configurations and defaults.

Making it easier to maintain secure email servers gives communities and organizations agency over their communication infrastructure and contributes to better federation and sustained social federations of email systems.

26 https://www.tarsnap.com/index.html

3.2 Secure Messaging

Based on basic key management and federated identity use cases we also describe a few use cases related to the actual messaging part.

> **Privacy-preserving message sending:** As a user, I want my mail program to send a message using verifiable encryption keys for the destination(s) without asking me to confirm trust in the keys. I also want my mail program to sign the message using my private local key material.
>
> **Message authentication:** As a mail program, I want to verify that a received message is signed from the original sender by asking the respective email provider for a verification key without asking my user to confirm trust in it.

A challenging use case of secure messaging for high-latency communications relates to perfect forward secrecy:

> **Forward Secrecy:** As a user, in the case of theft of my private key material, I want most of my past mail conversations to remain unreadable to others even if the encrypted data was recorded by intermediaries (e.g. the mail provider which sees all messages).

4 Early Protocol Considerations

4.1 Federated identity / Provider accountability

Core research and implementation issues relate to techno-social provider responsibility when transparently managing end-to-end encryption for email users.

> **Social:** How can providers monitor each other's handling of user's key material such that they can raise a red flag for corrupt providers? How can this be done in a privacy-preserving manner? How can the end-user get certainty on this process?
>
> **Technical:** How can users have the programs on their devices mathematically verify that their provider serves the correct key material?

Note that the act of signing, encrypting and decrypting messages can be done using the well-established GnuPG project which is already integrated into many mail programs. While it comes with a "web-of-trust" model for verifying keys among end-users we think it's safe to say that it won't get to mass-deployment. We rather focus on email providers managing the keys for their users but we can consider this a complementary process: once end-user mail programs can easily obtain somewhat trustworthy public keys for their contacts they can check if the keys are correct through out-of-band means or web of trust tools. This remains important for people in repressive situations.

One of the current proposals for allowing users to technically verify that their provider serves the correct public key material is discussed in the CONIKS[27] paper and ongoing related research considerations as stated by CONIKS author Marcela Melara. In her March 2016 post "why-making-johnnys-key-management-transparent-is-so-challenging"[28] she highlights several practical issues of CONIKS, even though it is currently being worked on in an unfinished open-source code-base by Google and Yahoo! for their own end-to-end e-mail encryption projects.[29] Moreover, it is unclear how suitable the current CONIKS model is for the email situation. CONIKS works from the perspective of having uniform control over the software on both the end-user system and the provider. This is to some extent a natural assumption for Web software where the operator of a platform also determines the software at the end point. And it is also natural for organizations like Google and Facebook which control both their platforms and the end-user apps which interact with their platforms. Email messaging, however, happens through socially federated operations of email platforms and a multitude of mail programs that interact with the platforms through specified and standardized protocols. The proposition to change software on both the end-user and the provider side is much more problematic compared to the "web" case.

We are thus interested in protocols and processes which achieve provider accountability with only minimal change and very small operational requirements on end-user mail programs. From this perspective, asking from providers activities like periodically publishing information or auditing other providers' information is less of a problem compared to asking email programs to do the same, as providers already operate and monitor services which include such periodical activity.

We are nevertheless also interested in advancing CONIKS itself, this time working from the assumption that it is acceptable to require a lot of changes on the end-user device. At the IFF we met with Marios Isaakidis (NEXTLEAP, UCL) and Daniel Kahn Gillmor (ACLU) to discuss federated identity approaches. One part of the discussion focussed on the CONIKS protocol and problems with key updates: In default mode identity providers will accept key change requests from users after authentication. With commonly deployed authentication systems there is no way for the provider to prove that it authenticated the key change properly. So in case of conflict it remains uncertain whether the identity provider or the user changed the key. This is a risky situation for both parties, and makes the service provider vulnerable to reputation damage. Any user could change their key and claim the provider did it without their consent. Users on the other hand would not be able to check if the fault was on their end.

We followed up on the discussion with a proposal expanding CONIKS and

27 https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-melara.pdf
28 https://freedom-to-tinker.com/blog/masomel/why-making-johnnys-key-management-transparent-is-so-challenging/

29https://github.com/yahoo/coname

integrating it with secure remote password (SRP) which LEAP is already using. The CONIKS team later moved in a similar direction with CONIKS 2.0 so this seems like a viable solution for the future. We also introduced this solution at a joined meeting of LEAP and PIXELATED in Brazil. Both teams agreed that this seems to be a good option.

4.2 Secure Messaging

The Signal/Axolotl protocol offers perfect forward secure messaging and a number of other interesting properties. Early discussions happened between NEXTLEAP’s Harry Halpin and Moxie Marlinspike, the author of the Signal app and the protocol which is now used by Whatsapp on a massive scale. We plan to engage in a prospective joint workshop and consider integration activities from there.

It remains to be seen how the protocol can be used in the context of a federated system. Adam Langley’s Pond, which combined a similar protocol with federation and resistance to traffic and social graph analysis, is in stasis according to the author himself.

Another promising protocol to reduce metadata leakage in particular for synchronous messaging is DP5. It allows organizations to provide a service offering presence information (and associated data) to their users, while using strong cryptographic means to prevent the organization itself from learning private information about its users, such as their lists of friends. One of the authors - George Danezis (UCL) - is part of NEXTLEAP, and we has been discussing implementation ideas with Ian Goldberg around the Internet Freedom Festival and the MirageOS hackathon.

5 Development and validation guidelines

Our focus is to contribute to the development of the communities we work with, their surrounding ecosystems and to give feedback to the academic communities in the relevant fields. We ensure continuous contributions by engaging in the respective development processes. This enables us to contribute small pieces at a time and blend into the ongoing process. We therefore avoid long discussions and merge conflicts at the end of an isolated development effort.

We aim to get to know personally and work with core developers of integration communities because it provides a better basis for collaboration and getting changes merged. Typically, such meetings can happen at conferences or sprints which take place at or around conferences. We gave information on our outreach earlier on which also served to describe the use cases that we are basing our prototyping and protocol work on.

5.1 Our approach to open source development in 2016

From previous merlinux experiences, e.g. with the EU PyPy project 2004-2007, we know that having a rhythm of personally meeting up for collaboration on multi-day sessions or sprints every few months helps tremendously in developing shared perspectives and trust. While the technicalities of how to create a merge/pull request support such collaborative group development processes, they are a tool, and not themselves the main reason why a project succeeds. This view ties back to our earlier stated view of regarding software developments as techno-social developments, i.e. the conjunction of social, personal and technical coding collaborations. Moreover, in 2016 there are millions of open source projects and it is not as easy as it was in 2000 to gain attention for any new effort without having "community clout", i.e. being known to people in the relevant fields. Forming personal relations with researchers, other developers, operators and users of the envisioned infrastructures and protocols helps to realize the impact of the work that we are aiming for within our NEXTLEAP efforts.

5.2 NEXTLEAP repositories and development

We are to use the NEXTLEAP organisation on GitHub for creating code repositories and our development processes around it and we will generally follow these development guidelines:

- Use the https://github.com/nextleap-project organisation to host our repositories implementing key management protocols and prototypes.
- Help other projects to integrate our prototypes by attending their sessions or inviting them to attend sprints and sessions that we organize.
- Create repositories as needed for prototyping and documenting protocols for key management and messaging (WP2) for use in integration communities.
- Offer/give committers from integration projects commit access to our coding repositories in order to grow a community with shared interests.
- Use a standard peer-review process where one person writes a Merge Request / Pull Request and someone else reviews and finally accepts it.
- Use MIT licensing wherever possible for publishing our general purpose library code so it can be used by a wide variety of projects with different licenses, although GPL 3.0/AGPL will be used for other components where inherited (LEAP, Signal, etc.)
- Otherwise follow the licensing, contribution and copyright policies of integration projects to ensure the easy integration of our contributions.
- Attend developer and community meetings regarding secure messaging and email, to help guide the prototyping and increase the likelihood of integration of our research-driven modules into actual projects.

- Use continuous integration tools commonly used by the communities we work with.

As we are otherwise not directly involved in offering end-user facing services, we will instead seek feedback from integration project developers and their respective users.

5.3 Contributing to LEAP

As our primary integration community is LEAP we describe its processes here in more detail. We also aim to collaborate with Ksenia Ermoshina and Francesca Musiani (CNRS Paris) who, from Month 6 to 24 of the project, will be doing fieldwork interviews and in-depth case studies within the NEXTLEAP project on three selected communities, one of which will most likely be LEAP.

LEAP hosts its core repositories itself at the leap.se host and mirrors these to the https://github.com/nextleap-project/[30] site. In total LEAP has 82 repositories at GitHub and 108 repositories on more internal sites. For continuous integration the Ruby-centric web application and some similar projects use travis-ci. Gitlab also provides continuous integration, so the platform and some of its components are built on Gitlab. For the Python parts LEAP uses http://buildbot.net/ to run tests and build bundles; for the platform parts https://jenkins.io/ is used for testing and building debian packages.

The distributed team has four meetings every week - each with a specific focus:

- Dev scoping meeting
- Sysdev meeting
- Meeting with Pixelated (web application development)
- General meeting or hack session

Communication channels are quite dispersed and include:

- Crabgrass: we.riseup.net/leap, groupware for organizational knowledge and internal archives, private wiki pages.
- Redmine: leap.se/code, issue tracker and public wikis.
- Etherpad: real-time collaborative text editor for meetings.
- Mailing lists: discuss@leap.se is the public mailing list.
- Mumble: for conference calls
- Calendar displays for group meetings, events and individual work hours.
- leap.se is the external facing site
- developers@leap.se and sysdev@leap.se

The development process is organized in three-month cycles with a specific focus. Each week starts with a scoping meeting to define a narrow focus for the given week. Due to the distributed development approach most code is developed by one person and then reviewed by another using merge requests. Quality assurance happens both through continuous integration and by hand following QA scripts before the platform and Bitmask releases.

30 https://github.com/nextleap-project/