



NeXt generation Techno-social Legal Encryption Access and Privacy nextleap.eu
Grant No. 688722 Project Started 2016-01-01. Duration 36 months

DELIVERABLE D5.2

Releasing Initial Prototypes

Holger Krekel (merlinux)

Azul (merlinux)

Beneficiaries:

merlinux, IMDEA, UCL

Internal Reviewers:

K. Ermoshina (CNRS), C. Troncoso (IMDEA), H. Halpin (INRIA)

External Reviewers:

Phillip Rogaway (University of California, Davis, USA),
Noah Swartz (Electronic Frontier Foundation, NY, USA),
maxigas (Universitat Oberta de Catalunya, Internet
Interdisciplinary Institute),
Björn Petersen (Delta-Chat)

Description:

The initial prototypes for protocols for preserving identity and privacy have been iteratively released to the public, conceived and developed through intense community involvements and collaboration.

Version:

1.0

Nature:

Report (R)

Dissemination level:

Public (P)

Pages:

Date:

2017-06-30

1. Overview and Introduction	4
1.1. Usable security for tools with proven utility	4
1.2. Incrementally augmenting federated E-Mail infrastructure	6
1.3. Quick introductions to Autocrypt and ClaimChains	6
2. Involvements in E-Mail encryption developments	8
2.1. PGPSummit 2016: questioning provider mediated keys	8
2.2. LEAP hackathon: Spam and e2e mail and the move towards in-band protocols	8
2.3. OpenPGPConf: towards in-band key transmission	9
2.4. NEXTLEAP research meeting: in-band federated identity	9
2.5. AME2016: automatic mail encryption hackathon	10
2.6. 33c3 Panel on mail encryption “we fix the internet”	10
2.7. Webmail e2e: Mailvelope and Roundcube meetings	10
2.8. Internet Freedom Festival 2017: more Autocrypt momentum	11
2.9. Transnational Secure Messaging Meeting (TSM)	11
2.10. EasterHegg 2017: E-Mail encryption and Autocrypt session	12
2.11. Informal Autocrypt / NEXTLEAP meetup: avoid passphrases	13
2.12. NEXTLEAP Launch meeting	13
2.13. Autocrypt level 1 meetup: finalizing the spec ...	13
2.14. A note on community involvements and the “Project Stakeholder Committee”	14
3. Prototypes and releases	15
3.1. Updated “use cases” that underly our prototyping efforts	15
3.1.1. Messaging use cases	15
3.1.2. Accountability use cases	16
3.1.3. Community agency	16
3.2. Autocrypt docs/specs for federated E-Mail e2e encryption	16
3.2.1. Introducing Autocrypt: E-Mail Encryption for Everyone	17

3.2.2. The social Autocrypt approach	17
3.2.3. The technical Autocrypt approach	17
3.3. Usability prototypes	18
3.3.1. Plain UI prototype for configuration settings	19
3.3.2. Message exchange prototype in Rails	20
3.3.3. User interface prototype as single page application	22
3.3.4. User interaction test prototype	23
3.4. Python command line tool for prototyping, integrating e2e encrypted mail	24
3.4.1. Command line tool	24
3.4.2. Claim Chain prototyping	25
3.4.3. Online bot answering mails with Autocrypt headers	26
3.5. Privacy-preserving ClaimChain prototyping	26
3.6. Prototyping with mailman integration community	26
3.7. Ongoing collaboration with the LEAP integration community	27
3.7.1. Nickserver: key management for federated LEAP e2e platform	27
3.7.2. Omniauth-sso: secure login to federated LEAP e2e platform	27
3.7.3. Leap_web: password recovery within the federated LEAP e2e platform	28
4. Outlook on period 2 activities	29

1. Overview and Introduction

In this report, we describe our efforts which aim to support implementing user-friendly, privacy-preserving and federated encrypted messaging.

All of our prototypes were developed in an agile incremental manner, discussed at several conferences, and at NEXTLEAP internal and user feedback sessions that we describe in Section 2. The chronological list of events shows how our “provider mediated key” approach described in D5.1 evolved into the “in-band” approach which culminated in a new community-driven effort called “Autocrypt” for securing E-Mail communication in December 2016. The new approach does not require changes from providers but only from mail clients. This minimizes the number of players needed to significantly improve privacy.

Section 3 then describes updated use case descriptions and the main prototyping areas and releases during reporting period 1. Our most impactful socio-technical effort to date has been to help create and energize [the Autocrypt effort \(section 1.3 and 3.2\)](#) which incrementally augments the federated E-Mail identity system with end-to-end encryption protocols. The evolving Autocrypt specification is being implemented in several mail clients during Summer 2017, mostly by the developers of these clients themselves. Already in June 2017, a new promising third-party [Delta-Chat messenger app](#) was released which makes use of Autocrypt for end-to-end encryption. Section 3 also describes our prototyping efforts around ClaimChains (see D2.2, D4.2), UI-testing, password-recovery and single-sign-on use cases for E-Mail providers.

We conclude in Section 4 with an outlook showing our continued strong focus on improving the E-Mail encryption space. This outlook is based on continued NEXTLEAP funded research collaboration and prior discussions with many external collaborators and stakeholders on how to pragmatically and fundamentally enhance the worldwide E-Mail ecosystem.

In the remainder of this introduction we discuss recent interdisciplinary usability research from NEXTLEAP partners and how it relates to our own experiences during our community and prototyping activities of period 1. We also highlight our approach for establishing real-world privacy-preserving federated identity systems through our “tools with proven utility”, the E-Mail infrastructure. We conclude with a quick introduction to Autocrypt and ClaimChains, core protocols and data structures we are co-evolving for our WP5 work.

1.1. Usable security for tools with proven utility

NEXTLEAP research partners have discussed the cross-influences between secure messaging implementations and user expectation in a recent paper for the Eurosec2017 conference¹. They arrived at this main thesis:

Developer-User Disconnect: We hypothesize that properties of protocols are not understood by users. The core of the problem is the methodology currently used in the developer community to design protocols, where developers of secure-messaging applications hypothesize what properties a protocol should have based on their beliefs about users.

This resonates with our experiences within period 1 of the NEXTLEAP project. As we interacted and collaborated with external stakeholders and open-source developers we repeatedly started from usability considerations to inform crypto design considerations. Moreover, we aimed our efforts to help evolve existing projects and apps rather than trying to invent complex new systems. In a recent user-study on [“Secure Messaging Obstacles”](#) researchers interviewed 60 users regarding their adoption of secure messaging tools and also reached this major conclusion:

“Secure tools with proven utility. *We encourage the security community to prioritize securing the communication tools that have already been adopted by mainstream users over improving the usability of different secure tools. The users’ goal to communicate with others overrides everything else, including security. Growing a user base for a new tool is difficult and unpredictable. Therefore, we encourage security researchers to work with today’s existing popular tools.”*

We consider it a common pitfall for both EU projects and personal side projects that they develop a new system for addressing a problem, without reasonable considerations for user adoption. Thus a lot of effort goes into something that is hardly used. This might have to do with the obsession of modernity with *new* technologies and *innovation*, whereas historians of technology have long agreed that *maintenance* and *repair* are just as important.²

Another cultural bias that constitutes a barrier for effectively addressing crypto challenges is sexism. Reproductive work, especially care work, have historically been feminised. Patriarchal social structures have marginalised and underappreciated feminine labour. Therefore, these roles seem less attractive to both funders and developers (who are still mainly men). They are also valued less in policy regimes and developer communities, not to speak of Silicon Valley’s “disruptive innovation” story selling. Our goal is to counter these dynamics through caring for perspectives from maintainers of existing tools and by aiming for incremental improvements

¹See here for the presentation: http://nextleap.eu/res/Ermoshina_EuroUSEC_final.pdf

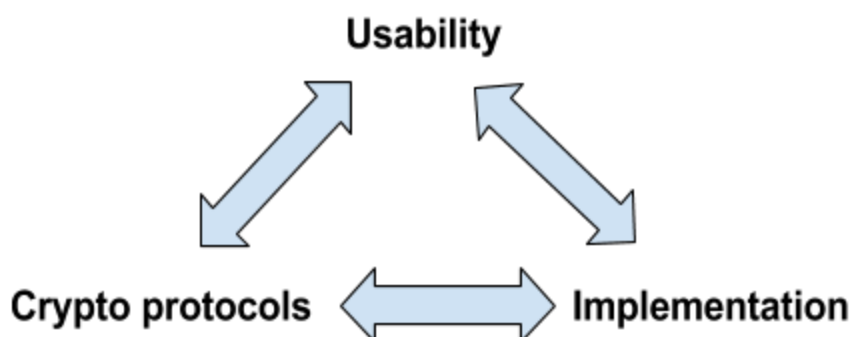
² See Edgerton, David. *The Shock of the Old: Technology and Global History Since 1900*. London: Profile Books. 2008.

rather than focusing on “disruptive new systems” which others are to recognize and implement. In fact, our community activities around the Autocrypt effort are part of our trying to counter the “it’s all about creating the latest and greatest mathematically proven features of a crypto-protocol” and rather start from pragmatic maintainer aspects and try to embed advanced cryptographic concepts based on usability considerations.

1.2. Incrementally augmenting federated E-Mail infrastructure

Our “tool with proven utility” is the existing federated E-Mail infrastructure. We state both in the D5.1 report’s introduction and similarly on the Autocrypt web page: *“E-Mail remains the largest open federated identity and messaging ecosystem, anchors the web, mobiles, and continues to relay sensitive information between people and organisations.”* We are happy that with our research-, community- and prototype-oriented activities we could in Period 1 help facilitate this Autocrypt effort which we consider a very promising attempt to bring pervasive end-to-end encryption and privacy into the federated E-Mail ecosystem.

For the relationships between usability, implementation and crypto-protocol design efforts we assume no clear-cut hierarchy. Rather each aspect influences the others as indicated by the following diagram:



We consider existing mail client architectures and their often under-resourced developments as constraining both protocol design and usability efforts. In particular, many mail clients are developed by individuals, sometimes in their spare time. The crypto-protocol efforts of WP2 and WP4, as well as our related WP5 implementation, efforts thus need to be weighed against the complexity they introduce for maintainers of E-Mail clients and infrastructure, our “tools with proven utility.”

1.3. Quick introductions to Autocrypt and ClaimChains

Autocrypt is an evolving effort and a specification³ for mail clients (often called MUAs in specification terminology) of how to manage and send cryptographic key material along outgoing mails and how to interpret headers when receiving E-Mails. The following “happy

³ See <https://autocrypt.org/en/latest/level1.html> for the full specification.

path” mail flow graphic is documented and further discussed on the web page⁴ and shows how Autocrypt headers are transferred “in-band”, i.e. they are transferred along mails that Alice and Bob send each other.



This visualization shows three mails -- the first from Alice is a cleartext mail and contains the Autocrypt header with Alice’s encryption key, the second mail from Bob to Alice is already encrypted and contains Bob’s encryption key. In the third mail, Alice writes an encrypted E-Mail to Bob because she received the encryption key in the second mail from Bob. Autocrypt aims to prevent unreadable mail for users, particularly in multi-device situations, and therefore requires Alice and Bob to both have their mail clients set “prefer-encrypt=mutual” before mails are encrypted by default.

ClaimChains, as described in the NEXTLEAP report D2.2, are append-only logs of “claims”. They are to be managed by mail clients and to make it costly or impossible for providers to fiddle with Autocrypt headers and cryptographic key material when they try to read encrypted mails through a so called Machine-in-the-middle attack. The Autocrypt Level 1 specification itself deliberately⁵ postpones the question of how providers can best be prevented from bad behaviour -- a topic rather considered for future Autocrypt levels and aligning with our ambitions to apply ClaimChain research. Claims can be statements about cryptographic keys and can also

⁴ See <https://autocrypt.org/en/latest/examples.html#happy-path-example-1-1-communication>

⁵ The autocrypt.org website says: “**Protect first against passive data-collecting adversaries**, resist the temptation to early-add complexity which aim to prevent active attacks. See [RFC7435 A New Perspective](#) for some motivation..”

reference claims from other append-only logs (then forming a graph) . Within WP5 we in fact use it for creating claims about own cryptographic material and about out-of-band verifications, see sections 3.4.2 and 3.5 of this report. In combination with the Autocrypt in-band approach WP5 focuses on the “in-band” ClaimChain variant which is also discussed and simulated in the D4.2 report.

2. Involvements in E-Mail encryption developments

Since our first report (D5.1, June 2016) a number of new developments and collaborations evolved, which we describe chronologically. We mostly focus on E-Mail encryption communities and players because that is where we see effective opportunities for real-world change with our coding and specification activities.

2.1. PGPSummit 2016: questioning provider mediated keys

During 8-10 July 2016, Holger and Azul participated in the PGPSummit where several key players in the E-Mail encryption and PGP space were present. Holger gave a talk about the NEXTLEAP contexts and the “provider mediated keys” model which we wrote about in our D5.1 report from just a month earlier. Several sessions took place over the 3-day meeting, among them one with [K-9](#) developer Vincent Breitmoser and [OpenKeychain](#) developer Dominik Schürmann. These, however, focused on “keyserver-less” key distribution: the decade old PGP keyserver model, along with cryptoparty practises, have known issues, including leaking the social graph of users and all the difficulties arising from involving users in key-management decisions (import, transfer, export, trust). We started exploring sending encryption keys along with outbound mails and wondered whether to put them in headers or as attachments. We later termed this approach “in-band” key transmission and still later as “Autocrypt”, see below.

2.2. LEAP hackathon: Spam and e2e mail and the move towards in-band protocols

During the [LEAP](#) hackathon, 10-13 July 2016, we involved ourselves with the LEAP E-Mail encryption project. We had good discussions with Daniel Kahn-Gilmore (dkg) from the [American Civil Liberties Union](#) who is also part of the [GnuPG](#) core team. Most notably, we discussed severe problems with public lookup keystores based also on the seminal post from Mike Hearn, who worked with GMail and the Google Anti-Abuse team for many years. He [wrote in 2014](#), mostly referring to the problem of SPAM mails in the encrypted E-Mail context:

When you look at what it's taken to win the spam war with cleartext, it's been a pretty incredible effort stretched over many years. "War" is a good analogy: there were two opposing sides and many interesting battles, skirmishes tactics and weapons. [...] Trying

to refight that in the encrypted context would be like trying to fight a regular war blindfolded and handcuffed. You'd be dead within minutes. So I think we need totally new approaches.

In our discussions with dkg we noted that in-band key transmission leaves the first mail unencrypted because with an initial mail there is no key we could encrypt to yet. The first mail being cleartext means it's detectable by current anti-spam measures. Only if a user replies will encryption keys be sent back. This contrasts with provider mediated or traditional key servers where encryption keys are easily available for a given E-Mail address. With the in-band approach, spammers can only get at sufficiently large numbers of encryption keys through compromising mail providers of which many are, however, their long standing enemies (see the "war" wording from Mike Hearn above). While sending keys in-band might not fully solve the "spam e2e mail" problem, our hypothesis is that it mitigates it to a significant degree. This tentative conclusion reinforced movements towards "in-band" key discovery mechanisms and away from public or provider operated keyservers.

Other topics at the LEAP Hackathon revolved around Soledad, a core component to synchronize keys and messages across devices for the new E-Mail infrastructure that LEAP is building. We participated in coding and discussion efforts and started prototyping "provider mediated keys" for key lookup at providers (see section 3.7.2).

2.3. OpenPGPConf: towards in-band key transmission

Due to a community split⁶, another PGP related conference took place in September 2016, just two months after the PGPSummit. Holger gave a state-of-our-NEXTLEAP-thinking talk there on "in-band opportunistic mail encryption", discussing why it might be preferable over "provider mediated keys". Werner Koch, core GNUPG developer, presented current works on the [OpenPGP Web Key Service](#) which is such a provider mediated key scheme. One particular problem with provider mediated schemes is that they not only require support from mail clients but also from providers which need to serve encryption keys. Serving encryption keys turns providers into a kind of "certificate authority", which in turn makes them more valuable hacking targets. In subsequent discussions with dkg, Vincent and several others we continued discussing various aspects of in-band keys.

2.4. NEXTLEAP research meeting: in-band federated identity

During 2-6 October 2016 we met with Carmela Troncoso (IMDEA), George Danezis and Marios Isaakidis (UCL), Francesca Musiani and Ksenia Ermoshina (CNRS), Nadim Kobessi and Harry

⁶ We can't fully trace the reasons behind the split. There were discussions around the fact that the PGPSummit was a fee-less event requiring application or invitation and it used the "Chatham house rules" which precludes participants from disseminating discussions without prior consent. And other discussions around the fact that OpenPGPSummit required paying a somewhat hefty 400 EUR to attend, precluding attendance from those who can not spent such money.

Halpin (INRIA), presenting our evolving in-band federated identity approach and discussing it in relation to ongoing crypto work (later termed “ClaimChain”) and usability studies and findings. Our intention to gather mail developers around the topic of how to achieve “Automatic E-Mail Encryption” solidified and we started organizing the AME2016 meetup. The D2.2 report grew a special “7.2.2 In-Band” section and thus incorporates our own and WP5 community discussions with the stakeholders as discussed in the introduction of Section 2.

2.5. AME2016: automatic mail encryption hackathon

Using the traction of discussions in previous events, Holger and Azul organized a 4-day meetup termed “Automatic Mail Encryption 2016” December 2016 in Berlin, and started drafting first specs for the “in-band opportunistic mail encryption” effort. Around 20 people attended at the OnionSpace, which was generously provided by the Renewable Freedom Foundation. Several maintainers of encryption-capable mail clients took part: Vincent Breitmoser ([K9](#)), Patrick Brunschwig ([Enigmail](#)), Bjarni Runar Einarsson ([mailpile](#)), Oliver Wiese ([Enzevalos](#) project), Kali Kaneko ([Bitmask/LEAP](#), also working with the EU project PANORAMIX) and researchers such as dkg ([ACLU](#)), Ksenia Ermoshina and Marios Isaakidis (both from NEXTLEAP). The conviction for the in-band approach grew stronger and eventually participants arrived at naming the joint effort “Autocrypt”. The gathering then created the [Autocrypt website](#) and implemented support into some mail clients. The tentative mission motto became “E-Mail encryption for everyone”. As part of AME2016, we also arranged, with help from Ksenia Ermoshina (CNRS), talks and discussions with Matthew Hodgson from [Matrix.org](#) (a promising federated chat open source solution with e2e encryption) and Alan Durac from [Wire](#), as well as with Dominic Tarr from [Secure Scuttlebot](#), who presented cryptographically secured append-only logs for messaging, next to a first theoretical presentation about ClaimChains from Marios Isaakidis (UCL). This unique gathering allowed us to discuss cross-cutting concerns between asynchronous/offline and synchronous/online encrypted messaging, as well as evolving a joint language for talking about decentralization and social federation of services.

2.6. 33c3 Panel on mail encryption “we fix the internet”

During the annual 13,000-people “Chaos Computer Club” convention, Holger took part in a panel discussion on E-Mail encryption. The other panelists were Volker Birke (pEp project) and Neal Wallfield (GNUPG core team) discussed community and implementation questions. While pEp aims to offer a library that can be integrated on all platforms, Autocrypt aims at a specification and implementation within existing mail clients and without the addition of new libraries. Both projects follow the in-band approach, however, which puts all logic for encryption into the end-user mail client code. The panel and subsequent discussions provided good opportunities for outreach and dissemination of our NEXTLEAP activities.

2.7. Webmail e2e: Mailvelope and Roundcube meetings

During January 2017 Holger met for a few hours in Karlsruhe with Thomas Oberndoerffer who authored the [mailvelope](#) plugin for email e2e encryption that is integrated by some large providers (posteo.de, Gmx.de, web.de). Thomas expressed interest in supporting Autocrypt within the mailvelope plugin but pointed out that providers (which integrate mailvelope) will need to add support as well.

In another separate meeting in Basel, Holger met with Thomas Bruederli, core maintainer of the popular [Roundcube webmail project](#). They discussed the difficult question how to get Autocrypt e2e encryption working in a webmail context. The current “1.x” architecture of roundcube, widely used around the world, does not easily allow end-to-end encryption because the actual “MIME” E-Mail message is currently constructed on the server-side. How to best bring end-to-end encryption to the webmail space remains a topic for Period 2 of NEXTLEAP.

2.8. Internet Freedom Festival 2017: more Autocrypt momentum

In the beginning of March 2017, we organized Autocrypt sessions in Valencia during the days ahead of the 700-people Internet Freedom Festival (IFF). During the sessions, we converged on a simplified approach to multi-device support (@@CAN YOU DESCRIBE?). We found a way to change the user workflow so that it is not necessary to rely on the IMAP protocol for multi-device support. This makes it easier for developers to implement Autocrypt support in mail clients, and opens the possibility for collaboration with email providers who do not support the IMAP protocol in their email ecosystem, such as Microsoft who rather uses “Exchange” protocols.

During the IFF, Azul, Holger, Ksenia Ermoshina (all from NEXTLEAP) along with dkg (ACLU) and Vincent Breitmoser (K9-Mail/Panoramix) presented usability prototypes for Autocrypt and gathered early user feedback. Several discussions with trainers and interested developers took place and resulted in consolidation of the Autocrypt in-band approach and preliminary plans for further collaboration.

2.9. Transnational Secure Messaging Meeting (TSMM)

TSMM brought together human rights activists, service provider technologists, open source programmers and academic cryptographers – from countries across Europe, North America, Latin America and Africa and organizations including LEAP Encryption Access Project, Autocrypt, GnuNet, Inria and Association for Progressive Communications – for a week of face-to-face discussions. The meeting took place at Calafou hackbase near Barcelona, Spain, during the week following the Internet Freedom Festival.

The meeting had three core objectives:

1. to bring together activists, service providers, academics and developers to discuss end-to-end encryption of email;
2. to conduct usability testing and integrate the results through discussions between developers and users to determine needs and define priorities;
3. to make progress on proposals for an open standard protocol with verified cryptography for free and open-source implementations of mix-nets to encrypt email meta-data in transit.

User testing took place throughout the IFF with both end-users and service-providers. At the subsequent TSM tests were demonstrated and feedback discussed between end-users, service provider technologists and developers. Service provider technologists from Kenya, Colombia, Mexico, North America and Spain were present to give input on their existing and prospective user bases, their needs and limitations and to discuss potential collaboration going forward. Documentation was highlighted as a particular issue here as multiple versions create confusion. Cross-platform development was also highlighted as an important area for attention given the diversity of the potential user-base between region-specific providers, with a focus on improving mobile development. A simplified mock user-interface that also exposes “what the crypto is doing” was made as well as coherent plan for ongoing usability research with the end-users at the meeting.

Specific sessions were allocated for discussion of technical details between developers from parallel open-source encrypted email projects including Autocrypt, GnuPG and LEAP, who are each taking different approaches to automating PGP. Cryptographers from INRIA and UCL were also able to explain mix-networking to participants and to engage with direct feedback on the practical implementation of their ideas with both developers and activist end-users. Currently, the space is still fragmented but this gave momentum to improve, with a focus on getting rid of attachments that end-users find confusing as well as minimizing user interaction with key material. The goal should be to minimize the confusion in the interface while still alerting users accurately on the security of their message.

2.10. EasterHegg 2017: E-Mail encryption and Autocrypt session

In April 2017, [Holger gave a 2h session](#) during the 600-people traditional “Easterhegg” gathering of the Chaos Computer Club in Germany. Someone from the audience was picked to try our Autocrypt usability prototype (all in the video recording linked above). Many attendants afterwards got back with positive feedback and wanted to install Autocrypt. Discussions ensued also with the people associated with the four-year old “pEp” (pretty easy privacy) project producing an open-source library for email end-to-end encryption. While the idea of using in-band key transmission is shared between Autocrypt and pEp the approaches differ in one major respect: Autocrypt is a specification with independent implementations whereas the “pEp engine” is a product and open-source library and the pEp proposition is that mail clients integrate it as an external dependency. Nonetheless, discussions about possible collaborations took place and continue to this day.

2.11. Informal Autocrypt / NEXTLEAP meetup: avoid passphrases

At the end of April 2017, we organized a multi-day gathering of E-Mail app programmers in Freiburg, Germany. Several people involved in the Autocrypt effort took part and continued discussions and coding together. Maintainers from enigmail, K9 and mailvelope discussed usability aspects for requiring “passphrases” for protecting secret key data at rest. We determined that, security-wise, passphrases are not adding much and thus [agreed to aim for removing passphrases](#) from default setups of e2e mail clients.

2.12. NEXTLEAP Launch meeting

Holger participated in the NEXTLEAP Launch meeting beginning May 2017 in Paris. He gave a quick summary talk about Autocrypt and other NEXTLEAP WP5 activities and discussed with NEXTLEAP collaborators on a panel before an interested wider audience in the Centre Pompidou, including the EU project officer Fabrizio Sistini.

2.13. Autocrypt level 1 meetup: finalizing the spec ...

In the middle of June 2017, we organized a “finalization” gathering for the “Autocrypt Level 1” specification in Freiburg, Germany -- headquarters of merlinux, which is heading WP5 work. We had several external developers and experts attending (many of whom were mentioned during the events above) and had a very productive meeting with the following major outcomes:

- Simplification of cross-user so called “prefer-encrypt” settings aiming to improve usability by reducing the amount of “unreadable” mail for end-users. It is now a boolean value with “nopreference” and “mutual” values. If two peers mail each other with “prefer-encrypt=mutual” then the communication will be encrypted by default. Otherwise the user only has a choice to encrypt. This feature of Autocrypt differs substantially from traditional approaches, which often took the existence of encryption keys at a key server or local keyring to mean that the peer prefers encryption -- when in fact many users have lost their secret key already, or do not want to receive encrypted mail by default because they are, for example, using Google Webmail without the mailvelope extension, or they are using a mobile untrusted phone.
- Addition of a key-gossip mechanism⁷ when sending encrypted mail. This addresses a long-standing usability problem with encrypted mail to multiple recipients. A mail client supporting key gossiping allows recipients to reply encrypted in almost all cases. Several mail client developers confirmed they want to implement this new spec feature.
- Defined the setup process and “Setup Message” that allows a user to synchronize two of his Autocrypt supporting mail clients with each other, thus exchanging key secrets and allowing to read all mail from both devices. “Level 1” uses 36 numbers (grouped in 9x4)

⁷ See here for the current specification of key-gossip:
<https://autocrypt.org/en/latest/level1.html#key-gossip>

for the Setup Code which cryptographically prevents the provider from getting access to the secret key contained in the Setup Message⁸.

- We formalized the recommendation to use “passphrase-less” keys so that users do not need to enter long passphrases when they want to send or receive encrypted mails.
- Bjoern Petersen took part from the [promising Delta-Chat project](#). Delta-chat is a Messenger over E-Mail with similar features as Telegram, Signal or Whatsapp. Delta-Chat already uses the Autocrypt spec and got deeply involved in specifying various aspects of the spec so that traditional E-Mail clients and the new Messenger interoperate well. For period 2 we plan to involve ourselves with Delta-Chat as it presents a natural way to bring Autocrypt and NEXTLEAP research to the synchronous messaging space, reusing the fundamental socio-technical federation that is available today through the E-Mail ecosystem.
- A joint rough roadmap to get widely deployed releases out in autumn 2017 and the spec finished and stabilized by then. We aim to help organize a “Autocrypt release party” as a major milestone also for our NEXTLEAP efforts.
- For various reasons the Autocrypt gathering decided from now on to talk about “Level 1 Autocrypt” instead of the previously used “Autocrypt Level 0”. To avoid confusion we consistently now talk about “Level 1” only, also in this D5.2 report.

2.14. A note on community involvements and the “Project Stakeholder Committee”

We treat the many mentioned external developers, trainers and activists around the evolving Autocrypt effort as stakeholders who provide feedback on and thus drive our NEXTLEAP efforts. We account for the various communications with those external stakeholders and how they influenced our own thinking and prototyping as well as those of partnering NEXTLEAP researchers in both the D5.1 and section 2 of this D5.2 report. We aim to continue this successful, iterative dynamic feedback process in period 2. Trying to rather select a more formal NEXTLEAP committee could have negative effects in relation to our WP5 community involvements. On the one hand, it could put existing contributors who are selected to the committee in the spotlight and introduce hierarchies. On the other hand, it could alienate potential contributors who are not selected for the committee. Maybe they would feel they are not part of the common effort or that they are marginalized in the initiative. Therefore we do not aim to dwell much on formalizing our outreach into a formal Project Steering Committee as it could be potentially harmful from a community development point of view.

⁸ See here for the current specification of Setup Messages and Setup Codes:
<https://autocrypt.org/en/latest/level1.html#autocrypt-setup-message>

3. Prototypes and releases

Through the community involvements outlined in Section 2 we deepened our contacts to integration communities (see D5.1 Section 2) and established new ones. Our involvements resulted in and were based on a number of prototypes and releases that we will outline and explain in this Section. We also update the use cases mentioned in D5.1 because of what we learned through interactions with individuals and communities.

3.1. Updated “use cases” that underly our prototyping efforts

In D5.1 Section 3 we described several use cases driving our efforts within NEXTLEAP. We realized later and in the course of the discussions described in the previous section 2 that we better state them from a more user-centered perspective. We therefore partly refine the previously stated use cases and relate them to our prototyping releases and activities of period 1.

The most significant change to the D5.1 use cases is that we previously talked about “providers” but most users do not have a clear model of what a provider is or what the separation between mail clients and mail providers looks like. In fact, this separation is different between webmail and device native applications. In any case, many end-users are less interested in the technical details than the usable artifacts of a technical arrangement.

3.1.1. Messaging use cases

We rephrase and simplify the descriptions of the messaging and encryption use cases and write them like this now:

Sending/reading encrypted messages: As a user, I want to read and write encrypted mail. As a user, I understand encryption to mean that only I and the recipient(s) can read the content of the message.

Multi-device handling: As a user, I want to read and write encrypted E-Mails from all my devices.

Out-of-band verification: As a user, I want to know if E-Mail encryption works correctly at the moment, and if it worked correctly in the past. I want to determine if the organisation or company that takes care of my email address tried to intercept and read my encrypted communications.

These use cases are covered by our in-band Autocrypt approach described below. Sending and reading encrypted messages is handled as part of “Autocrypt Level 1”, which also gives rudimentary means to manually manage secrets between devices. “Level 2” will then aim at

automating this management through secure pairing between devices, and also introduce out-of-band verification.

3.1.2. Accountability use cases

Within D5.1 we described these “accountability” use cases:

Social provider accountability: As a user, I want to know 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 client 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?

The “social provider accountability” we are trying to address by experimenting and documenting a state-of-the-art mail server that can be used by different communities (discussed above). The technical provider accountability we are trying to address with our ongoing ClaimChain prototyping activities described below.

3.1.3. Community agency

The original use case of D5.1 was described in Section 3.1.3 like this:

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.

This use case we tackled by the our [work on the VM-template](#) (see below) of a community run mailing list server which we aim to get real-life feedback on during period 2 of NEXTLEAP.

3.2. Autocrypt docs/specs for federated E-Mail e2e encryption

URL: <https://autocrypt.org/en/latest>

The Autocrypt specifications are extensive and actively being worked on by us and many external stakeholders. The most important part (for now) is found on the "Level 1" link in the left side navigation. It defines the current agreements on what mail client developers need to implement within their respective mail clients (often called MUAs in specifications). After the [June 2017 Level 1 meeting](#) there was agreement to soon go for public releases of the spec in conjunction with the upcoming autumn releases of mail clients with Autocrypt support.

As discussed in the introduction of our D5.1 report, we consider the socially federated E-Mail infrastructure to be in dire need of providing usable end-to-end encryption, and this area remains where we want to achieve impact with our NEXTLEAP implementation work. Most notably, we took a major part in organizing and establishing a new E-Mail encryption approach titled “**Autocrypt: E-Mail encryption for everyone**”. It grew in relation to our perspective that what is lacking with email client development is a community where to discuss and argue jointly. mail clients are written in all kinds of languages on all kinds of platforms. At language and platform specific conferences mail developers do not meet each other, resulting in rather isolated developments and diverging decisions also regarding user workflows, making the situation for users more difficult as well.

We cite here how the Autocrypt effort describes itself on the main web page (as of June 2017):

3.2.1. Introducing Autocrypt: E-Mail Encryption for Everyone

If users ask how they can secure their E-Mail the answer should be as simple as: use an Autocrypt-enabled mail app!

Why improve E-Mail? E-Mail has been declared dead many times but refuses to die. It remains the largest open federated identity and messaging ecosystem, anchors the web, mobiles and continues to relay sensitive information between people and organisations. It has problems but do you prefer the proprietary, easy-to-track mobile phone number system to become the single source of digital identification?

Why a new approach to E-Mail encryption? Encrypted E-Mail has been around for decades, but has failed to see wide adoption outside of specialist communities, in large part because of difficulties with user experience and certification models. Autocrypt first aims to provide convenient encryption that is neither perfect nor as secure as traditional E-Mail encryption, but is convenient enough for much wider adoption.

3.2.2. The social Autocrypt approach

The Autocrypt project is driven by a diverse group of mail app developers, hackers and researchers who are willing to take fresh approaches, learn from past mistakes, and collectively aim to increase the overall encryption of E-Mail in the net. The group effort was born and named “Autocrypt” on December 17th 2016 by ~20 people during a 5-day meeting at the OnionSpace in Berlin. It’s a dynamic, fun process which is open to new people, influences and contributions. See [contact channels and upcoming events](#) on how you may talk with us and who “we” are currently.

3.2.3. The technical Autocrypt approach

Autocrypt uses regular E-Mail messages between people to piggyback necessary information to allow encrypting subsequent messages; it adds a new *Autocrypt* E-Mail

header for transferring public keys and driving encryption behaviour. By default, key management is not visible to users. See [Autocrypt features](#) for more technical and UI cornerstones.

We are following this approach step-by-step using different “levels” of implementation compliance. Driven by usability concerns, we are refining and implementing [Level 1](#) in several mail apps during Spring 2017. If you are interested to help please [join our channels and look at where we meet next](#).

Besides our focus on both technical and social features of the approach, you will note that we do not mention “NEXTLEAP” on the front page. This stems partly from the fact that Autocrypt is larger than the EU project and partly from our view that it’s not the NEXTLEAP “brand” we want to establish or advertise for. In our view, trying to push the NEXTLEAP brand to the forefront could jeopardize this community effort as it could be easily seen as a kind of appropriation by a funded project, and we believe a grassroots effort supported by the European Commission but not viewed as *controlled* by the funding will be more successful in reaching more developers

3.3. Usability prototypes

We made extensive use of rapid prototypes to illustrate workflows and inform the discussion for the Autocrypt specs and the ClaimChain protocol considerations. We also used them for demonstrating the workflows in presentations and for user testing. Basing the spec discussions in user workflow prototypes also makes sure that the specs are based on actual impact for the users.

Autocrypt aims to enable end-to-end encryption most of the time with the least user interaction possible. However we need to make sure users know whether their emails will be encrypted to prevent a false sense of security. Therefore the user interface has to be minimal but to the point. The only way to achieve this is through testing user interfaces with actual users to find out how they interpret the information offered. We have defined user prototypes to start testing with, and intend to use, where applicable, a modified version of the Information Sheets and Informed Consent (see D7.1) that was used in the user-studies by CNRS in WP3.

One aspect of autocrypt that requires user interactions is the setup and configuration of autocrypt. In level 1 we cannot enable autocrypt by default because the user might be using multiple devices. We need to avoid the unpleasant surprise of not being able to read encrypted emails on other devices.

Another interesting UI element is the exchange of messages between autocrypt users. We need to convey which emails are encrypted and the fact that an initial unencrypted mail exchange is needed.

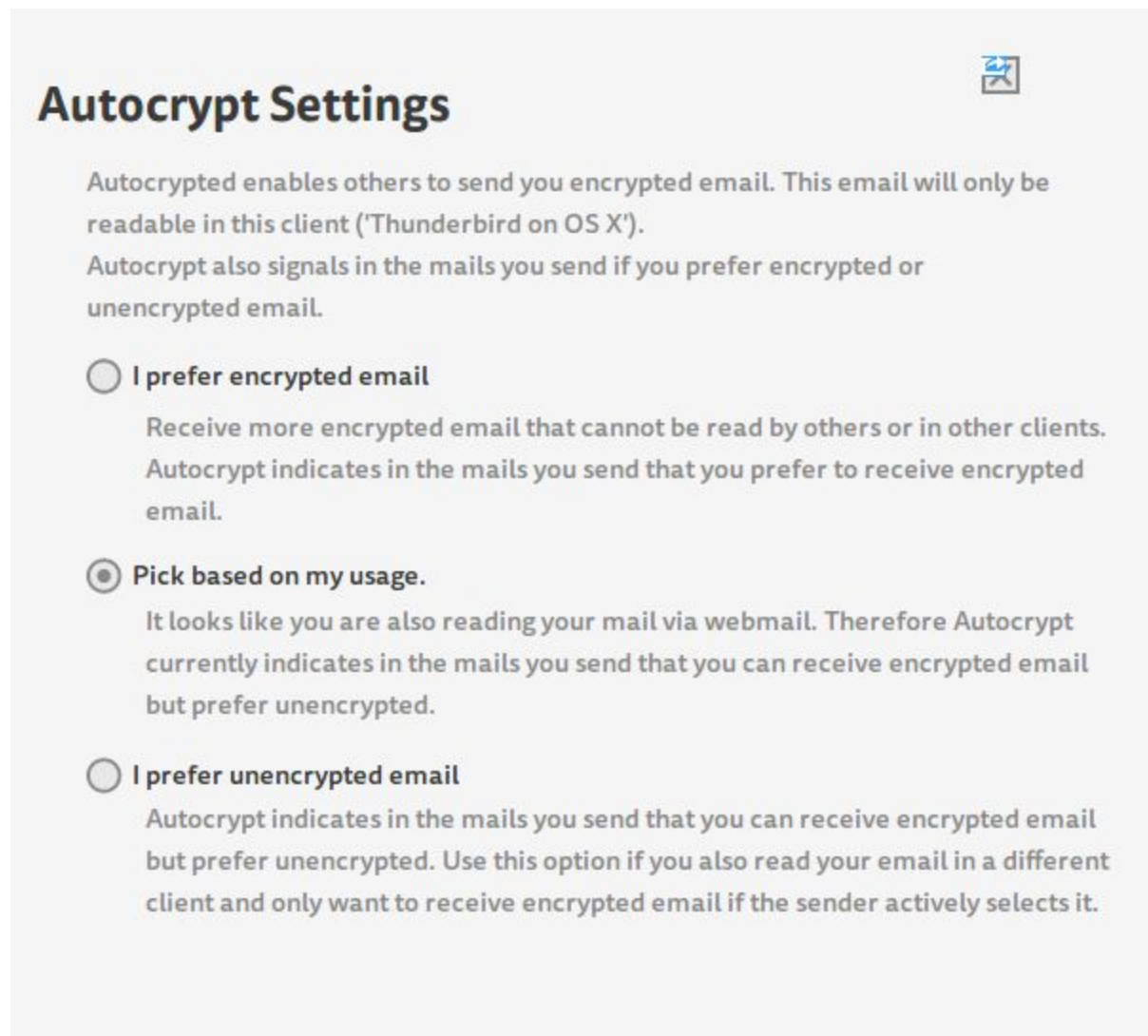
In order to inform the development of autocrypt enabled clients we so far developed three prototypes:

- A static UI prototype for the configuration settings
- A mail composition and reading prototype for the user interaction backed by a Ruby on Rails server
- A single page HTML application that combines the configuration settings and the mail exchange UI.

We ran a number of UI walkthroughs with the prototypes and iterated on the user visible interface with volunteer developers. We also gathered advice on the user interface from EFF experts at the UI helpdesk at the IFF. In addition the prototypes proved useful to demo the autocrypt concept at workshops and introduce people to the ideas behind it from a user perspective.

3.3.1. Plain UI prototype for configuration settings

URL: <https://gomockingbird.com/projects/cf74t2m>



Autocrypt Settings

Autocrypt enables others to send you encrypted email. This email will only be readable in this client ('Thunderbird on OS X').

Autocrypt also signals in the mails you send if you prefer encrypted or unencrypted email.

☐ **I prefer encrypted email**

Receive more encrypted email that cannot be read by others or in other clients. Autocrypt indicates in the mails you send that you prefer to receive encrypted email.

☒ **Pick based on my usage.**

It looks like you are also reading your mail via webmail. Therefore Autocrypt currently indicates in the mails you send that you can receive encrypted email but prefer unencrypted.

☐ **I prefer unencrypted email**

Autocrypt indicates in the mails you send that you can receive encrypted email but prefer unencrypted. Use this option if you also read your email in a different client and only want to receive encrypted email if the sender actively selects it.

At [AME2016](#) we used different mockups to discuss the configuration settings - in particular the prefer-encrypt setting. After coming up with a proposal in a working group we invited a usability expert who had not been part of the discussion to do a UI walkthrough. The walkthrough exposed that we were not able to communicate the impact of the different UI options well. As a result we continued investigating possible solutions.

3.3.2. Message exchange prototype in Rails

<https://github.com/autocrypt/message-flow-prototype>

Azul Emails Compose Settings

Settings

Autocrypt

Autocrypt allows exchanging encrypted email.

Autocrypt is enabled.

[Start encrypting to your contacts...](#)

1. Send an email to the contact you want to encrypt to.
2. Tell them to enable autocrypt. Now they can encrypt to you.
3. Once you received an autocrypt enabled email from them you can also encrypt to them.

[Advanced Configuration...](#)

Autocrypt can indicate in the mails you send whether you would like to receive encrypted email or not. Encrypted email can only be read in this application.

☐ Prefer encrypted email.

You only read your email in this application or you accept that you cannot read encrypted email elsewhere.

☒ No preference.

Your contacts decide whether they want to encrypt email to you.

☐ Prefer unencrypted email.

You also read your email in a different application and only want to receive encrypted email when your contact insists on it.

Save Settings

For the NEXTLEAP meeting in Madrid and the IFF 2017 we prepared a Ruby on Rails based prototype that demoed the message exchange flow and used GnuPG as a backend for key creation and encryption.

Besides using the prototype to demonstrate the message flow at both occasions we also used it to run 4 rounds of user testing amongst Autocrypt participants and the UX helpdesk at the [IFF](#).

3.3.3. User interface prototype as single page application

<https://github.com/nextleap-project/ui-autocrypt>

Write mail Mailbox Preferences

Logged in as
Alice
([change](#))

Autocrypt Settings [☒ enable]
Please enable Autocrypt on *only one* device. ([why?](#))

[Hide Advanced Settings](#)
☒ I prefer to receive encrypted mail.
☐ I prefer not to receive encrypted mail.
Autocrypt will encourage your peers to send you encrypted mail.

In parallel to the Message Exchange Prototype, Daniel Kahn Gillmor developed a prototype for the configuration settings in HTML, CSS and Javascript. He demoed this prototype at IFF as well and we decided to combine our efforts and move forward with this approach. It has the advantage of not being tied to a particular framework like Ruby on Rails and thus allows more people to contribute.

Throughout the following months we adapted the prototype in order to answer different questions and conducted user-testing sessions at [IFF 2017 \(March\)](#) and [Easterhegg 2017 \(April\)](#) with a variant of it. It informed the debate about the controversial prefer-encrypt setting and helped come up with a workflow for minimal support for multi device setups in Level 1 of the Autocrypt spec.

Along the way we tagged different releases that capture the state of the prototype that informed a certain discussion. You can find the different releases here:

URL: <https://github.com/nextleap-project/ui-autocrypt/releases>

After unpacking a release you can point your browser to "index.html" and get started with the UI demo. First action is that you go to preferences and "enable autocrypt", then send a mail to Bob, then "CHANGE" to Bob login, enable autocrypt and reply with encryption toggled to on.

The last discussion the prototype informed was enabling private key transfer in Level 1. This workflow is essential to prevent people from being locked in to one particular client instance and already allows basic multi device usage.

The screenshot displays a web-based email interface. At the top, there are three tabs: 'Write mail' (active), 'Mailbox', and 'Preferences'. On the right, it says 'Logged in as Alice' with a large green 'Alice' and a '(change)' link below it. The main form has fields for 'From: Alice', 'To: Bob', and 'Subject: Re: subject'. Below these is a checkbox labeled 'Encrypt' which is checked. A 'Send' button is located to the right of the text area. The text area contains the message: 'Hey Bob, How are you? Great you got the encryption setup. I can now also encrypt to you.' To the right of the text area is a black silhouette of a smartphone with a white screen and a home button. Below the phone icon is a '(change)' link.

3.3.4. User interaction test prototype

During the Level 1 hackathon we demonstrated the User Interface Prototype and it informed the discussion around the Autocrypt Setup Message workflow (see [the summary of the related gathering](#)). We asked implementers what they would like to see as next steps. They were happy with what the current UI prototypes achieved and we came to the conclusion that the Level 1 features can soon be tested from real-life code instead of UI prototypes. The real-life implementations are progressing nicely and we expect to have email client releases during Autumn which can then be used for driving real-life user tests.

This allows us to strengthen our user focus and test how much friction the Autocrypt workflow creates for users and also to prototype and push ahead with "Level 2" features such as out-of-band verification and pairing where we aim to use a suitable version of ClaimChains (D2.2). Based on the User Interface Prototype we can test this in a environment with little distraction. Existing email clients have lot's of UI elements to support existing requirements and workflows including traditional key management. With the prototype we can leave these out and test user interaction in a streamlined environment.

We are currently collecting feedback about this approach and asking implementers for specific scenarios, ui elements, concepts or wordings they would like to see tested. We expect the next incarnation of the prototype to inform decisions in the spec, ui recommendations and the actual implementations.

3.4. Python command line tool for prototyping, integrating e2e encrypted mail

URL: <https://github.com/autocrypt/py-autocrypt>

The "py-autocrypt" command line tool is installable by typing "pip install autocrypt" on Linux and Mac platforms. You will need a "gpg" or "gpg2" command line tool also installed on your machine. See released docs for getting started: <https://py-autocrypt.readthedocs.io/en/latest/>

The py-autocrypt prototype is used for prototyping, experimentation, custom mail server setups and eventually for adding Autocrypt support to existing mail server software. We, however, do not aim to integrate it into existing encrypted Mail apps like "Thunderbird/Enigmail" or "K9/Openkeychain/Android" because they are written in different programming languages and in very different environments. We do not think that a single code base could be used easily from the very diverse set of mail clients which is why our Autocrypt prototypes relate to the specification and UI-prototyping efforts to help implementers add support themselves. We already merged contributions from external contributors like "juga" who did the groundwork for using the [pgpy](#) backend instead of the currently required "gpg" command line tool.

3.4.1. Command line tool

The "autocrypt" command line tool allows to generate keys and process incoming and outgoing mail. When you type "autocrypt" you get an overview of subcommands:

```
Usage: autocrypt [OPTIONS] COMMAND [ARGS]...
```

```
    access and manage Autocrypt keys, options, headers.
```

Options:

```
--basedir PATH    directory where autocrypt account state is stored
```

```
--version          Show the version and exit.
```

```
-h, --help         Show this message and exit.
```

Commands:

<code>init</code>	<code>init autocrypt account state.</code>
<code>status</code>	<code>print account and identity info.</code>
<code>add-identity</code>	<code>add an identity to this account.</code>
<code>mod-identity</code>	<code>modify properties of an existing identity.</code>
<code>del-identity</code>	<code>delete an identity, its keys and all state.</code>
<code>process-incoming</code>	<code>parse autocrypt headers from stdin mail.</code>
<code>process-outgoing</code>	<code>add autocrypt header for outgoing mail.</code>
<code>sendmail</code>	<code>as process-outgoing but submit to sendmail...</code>
<code>test-email</code>	<code>test which identity an email belongs to.</code>
<code>make-header</code>	<code>print autocrypt header for an emailadr.</code>
<code>export-public-key</code>	<code>print public key of own or peer account.</code>
<code>export-secret-key</code>	<code>print secret key of own autocrypt account.</code>
<code>bot-reply</code>	<code>reply to stdin mail as a bot.</code>

3.4.2. Claim Chain prototyping

The Autocrypt effort itself so far has deliberately not specified a mechanism for securing against providers maliciously manipulating encryption keys in messages. However, we prototyped a version of the D2.2 “ClaimChain” data structure from the report on “privacy-preserving federated identity” protocols. The data structure is used as a basis for managing and verifying key material within our py-autocrypt efforts. These efforts are using the “in-band” variant of ClaimChains, which is discussed in section 7.2.2 of the aforementioned report. We have implemented a basic append-only log of claims where each entries is persistently addressable, allowing to express claims which reference entries in other chains consistently.

We store ClaimChains locally and aim to exchange and cross-reference them during out-of-band verification. The chains therefore remain completely out of sight of providers or storage services and only two users’ devices will see each other’s claim chains. Particularly E-Mail providers have then no way of knowing when and if users out-of-band verify with each other, turning active attacks into risky business. Later on we want to experiment with supplementing this out-of-band use of ClaimChains with in-band gossiping to support users in maintaining uncompromised communications. A central concern here is the ability of users to distinguish “lost device” events from “man-in-the-middle” attacks -- something that current state of the art messaging apps do not provide.

Within current E-Mail ecosystem practises it is not directly feasible to have lots of key changes (like is more common with some low latency messaging systems) and therefore we currently do not need to consider large claim chain lists and thus do not need to implement “skip lists” and other optimizations described in D2.2.

The ClaimChain prototyping efforts are part of the [py-autocrypt-0.7 release](#) which is automatically tested and well [documented](#).

3.4.3. Online bot answering mails with Autocrypt headers

Contained in py-autocrypt is also the code that drives the responses when you send E-Mail to the publically reachable **bot@autocrypt.org** address -- it will reply with what kind of autocrypt headers it saw from you and also will send its own Autocrypt header. You can typically inspect the headers of any received mail by tapping something like “Show Source” or “Show original mail” in your mail client.

3.5. Privacy-preserving ClaimChain prototyping

The implementation of ClaimChain described in 3.4.2 is based on the initial ClaimChain design described in D2.2, and it is used to test the ease of integration of the NEXTLEAP federated identity proposal into the email ecosystem. However, that initial design has no privacy protection regarding the social graph (i.e., readers learn about all ClaimChains references to other chains). This lack of privacy was solved through a new design described in D4.2 whose operation is in essence the same as the one build for testing within WP5. A prototypical implementation of this new design by UCL and IMDEA lives at <https://github.com/gdanezis/claimchain-core>. The performance of this implementation is tested via simulation in order to make sure that it is suitable for being integrated in the end-to-end mail encryption scenario in the next period of the project.

3.6. Prototyping with mailman integration community

Merlinux has started researching, documenting and showcasing a state of the art mailing list server setup. This setup is is to serve as a real-life basis for integrating Autocrypt with Mailman3, a popular mailing list implementation in period 2. This prototyping effort aims to gather real-life experiences and also to experiment with how admins can collaborate in a lightweight manner on maintaining and caring for a mail server. Florian Schulze (merlinux) lead the effort to setup a mailing list server on the publically reachable internet domain <https://lists.codespeak.net>. We aim to develop it as a template for a community-run Mailman-based mailing list server and also as the basis for integrating Autocrypt and ClaimChain functionality as envisioned in Sections 2.3 (Mailman) and 3.1.3 (Community run infrastructure) of the earlier D5.1 report. As of June 2017 this effort resulted in the following preliminary results:

- deploying lists.codespeak.net as a VM based on the just released Debian 9 with mailman3, letsencrypt, postfix, nginx, opendkim -- all considered best practise services
- [Etckeeper](#) is popularly used for version-controlling all system configuration and we suggest it as a primary tool for the collaboration between multiple “spare time” admins. We also setup an “admin” mailing list where all admins of a machine are subscribed. The change log of the system configuration along with commit messages also serves as documentation for why and who changed system configuration
- [Rspamd](#) is used for spam filtering and mail-processing hooks for later use with py-autocrypt
- [borgbackup](#) is used to remotely backup system state + configuration of the whole VM

We consider this community-run mailing list server infrastructure as complementary to the effort of the larger LEAP provider platform which targets professional providers with thousands of users and does not include mailing list services.

3.7. Ongoing collaboration with the LEAP integration community

While our main efforts revolve around Autocrypt and ClaimChain we followed up on our involvement with LEAP, engaging within the community and contributing where possible.

3.7.1. Nickserver: key management for federated LEAP e2e platform

<https://github.com/nextleap-project/nickserver>

As part of our earlier provider mediated key approach discussed in D5.1 we added key lookup to LEAP's keyserver alternative nickname. Nicknym follows the idea of providers certifying keys for their users. There are similar but slightly different approaches like Werner Koch's web key directory and Mailvelope's key servers.

In order to work around these differences nickserver works as a proxy for the provider's own users for key lookup to other providers. It offers an API to lookup the key for a given email address. It will then try different strategies to retrieve a key.

While we have shifted our focus to client centered approaches, we support maintaining the nickserver codebase and adopt it to changes. Integrating with provider based approaches makes more keys available. It therefore opens up the possibility for users to encrypt to more people.

3.7.2. Omniauth-ss0: secure login to federated LEAP e2e platform

<https://github.com/nextleap-project/omniauth-ss0>

Providers are faced with the challenge of allowing the user to access different services with a single password. Providing a so called “single-sign-on” (SSO) mechanism for this has both security and usability benefits. For LEAP we researched implementing a single service for

handling authentication and password management and nothing else. Other services can then rely on this and therefore cannot leak the password.

With UnlimitID NEXTLEAP partners have developed a protocol that additionally preserves privacy - or to be more precise, unlinkability - between the identity provider (idp) and the apps. The idp does not learn which apps a user uses. But since we wanted to authenticate different services of the same provider, privacy cannot be preserved. If the idp and the app are served by the same party they can always overcome unlinkability.

We therefore looked into the most commonly proposed solutions such as OAuth2 and OpenID connect. Some of them have been formally verified recently. But the complexity requires a lot of checks in the implementation to prevent security issues and even companies with well funded security departments such as [github](#) have fallen into these traps. On the other hand LEAP's requirements are really simple. They need authentication (identify the user) rather than authorization (allow the app to act on behalf of the user).

Due to these considerations we decided to implement the simplest thing that that could possibly work. We found this to be [ai/sso](#) developed by Autistici/Inventati which we implemented [a ruby version](#) of and [integrated](#) in the most common ruby authentication mechanism omniauth.

An initial deployment at a provider with more than 100.000 users proved useful but also revealed that the provider also wanted single log out (SLO). Since our current approach cannot provide that we might look into a solution to both SSO and SLO, that will require an additional communication channel between the idp and the relying party.

3.7.3. Leap_web: password recovery within the federated LEAP e2e platform

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

One of the hard problems in identity management is recovering from exceptional situations like device loss or forgotten passwords. In collaboration with ThoughtWorks we looked into these issues. A LEAP provider will store recovery data and in both scenarios recovery happens in two steps:

1. Authentication to the provider to access the data
2. Recovery of the key material to decrypt the data

While we need to be able to recover the account from a single high entropy password it is crucial that the provider cannot use the data stored for authentication to recover the key material and decrypt the data.

Since with secure remote password (SRP) we already have a mechanism in place that achieves the same thing for the normal password workflow we decided to use the same mechanism for password recovery. Now the recovery codes can be created on the client and registered via

SRP with the provider to allow for Authentication. The client can then regain access to the data with the recovery code and use it for recovering the key material.

4. Outlook on period 2 activities

Our main collaboration and integration activities around our privacy-preserving “federated identity” and “secure messaging” implementation efforts remain situated in the E-Mail ecosystem. We briefly outline our current planning for period 2 activities here:

- **Further involving ourselves in the Autocrypt community, helping it to succeed and thus impact widely used mail clients as well as helping new efforts to evolve.** See to get a version of ClaimChains become part of some Autocrypt supporting mail apps to help securing key distribution among the E-Mail ecosystem. Finalize and help make Autocrypt level 1 deployment a reality and start helping with Level 2 (multi-device and pairing).
- **Explore collaboration possibilities with Bjoern Petersen from Delta-Chat as this opens up our Autocrypt NEXTLEAP work to synchronous low-latency”chat” messaging.** Delta-Chat competes with the likes of Signal, Telegram and WhatsApp and strongly relates to Task 5.2 and Task 5.3 of the NEXTLEAP proposal. Autocrypt provides the end-to-end encryption feature of Delta-Chat and thus a collaboration provides an excellent opportunity to extend the use of the E-Mail federated ecosystem and provide a more decentralized alternative to the centralized Signal, Telegram and WhatsApp messengers.
- **Further explore user-testing through Autocrypt UI prototypes, in particular for implementing out-of-band verification workflows and device pairing.** Explore co-operating with TU Darmstadt researchers (external to NEXTLEAP) on doing usability studies after initial friendly contacts. Discuss also with CNRS their draft of use case studies and with CNRS and UCL jointly how to do more user testing in period 2.
- **Release refined and extended versions of py-autocrypt to support ongoing efforts to bring E-Mail encryption to various email clients and projects including Bitmask / LEAP and mailman.** Both are written in Python and can thus use our well-tested code base. We also consider to turn py-autocrypt into a more generally useable server-side component for helping all kinds of server-side mailing software to handle encryption according to Autocrypt specs. We also aim to evolve our “community-run” mailing list server with integrated Autocrypt support.
- **Develop a concrete ClaimChain design for Autocrypt in-band key exchange.** We aim, with the help of our partners, to design and document concrete claims related to out-of-band verification for Autocrypt identities and for verifying key consistency between peers. A primary usability concern here is to alert the user about provider malfeasance and not send the same alert for the typical “device loss” scenario. We are to discuss with our NEXTLEAP partners how these efforts can benefit from period 1 results of WP2 and

WP4 activities and how the evolving Autocrypt developments can be supported by their period 2 efforts.

- **Collaboration on considering integration with mixnet technologies as developed during the ongoing EU project Panoramix, in particular with Vincent Breitmoser and Moritz Bartl from the Renewable Freedom Foundation.** Also Collaboration with MAZI and potentially other EU/CAPS projects, helping them make use of the upcoming Autocrypt standards and mail apps which support it.
- **Continue engaging with a diverse range of implementers in the “zero-knowledge” application space** e. g. from matrix.org, zerotier.com, cryptpad.fr, borgbackup and several others. *Zero-knowledge* means here that a service provider does not have cleartext user data and thus can not be compromised to leak that data. This is typically achieved by end-to-end public key encryption but is also realized through symmetric encryption where the secret is shared between users but unknown to the mediating server. We consider helping to facilitate an international event bringing together the diverse set of interesting players in that space.