Can Johnny Build a Protocol? Co-ordinating developer and user intentions for privacy-enhanced secure messaging protocols

Ksenia Ermoshina
CNRS
Paris, France
ksenia.ermoshina@cnrs.fr

Harry Halpin
Inria
Paris, France
harry.halpin@inria.fr

Francesca Musiani
CNRS
Paris, France
francesca.musiani@cnrs.fr

***Abstract*—As secure messaging protocols face increasingly widespread deployment, differences between what developers "believe" about user needs and the actual needs of real-existing users could have an impact on the design of future technologies. In the domain of secure messaging, the sometimes subtle choices made by protocol designers tend to elude the understanding of users, including high-risk activists. We'll overview some common protocol design questions facing developers of secure messaging protocols and test the competing understandings of these questions using STS-inspired interviews with the designers of popular secure messaging protocols ranging from older protocols like PGP and XMPP+OTR to newer unstandardized protocols used in Signal and Briar. Far from taking users as a homogeneous and undifferentiated mass, we distinguish between the low-risk users that appear in most usability studies (such as university students in the USA and Europe) and high-risk activist user-bases in countries such as Ukraine and Egypt where securing messages can be a matter of life or death.**

I. Introduction

In the wake of revelations of mass surveillance and increased privacy concerns from the general public, many hope that secure messaging applications can converge to be the "default" option for communication, yet developers still are in a state of flux about their security and privacy properties and users have not converged on a single application. For example, there is still debate on cryptographic properties such as forward secrecy, group messaging, and repudiation. There is no clear standard to adopt with all these properties, as older standards like PGP not offer these properties. In terms of privacy, work is much more immature than security properties; applications such as Signal expose metadata via associating users with their the phone number.

Due to this lack of agreement, next-generation secure messaging is unstandardized and fragmented, leading to state of play where secure messaging users currently exists in dozens of "silos" that are completely unable to interoperate with each other: WhatsApp users cannot chat with Signal users, Cryptocat users cannot communicate with ChatSecure users, and so on. This is in stark contrast to older federated, standardized, and freely licensed technologies such as XMPP with Off-the-Record (OTR) messaging or e-mail with PGP. For example, any email service can openly communicate with another (Gmail to Outlook, etc.) in a federated fashion. To summarize, the properties for new protocols and applications can be classified into six broad categories:

- Security Properties
- Group Support
- Privacy Properties
- Decentralization
- Standardization
- Licensing

Currently developers simply imagine what properties users likely need, and these properties may or may not actually satisfy the needs of end-users. In particular, high-risk users may care about very different properties than low-risk users in terms of their threat models. If developers themselves are relatively low-risk users and building tools aimed at high-risk users, then the tools may or may not match the needs of these high-risk users.

Foundational papers in usable security studies call for end-users to be helped [23], [2]. Recently, more papers have asked for developers to be helped [14], [1]. Yet this is the first paper to study *the interaction between developers and users* to our knowledge. In this paper, we first state our initial theses in Section 2, with background is given in Section 3 in terms of the six aforementioned categories. Our qualitative methodology is explained in Section 4, with the results from interviews being delved into in Section 5, and conclusions in Section 6.

II. Problem Statement

Our first thesis is the ***Developer-User Disconnect:*** We hypothesize that *the 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

EuroUSEC '17, 29 April 2017, Paris, France
Copyright 2017 Internet Society, ISBN 1-891562-48-7
http://dx.doi.org/10.14722/eurousec.2017.230016

developers of secure messaging applications hypothesize what properties a protocol should have based on their beliefs about users. These properties may or may not line up with the expectations of users, and therefore the goal of our project is to determine the properties developers believe are important and see if these properties match the properties wanted by users. Though some attempts are made to gather and analyze user experience via online feedback forms and rare offline workshops (observed at international events such as CCC, IFF or RightsCon), contact between high-risk users and developers seems minimal. Such feedback can be produced by tech-savvy high-risk users willing to contribute in co-developing free and open-source projects, although high-risk users are of course often engaged in more pressing issues at hand. Users and developers need to converge in order to harmonize the needs of users with the concrete design decisions made by protocol designers.

Our second thesis is the ***High-Risk User Problem:*** We hypothesize that *high-risk users have different needs and behavior than low-risk users*. Although seemingly obvious, in most studies of end-to-end encrypted messaging, the features and problems users encounter may not be representative of actual end-users as the user base that is often studied in usability experiments in the United States and Western Europe are often a rather homogeneous selection of students from a low-risk background. Although it can be claimed that all users are to some extent "high-risk" potentially, we would argue that it is sensible to divide users between those *high-risk users* who can in the short-term suffer concrete physical harms such as long-term imprisonment and torture as a result of information security, and those *low-risk users* who do not have immediate consequences due to failures in information security. High-risk users tend to be in areas where information security is important due to political instability, although well-known activists and persecuted minorities in "stable" countries would count as high-risk users. Most usability studies over PGP, OTR, and Signal are still done with low-risk users such as college students, despite the use of secure messaging being considered to be important to high-risk users [22], [20], [19]. We would hypothesize that high-risk users have different threat models and so different requirements for privacy and secure messaging. As most protocol developers are not high-risk users themselves, they imagine the threat model of high-risk users as well as the feature set they may desire and what trade-offs are reasonable, but these projections of the needs of high-risk users could easily be inaccurate.

Our third thesis is that ***Security Trainings Differ by Risk:*** We hypothesize that *trainers in countries with high-risk users (Ukraine, Iran, Egypt, Russia) will suggest different practices and tools than trainers in low-risk countries*. This is not self-evident; it is also possible that trainers always train users the same way to use the "best of breed" applications for secure messaging and anonymity ("Use Signal, use Tor" as a popular saying goes), regardless of the context of the training. Security *trainers* can be defined as users who give public or private trainings in using secure tools such as end-to-end encrypted messaging to other users. The trainers themselves and their audience may be either high-risk or low-risk. Trainers are particularly interesting from methodological point of view as they act as facilitators that translate between the technical community and users, and have a (perhaps biased) view of a large swathe of their local user-base. The trainers we have interviewed either work at non-profits focused on human rights (ISC Project, Tactical Tech, EFF, Privacy International, etc.) or are involved in the informal "crypto-party" movement. [1] Some of these trainers had studied information security or programming, yet the majority were self-trained. STS considers trainers to be "knowledge brokers" [17] and "experience-based experts" [6] that have "interactional expertise" in the topic of security but do not necessarily code. They can communicate with developers by aggregating and translating user feedback and help "translate" to developers in threat-models and a nuanced analysis of users. They rapidly react to new technical tools appearing in the field, test them personally, and then decide whether or not to include in their trainings. Trainers coin a specific "pidgin"[12], a situated language based on metaphors and specific explanation schemes (often including visual materials) that offer a translation of cryptographic terms and information security jargon by embedding their understanding of the field into the specific contexts of usage.

III. Background

In this section, we discuss the history of secure messaging, with properties and applications that we studied summarized in Table I. Developers from the listed applications were the focus of interviews.

A. What is Secure Messaging?

Previous studies in security and usability, are normally based on looking at the usability of a single protocol such as PGP or Signal, regardless of the properties of the underlying protocol or how these protocols and their respective applications have evolved over time. Furthermore, although there has been excellent technical work in systematizing the security properties of secure messaging applications [21], there has been little historical work on the underlying reasons for the evolution of secure messaging from PGP-based applications to more modern secure messaging applications. The term *secure messaging* has been used increasingly in the technical literature [21] and popularization such as EFF's Secure Messaging Scorecard[2] to only refer to "post e-mail protocols," therefore excluding older open standards such as PGP. In contrast, we will take secure messaging in the broadest possible sense of the term to refer to all protocols and applications that offer end-to-end encryption, where any passive *adversary cannot read the cleartext of the message*, including the service provider. In other words, only the sender and intended recipient should be able to read the cleartext of the message, and thus the protocol must at least offer confidentiality. It may or may not offer integrity and authentication, as well as any privacy or anonymity properties. Since the space of possible active attacks is so large, the active attacker is excluded from this definition but defenses against certain classes of active attacks may be security features of the protocol (for example, stripping message signatures and forwarding messages [7]). This definition of secure messaging includes both PGP and newer "secure messaging" applications such as Telegram and Signal. We will include synchronous messaging protocols that are able to receive messages only when both users are online (these brands

[1] https://www.cryptoparty.in/

[2] https://www.eff.org/secure-messaging-scorecard

Application	PGP (LEAP)	OTR	Signal	Conversations	Wire	Briar	Ricochet
Repudiation (Security)	no	yes	yes	yes	yes	yes	yes
Group Support	yes	no	yes	yes	yes	yes	yes
No Metadata Collection (Privacy)	no	no	no	no	no	yes	yes
Decentralization	yes	yes	no	yes	no	yes	yes
Standard	yes	yes	no	yes	no	no	no
Open Licensing	yes	yes	yes	yes	yes	yes	yes

TABLE I. PROPERTIES OF SECURE MESSAGING APPLICATIONS STUDIED

of messaging applications are often called "instant messaging" or "chat" applications) as well as asynchronous messaging applications can receive messages when a user is offline, although many "chat programs support such functionality. *Group support* means that the application supports messages between one to two or more users. Other terms will be introduced to be defined later, such as "forward secrecy."

B. The Evolution of Secure Messaging Protocols

1) Encrypted E-mail: SMTP, the protocol originally used for transferring email, is one of the first messaging standards, but SMTP has no confidentiality of content or even authentication of headers for network-level routing. However, it is one of the oldest and most widely deployed standards for asynchronous messaging.[3] PGP (Pretty Good Privacy) was created to add end-to-end encryption capabilities to e-mail in 1991 by Phil Zimmerman, with the OpenPGP set of standards was finally defined years later in 1997 in IETF to allow the open implementation of PGP without conflicts with RSA patents or proprietary software.[4] OpenPGP is implemented in both desktop and mobile e-mail apps, including Outlook, Apple Mail, and Thunderbird through plug-ins. An alternative standard for encrypted email called S/MIME was developed that is also supported via plug-ins by most major e-mail clients, where the main difference between OpenPGP and S/MIME is that S/MIME requires the installation of certificates provisioned by centralized certificate authorities.[5] In contrast to centralized approaches, OpenPGP offloads the key management to the users via a decentralized "Web of Trust" model. In general, PGP was considered to have poor usability as users could not understand key management and judge the trust relationships in keys, or even understand the interface [22]. OpenPGP and S/MIME also work on mobile devices, such as the PGPMail for iOS and K-9 Mail (via plug-ins such as Openkeychain) for Android, but as OpenPGP binds the key to the particular device, there has often been concern about how to securely transport any long-term private key material between devices, and so mobile adoption of encrypted email is considered to be low among users and problematic in terms of security. Although these challenges of PGP on the mobile platform are well-known [16], mobile PGP has not been subject to usability studies in the same manner that PGP itself has. S/MIME has had some usability studies and in general shows better usability than PGP, insofar as key management does not have to be maintained by the end-user, but users still have trouble understanding the interface [13]. In terms of the underlying protocol, there are a number of flaws. First, PGP tends to allow all combinations of usages of encryption and signatures based on the preference of the user, but does not offer authentication of the headers (i.e. the "to" and "from" fields), allowing messages to be surreptitiously forwarded and otherwise redirected via signature stripping attacks [7]. Despite these problems being well-known, the IETF OpenPGP Working Group did not address any of these concerns, and so far has only re-convened in order to address upgrades in the underlying primitives in order to support elliptic curve cryptography and remove known-broken hash functions in fingerprint verification from the standard.[6] In general, PGP has been considered an open standard that has serious problems both in terms of security and usability, and this provoked the generation of competing technology such as Off the Record Messaging [4].

C. Off the Record Messaging

Released in 2004, "Off the Record" (OTR) messaging is a plug-in for synchronous instant messaging XMPP that features a number of radical changes in contrast to PGP [4]. On the level of security, different keys are generated per conversation when a conversation is started, so there is no long-term key material that is vulnerable to compromise. However, by virtue of this design choice OTR messaging limits itself to synchronous messaging between only two participants. Key management is much easier and verification of contacts is still encouraged via a shared secret established offline rather than key verification in PGP. Off the Record messaging also enables forward secrecy, i.e. that a key compromise cannot lead to the reading of past messages, by simply deriving a new key for every message in the conversation. It has undergone thorough academic analysis in terms of security, leading to newer versions of OTR being produced in response to various attacks [8]. Importantly messages can not be *repudiated*, i.e. it can not be proven that a message was actually sent by the sender. Furthermore, it could also not be proven that they were not tampered with, as malleable encryption was used. OTF was built as an extension to XMPP, an IETF standard that "provides a technology for the asynchronous, end-to-end exchange of structured data by means of direct, persistent XML streams among a distributed network of globally addressable, presence-aware clients and servers" that was mostly used for synchronous chat.[7] Like SMTP, XMPP does not provide any content confidentiality and so does not, by itself, count as secure messaging without OTR. OTR has application support via clients such as Adium and Pidgin that can be used on a number of platforms, and could even be used on mobile platforms via ChatSecure. While XMPP itself is standardized by the XMPP Foundation and the various versions of OTR is authoritatively described by its academic authors in a specification on their

[3]https://tools.ietf.org/html/rfc821

[4]https://tools.ietf.org/html/rfc2440

[5]https://tools.ietf.org/html/rfc2633

[6]https://datatracker.ietf.org/doc/charter-ietf-openpgp/

[7]https://xmpp.org/rfcs/rfc3920.html

webpage,[8] OTR's "current usage" is itself is clearly described by the XMPP Foundation (a small standards body devoted only to XMPP) as an XEP (XMPP Extension Protocol).[9] Therefore, although not as authoritatively standardized as PGP, XMPP+OTR is informally considered an open standard. A usability study was done on OTR that demonstrated users did not have trouble setting up keys but did not understand the offline authentication process [20]. The usability and user perception of more complex properties such as forward secrecy and repudiation were not studied in usability studies. Although academic work studied improving the authentication process [3], the combination of the restriction of OTR to synchronous messaging and the confusing authentication process – as well as underlying dependencies on the increasingly unused XML technology stack, unmaintained insecure clients, and excessive extensibility – all led to a decline in usage of OTR in the decade after its publication.

D. The Signal Protocol and beyond

As previous secure messaging around PGP and OTR started showing their age in terms of security and usability, it was not surprising that cryptographers wanted to derive new and better protocols in the wake of the Snowden revelations. Open-source developers started making strides in creating a next-generation secure messaging protocol, with the most advanced and popular protocol being the Signal Protocol used by applications such as Signal (formerly TextSecure, from whence the name of the protocol is derived) as well as WhatsApp. In brief, Signal used per-conversation key material in a similar manner to OTR, and thus unlike PGP did not force complex key management on the users. Like OTR, it maintained properties of repudiation and forward secrecy by virtue of the Axolotl Diffie-Hellman key ratchet[10] but added "future secrecy" so that messages indefinitely in the future cannot be read in the case of a key material compromise [5]. It solved the asynchronous messaging problem by virtue of allowing longer-term pre-keys managed by the Signal server, and offered group messaging implemented as point-to-point messaging. This protocol then started to attract attention from academic cryptographic community, and only minor flaws were found [11]. Although alternative approaches were developed and widely-deployed like MTProto by Telegram, these protocols developed their own cryptographic primitives and so received less attention from the academic community, although these protocols had a number of dangerous bugs [15]. With a minor variant implemented in the vastly popular WhatsApp messenger, the core Signal Protocol seems well on its way to clearly replacing the use of XMPP+OTR and becoming a competitive, if somewhat boutique, feature for mainstream messaging services (as shown by the adoption of the Signal Protocol as an optional feature by both Google Allo and Facebook Messenger). E-mail stubbornly remains unencrypted, due to a large part in problems with key management, and although there are efforts to revive encrypted e-mail such as the Google End-to-End project[11] and LEAP,[12] they have not yet been finalized or reached widespread adoption. Encrypted messaging applications like WhatsApp, Telegram, and Signal are now the default encrypted messaging application for users that consider themselves to be high-risk. Usability studies have shown that although Signal (similar to OTR) is easy to setup and use, even highly-skilled users fail to use verification correctly [19]. Currently, the Signal Protocol is centralized, as a single server mediates the setup of the protocol in most widespread deployments (Signal, WhatsApp, Google Allo, Facebook Messenger, Wire). Open-source alternatives that claim to use the Signal Protocol exist, such as centralized application Wire that uses a fork called Proteus, and decentralized projects such as XMPP-based Conversations use the Signal Protocol's double ratchet. While it seems that Signal is widely adopted and considered an improvement over both OTR and PGP, the core Signal Protocol remains officially unstandardized, even though there is an informal draft by Trevor Perrin and Moxie Marlinspike.[13] This has led to copying parts of the Signal Protocol by a draft XMPP Foundation standard called OMEMO for use by applications such as Conversations.[14] There are a number of critiques as well based on privacy: the Signal Protocol provides confidentiality but requires exposing phone numbers to the server and so allows the server (although the server of Signal currently minimizes logs) or a passive adversary to capture all the metadata, including the social graph of users. While some secure messaging solutions like Ricochet use Tor to hide the IP address of their users, none of the popular Signal Protocol-based messengers hide metadata.[15]

IV. Methodology

A. Science and Technology Studies

We combine the qualitative methodology of Science and Technology Studies (STS) to analyze the interfaces of messaging apps as "meeting points" between the intentional goals of developers and the needs of users [18]. In complement approaches such as traditional quantitative survey-based or protocol-based security usability studies on particular software, STS aim at providing a fieldwork-driven sense-making of emerging systems, artifacts, communities of practice, doing 'analytical thick descriptions' of events, artifacts, organizations – in particular, moments of crises, debates, controversies – to try and understand the life of a technical artifact, from its creation to its appropriation and reconfigurations by users, to its becoming a subject of public debate, of governance, of lobbying. A commonly-found term to describe this in STS literature is "problematization." Although it is clear that the field of secure messaging is riddled with problems, does not mean the production of new theoretical problems that would be possibly irrelevant to developers and users. Instead, problematization refers to the process of inquiring into "how and why certain things (behavior, phenomena, processes) became a problem" [10]. The primary methodology to achieve this goal is to observe, for relatively prolonged periods of time, specific case-study groups or communities, conducting on the side in-depth interviews with their members and reading appropriate documentation such as release notes, accounts of working sessions, etc. This generally requires to carefully select a limited number of case studies, which will be covered in

[8] https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html
[9] https://xmpp.org/extensions/xep-0364.html
[10] https://github.com/trevp/double_ratchet/wiki
[11] https://github.com/e2email-org/e2email
[12] https://leap.se
[13] https://signal.org/docs/specifications/x3dh
[14] https://xmpp.org/extensions/xep-0384.html
[15] https://ricochet.im/

depth, by making hypotheses on their meaningfulness. Ideally these case studies should be representative of wider trends and be cross-checked via multiple interviews or backed up with quantitative studies. Thus, STS employs primarily qualitative techniques from anthropology such as ethnography, but aimed primarily at the role of technology in society, which can provide insight that can form the foundation for quantitative work and future development of the field.

We argue that secure messaging very much needs this perspective at the present time, as an emerging field that is increasingly becoming a matter of interest for the general public. It is a moment in time when users cannot be taken as a 'separate sample' from the rest of the ecosystem, including developers themselves (and the different forms they choose to give to their projects, their level of openness, etc.), alongside a variety of trainers, regulators, the media, and so on. At the current level of maturity of encryption as a public concern and a concern of governance, this approach is very much needed and would do well to precede with more systematic and quantitative endeavors. One possible outcome of this qualitative approach from STS is that the very concepts and schemas traditionally used in cryptography and usability themselves can globally revised if needed. Due to the persistent presence of usability and adoption problems in secure messaging, it makes sense that some global revision of the conceptual schema used by developers may be necessary in order to harmonize their goals with that of users. Given that these developers differ both wildly and in subtle ways, a quantitative survey by itself would have difficulty summarizing the intentional structures of those working on and using secure messaging. Also, the results of such a work would likely not be statistically significant given the small amount of developers. The same holds for users: If user samples vary highly due to risk level and geographical location, this biases samples rendering traditional survey approaches problematic. Instead, we hope that these interviews help crystallize the key positions of these developers and users on issues, and that then larger-scale quantitative surveys and in-person protocols with specific software can be done to see if the developer intentions actually map to user needs.

User needs are also not as simple as they appear: In the tradition of "user studies" developed within STS, we understand users not as a homogeneous and passive group, but as active contributors participating in innovation and co-shaping technologies [18], which is possible in software development via routes such as bug reporting, pull requests on code, mailing list comments, and in person contact of users with developers. We distinguish users as high-risk or low-risk, with respect being paid to their own analysis and description of their situation. Our interviews include both tech-savvy users (who become trainers and teach other users) as well as low-knowledge users who are nonetheless possibly in a very high-risk situation (i.e. a situation where the misuse of secure messaging would likely lead to death or high prison sentences). In our methodology, at first we focused on interviewing users from western Europe who were not likely in high-risk situations (in particular, in Germany, France, Austria) as well as activists and journalists from Eastern Europe and the Middle East in high-risk situations in Ukraine, Iran and Egypt. The questions being used in our study are provided in Appendix A.

B. *Interview Selection Process*

Interview subjects that were developers were selected due to pre-existing personal relationships with the cryptographic research community. Although this does provide bias, we believe it can be countered by doing a large number of interviews as well as also recognizing the relatively small size of the global developer community. We also reached to some developers via the GitLab and GitHub pages of the projects without personal connections (e.g. Ricochet, Conversations). In contrast, user studies were done with individuals that were selected more by chance via their attendance at training events in their local environments (both high-risk, in the case of Ukraine, and low-risk in the case of France and the United Kingdom) or conferences in pre-selected venues that were determined to be likely to attract high-risk users that lived in areas that, due to the level of repression, made it difficult if not impossible to interview them in their native environment, or would make it such that they could not speak openly in their native environment due to repression. This was the case for users from Egypt, Turkey, Kenya, Iran, where the interviews took place in March 2017 at the Internet Freedom Festival and at RightsCon. All interviews were made between Fall 2016 and Spring 2017, for a total of 48 interviews. We interviewed (15) developers, experts from NGOs focused on privacy and security, such as EFF, Tactical Tech and Privacy International (6) and everyday users (27), for a total of 33 user interviews. Developers from LEAP and Pixelated (PGP), ChatSecure (OTR), Signal (including Wire and Conversations (OMEMO) implementations) were interviewed, as well as developers from Briar and Ricochet that use their own custom protocols. Within user groups we distinguish between high-risk users (12) and users (including researchers and students) from low-risk countries (21). The developers were all from the USA/Western Europe, and the high-risk users included users from Ukraine, Russia, Egypt, and Iran. Some high-risk users, due to the conditions in their country, had left (4) or maintained dual residency (2) between their high-risk environment and a low-risk environment. The "users" category also includes a subset (18) of security trainers, e.g. users involved in organizing seminars on security, disseminating privacy-enhancing technologies, practices and knowledge. We interviewed between trainers from high-risk (9) and low-risk countries (9). All questions that we used, including those given only to particular categories like trainers and developers, are given in Appendix A.

C. *Ethical Guidelines*

A specific protocol was developed in order to protect privacy of our respondents. We let users and developers suggest us a tool of communication of their choice if they wish to do the interview online. These tools ranged from PGP to Signal, meet.jitsi, Wire or WhatsApp. If an "in person" interview was preferred, the interview was recorded with an audio recorder isolated from the Internet. We use a dedicated encrypted hard-drive to store the interviews. Before the interview we asked our respondents to carefully read two user-consent forms related to the study and ask all the questions regarding their privacy, their rights and our methodology. The two forms were written in collaboration with UCL usability researchers and based on the European General Data Protection Regulation. The documents included an Information Sheet and an Informed Consent Form.

The first document, (Information Sheet) explained the purpose of the interview, described the research project and clearly mentioned the sources of funding for the project; provided information on the length of the interview, but also information about the researcher, including her email, full name, academic affiliation and the address of the research institution. The second form (Informed Consent) described the procedures regarding data processing methods, the period and conditions of data storage; it emphasized the right of the interviewees to demand, at any moment, to withdraw their data from the research. A copy of each document was given to the interviewee. Different forms were used for users and developers. These forms were given as a link here.[16] Additional measures have been taken to ensure better privacy for our interviewees. Thus, the name of the interviewee was not mentioned during the recording. We also adapted some questions to withdraw any elements of context (such as the country or the city, the precise social movement or affinity group a user was involved in and so on), if interviewees asked for this. We respected the right of our interviewees to refuse answering a specific question. However, our questions were specifically designed in order to focus on the tools, with no biographical questions. In this report, user names are used when the user gave permission, but are otherwise kept anonymized.

V. Interviews

The results of the interviews are presented in this section. For each category of questions, representative quotes have been chosen. The results of the interviews are summarized on a high-level in Table II where 'low' means that the topic was mentioned as a topic the user didn't care about or mentioned negatively by a majority of those interviewed and 'high' means it was mentioned positively. Note that we did not to statistical tests as our sample size was too small, and in future research we will increase the sample size so that such testing can be done.

A. Developer Motivation

Developer motivation was quite wide-ranging, but largely could be divided between those who wanted to start privacy-enhanced businesses that would serve both low and high-risk users to those who were primarily motivated by protecting high-risk users due to human rights concerns that are more traditionally dealt with by the NGO sector. In the case of Wire, the developers felt that they were addressing issues that they had neglected in the original design of Skype, as "after Skype was sold to Microsoft [they] had an idea of how to build a new Skype...or what Skype should look like 15 years after. One of the biggest gaps that was missing on the market was related to privacy and security." Nonetheless, they had very limited contact with high-risk activists and stated that the application was developed "mainly for private usage," leading to some technical limitations such as group chat only supporting up to 128 users that made it unusable for mass social movement organizational purposes. On the other hand, although Briar has very little use from high-risk activists, the entire concept was inspired by inquires from high-risk activists asking "if LimeWire would be suitable for communication" and that although the developer of Briar (who worked at Limewire at the time) felt "that it may be suitable ... we can build something suitable on a more social basis," which in turn led to the development of Briar. Some developers, such as those of ChatSecure, are surprised by the amount of downloads from high-risk activists (approximately twice as many downloads from the Russia as from the USA), "our Russian translation is pretty good and appstore description is good, and theres an XMPP server in Russia that recommends our clients ... I don't think it explains that much downloads." As discovered via interviews, a large existing user-base of XMPP+OTR users in the Russian anti-fascist movement looking for a mobile version of their desktop clients ended up being the root cause of the popularity of ChatSecure amongst Russian activists. Strangely, it appears that developers are motivated by high-risk activists, but have little actual contact with high-risk users in their systems.

B. High-risk vs. Low-risk Users

Developers tended to distinguish between low-risk users who are "privacy-aware" and high-risk users such as human rights activists in war-zones, and further distinguish these two groups explicitly from the "high-knowledge" expert (but usually "low risk") users, e.g. researchers and tech-savvy users who install the software to test out their capabilities. The division between high-risk and low-risk users held up in the interviews. High-risk users, unlike low-risk users, focus on active attacks and have a well-defined threat model. However, low-risk users had an implicit threat-model with a focus on passive threat models, such as server seizure. High-risk users worried about active attacks ranging from device compromise to active man-in-the-middle attacks but were not certain to what extent they were protected by secure messaging applications.

Due to these difference in threat models, high-risk users often try to verify keys (after they receive a notification that the key has been changed in Signal, WhatsApp, Wire or other applications) while low-risk users with a "passive" threat model did not. High-risk users tend to check the authenticity of a person if the key material changes, but may check for authenticity informally using context rather than using only cryptographic verification: "I verify keys in PGP, but...I verify the person by other means... we speak about same things. In Jabber also I often just do it manually, without shared secret. But I always check if I receive something warnings about the persons device" (K., trainer). High-risk users are afraid that the devices of their friends have been physically accessed, stolen or otherwise taken away by powerful adversaries willing to use physical force and subterfuge to access contacts lists. Some high-risk users tend to confound device seizure with keys being changed, and do not realize that if a device was seized an adversary could continue communicating using the seized key material. Some do realize this possibility but then try to ascertain the identity of their contacts using out-of-band channels: "If I get a message from Signal for example, saying that my contacts device has changed or his fingerprints changed ... I normally try to get in touch with the person ... I need to hear the voice" (Ukraine, trainer).

As has been observed among our interviews, in more high-risk situations such as Ukraine, the choice of secure messaging application can be due to the politics of its country of origin. These high-risk activists exclude applications and

[16]http://www.ibiblio.org/hhalpin/homepage/forms.zip

Interview	Developers	Low-risk Users	High-risk Users
Number	15	18	15
Repudiation (Security)	high	low	low
Group Support	high	high	high
Metadata Collection (Privacy)	high	low	high
Decentralization	high	low	low
Standard	high	low	low
Open Licensing	high	low	low

TABLE II. IMPORTANCE OF PROPERTIES OF SECURE MESSAGING INTERVIEWS

online services that have servers on the territory of Russian Federation or Ukraine and prefer American-based services, with even trainers advocating usages of Gmail and Facebook. Similar dynamics were observed in Iran (with no adoption of GPG and strong preference for Gmail with two-factor authentication), and Egypt (where WhatsApp is popular as the United States is considered as not being part of the threat model). For example, "Iranians use Google and Gmail a lot, they do not care about NSA spying on them, they care about Iranian government not having access to the data. We use Google Drive to upload our personal photos for example. For us the entire motto of 'Use Signal, Use Tor' does not make sense. As soon as the servers are inaccessible for the [Iranian] government, it can be used" (M., female, Iranian high-risk user and journalist). This is similar to the response given by high-risk Ukrainian and Russian users, although they note the configuration of trusted jurisdictions changes: "The most important thing for us is to convince people to stop using Russian services like mail.ru or yandex.ru. It is a direct backdoor to FSB office. We recommend also to switch from Telegram to WhatsApp. We recommend Gmail with two-factor authentication over PGP. It is easier to explain and people are already used to the interface [...] I don't think US will give out data on Ukrainians to Russians. [...] However, after Trump everything may change" (V., trainer, Ukraine). The high-risk users interviewed so far had similar well-conceived threat models but these could easily vary in a country-specific manner in terms of application preference.

C. Security Properties

The main advantage of OTR and Signal-style protocols is that key management is no longer a barrier to entry, and this appeals even to high-risk users. Trainers often found it too difficult to teach even high-risk users the precise security properties of key management, noting that "some trainers think there should be no key discovery at all, it is better to have opportunistic or automatic key discovery as it is happening with Signal. Different encrypted messaging apps have popped up that made it a lot easier to have just an app that will pass on your communication, and the encryption part will be transparent to the user. Having encryption as a default mode is the key part in making encryption popular." The vast majority of users preferred not having to do key management. Yet many high-risk users wanted some form of key verification and out-of-band authentication, even if it was hard to use in practice. Both high-risk and low-risk users insisted on the importance to "see encryption happening" in the interface, a desire also explored in earlier work on encrypting Facebook conversations [9].

All secure messaging applications offered confidentiality and integrity. The most controversial of security properties is authentication, in particular the idea of repudiation. The security of end-to-end encryption definitely includes confidentiality, but often surprisingly (post-PGP) protocols like Signal and OTR are defined not to include authentication using traditional cryptographic signatures, but instead include some other authentication code or shared secret. Repudiation is when it can not be proven that any given message *cryptographically* came from a particular user or by a unknown third-party. OTR even went as far as to use malleable encryption to enforce this property. Yet it is unclear if such a cryptographic technical construction would ever lead to plea of plausible deniability being socially accepted in court. This may lead to unnecessary complexity in protocols like OTR and Signal. In order to achieve plausible deniablility and "future secrecy," the use of Diffie-Hellman key ratchets leads to a plethora of keys, and this makes the protocol more difficult to understand and formally verify than a protocol that would be based on simpler mechanisms such as non-repudiable signatures [5]. Users were not aware of any cases where cryptographic deniability was used in actual court cases, as usually the social context and other circumstantial evidence was enough to determine if a user was involved in a conversation. Ukrainian users mentioned social networks (namely, Vkontakte and Facebook) as the main source for deanonymizing users and their connections and insisted on a need of changing privacy settings to close friend lists. For all high-risk users device seizures were mentioned as more important and more frequent threats than "man in the middle" (MiTM) attacks. It was generally viewed that ephemeral messages as such were more important than cryptographic deniability. In order to achieve "real-world" repudiation and deniability users tend to develop their own practices and combine usage of several tools. For example, Russian high-risk claimed to use so-called "one-time secrets" to share information via accessing specific websites via Tor as this offered a possibility to send and receive unique self-destroying messages (similar to, but not using, Ricochet). Therefore, it seems repudiation via cryptographic deniability ends up being an interesting design choice, but one much less important than preventing metadata collection.

D. Group Support

Developers of secure messaging apps perceive group support as one of the main challenges, and both high-risk and low-risk users we interviewed wanted some form of group chat. Developing group chats in peer-to-peer systems "is both an important usability feature and a scientific problem," a developer pointed out. Among the questions developers are working on in relation to the group chat are: Who has the right to invite and ban participants of a group chat? While Signal, Wire and Telegram offer a possibility to all members to invite new participants, Briar suggests a specific architecture for the group chat. Briar lead developer Michael, compared the group chat architecture of Briar with a "star": For a better metadata protection and anonymity, only the creator of the group has a right to invite and ban participants. "It leads to a certain centralization within a distributed system" remarked another Briar developer. Another open research and practical challenge is hiding the social graph of participants of the group chat. Projects we studied propose different solutions to this problem: For example in Briar every new member of a group chat is only connected with the creator of the group, but the links between members of the group do not exist, preserving users' contact networks. In contrast, Wire opted for a solution to analyze contact networks of users and, based on this analysis, to suggest new contacts to users. This function was criticized by both trainers and the security community as revealing metadata, which would be dangerous to high-risk users. Both high-risk and low-risk users also used different applications for different group purposes, using group chat on Facebook Messenger for their work, while preferring WhatsApp or Signal group chat for communications that they considered private.

High-risk users in Ukraine emphasized their usage of Cryptocat group chats during Maidan revolution, thanks to the relative anonymity (understood by them as absence of any connection a telephone number) that would not reveal the metadata of high-risk users in a group chat. As security trainers point out, Telegram group chats are also popular among high-risk users despite the fact that encryption for group chat offered in Telegram is very basic, defaulting to simple TLS rather than the more advanced M-PROTO protocol for group chat. We've observed several groups of activists and researchers working in Russia and Ukraine in a high-risk context (namely covering the events in the east of Ukraine) that trusted Telegram group chats over their secret group communications. Trainers explain the popularity of Telegram because of the self-representation of the app on the market: "They [Telegram] managed to present themselves as a secure messaging app, from the very beginning. It's like their main selling argument [...] People believe Durov [Telegram founder] because he left Russia" [V., trainer, Ukraine]. Thus, motivations for adoption of privacy-enhancing tools are also dependent on the reputation of their creators, as well as shifting geo-political alliances that may effect the reach of government agencies. The fear of Google and the NSA storing large amounts of data in the long-term is viewed as more of a problem by low-risk users, as this problem does not have immediate consequences for many high-risk users.

E. Privacy Properties

Many developers found increasing privacy through minimizing metadata collection to be the second most important feature after the development of end-to-end encryption: "End-to-end encryption is a first step. But besides there is a whole new dynamics that needs to happen that's related to all of the metadata and what is it used for." Yet many developers confused whether or not a third-party adversary could be passively monitoring the communication and so collect metadata with whether or not they as developers personally collected data, as exemplified by one developer that stated simply "I do not have anything in my code that would let me know how many people are watching the app right now." However, many developers also believed they would have to collect some metadata in order to interoperate with features such as push notifications of arriving messages, but they try to limit the harm: "With introducing the push messaging it's the first time were exposed to possible metadata. But we don't log anything, we don't know who is talking to who, we don't log any information." Most developers who were aware of third-party data collection of metadata were supportive of using Tor and on disabling the collection of phone numbers in particular, but lacked a comprehensive plan to minimize the collection of data as such. High-risk and low-risk users generally supported reducing data collection and increasing privacy, although often the encryption of data was assumed to hide metadata by non-trained users.

Developers and information security trainers underlined the urgency to find a reliable solution to the metadata collection problem and state that nothing in the field of end-to-end encrypted instant messaging apps offers good metadata protection: "Metadata connects you weirdly with other people, and there's more sense in the metadata than in the data itself for technological reasons [...] No one from the messaging apps is trying to solve that. Instead they suggest to sync your address books so they know exactly who you're talking to even though you trust them to somehow make it into hashes or whatever. That's the issue we are not solving with the apps, we make it worse. We now have centralized servers that become honeypots, and it's not about the data, it's about the metadata" (Peter S., Heml.is). Developers and trainers associated the leaking of metadata with centralization.

F. Decentralization

While centralized projects such as Telegram, Signal, or Wire prevail on the market and have larger user-base, most developers were more enthusiastic about decentralization. Even though they agree on the fact that decentralized systems are harder to design, their motivation to work on decentralized systems was grounded in both the political and technical aspects of decentralization. Politically, decentralization offers 'empowerment' to the user, as it gives users a means to 'control' their own data and developers believe it enables better metadata protection: "You're still not owning your data, all the metadata is controlled by a centralized system, they know all your contacts, who youre messaging at what time. I want people to run their own infrastructure." Some developers believed the choice of decentralization is inherently connected to not collecting metadata, and felt that models existed which were usable and decentralized: "With Signal it's impossible to create a decentralized system because phone numbers aren't decentralized. With XMPP it's like an email address. Even users who aren't technologically savvy can understand this is my user ID, and this is my server." Developers involved into production of decentralized protocols noticed that the market reality of secure messaging makes both federated and distributed projects less privileged in terms of financial investments than centralized projects: "It is more challenging to build federated systems because you have to coordinate with other implementers, but also the real problem is the funding! People work on XMPP clients in their free time, so it is not as perfect as a centralized system with proper funding."

Unlike developers, many high-risk users did not bring up the need for decentralization explicitly, but they brought it up implicitly in how they formed trust relationships. Decentralization is seen both as technical challenge and social experiment, as it provides infrastructure for specific communities to organize with less dependency on intermediaries. In this sense, developers, high-risk users, and trainers tend to build associations between political organization and technical infrastructure. For example, some developers and trainers justified decentralization as mirroring the organization of anti-authoritarian social movements. In terms of choice, there was a preference for systems that were considered trustworthy politically by high-risk users, and decentralization was generally viewed as a positive in this regard by the minority of high-risk users that wanted decentralization. These high-risk users expressed concerns about centralized systems collecting their metadata, although few realized this would be possible in most decentralized systems as well, albeit in a distributed form.

G. Standardization

Users tended not to care about standards and the topic was rarely mentioned or brought up, including by high-risk users. In stark contrast, developers care deeply about standards, as they felt standards were "something they would eventually be working on." Yet there was widespread discontent with existing standards bodies, as the "XMPP community is very conservative" and "the IETF is not the same beast it was in the early days." Instead, most developers shared the philosophy that they would build the application first, and then focus on standardization and decentralization via the use of open standards. In the case of secure messaging, it was still felt that more development was needed on the code, and standardization would only slow down existing development efforts. Developers adopted Axolotl (the Diffie-Hellman ratchet) because they felt it was the best design available, even if it was not fully standardized and they had to re-code it from scratch.

Tensions between centralization and decentralization go hand-in-hand with debates over standards in online debates within developer community. A well-known argument in favor of centralization and against standards was published by Moxie Marlinspike (Signal Developer) in his blog.[17] This blog-post called "The eco-system is moving" has attracted considerable attention is widely quoted by developers as a reason not to use standards, as centralization offers a better control while federation can be "dangerous" in terms of security, as it is hard to audit all the different implementations of the protocol and ensure correct updates. Developers from the PGP, XMPP, and other protocols (Briar, Richochet, etc.) strongly oppose to this critique from Signal in their own blog-posts.[18] For example, one XMPP developer working on encryption states that the "extensibility of XMPP is not a danger in itself. A good extension will spread naturally. Moreover, there's a permanent incentive to innovate in XMPP." This has led developers in certain communities to try to standardize a version of the Signal Protocol, the OMEMO standard, in the XMPP Foundation. Signal developers are concerned about the technical competence of having third-party developers standardize their protocol, as well the likely situation where the standard is not be updated rapidly enough in response to research and bugs. In terms of PGP, developers from encrypted e-mail providers such as LEAP are still trying to implement the PGP standards correctly, and hope in the future that the PGP standards can be extended to have properties such as future secrecy and easier key management like Signal, but they see fixing the standards as a far-off long-term goal. In contrast, Signal developers believe these older protocols like PGP and XMPP actually harm user security and should be abandoned.

[17]https://whispersystems.org/blog/the-ecosystem-is-moving/

[18]https://gultsch.de/objection.html

H. Licensing

Viewpoints on licensing varied between developers and users, although most preferred open-source licensing, albeit for different reasons. GPL usage was viewed as a lifestyle choice by low-risk users: "If I dont like mainstream in media, if I dont like mainstream in music – why would I like mainstream on my computer?" (Austrian user). High-risk users were concerned over metadata leaking by having to pay for applications and many did not have funds to purchase proprietary applications, while some low-risk users preferred closed-source commercial platforms like Threema with a commitment to privacy and were happy to have to pay for services. Developers found the GPL frustrating in terms of the Signal Protocol and it's lack of a standard, as it prevented the integration of their application with platforms like the Apple Appstore, and hoped to use the same basic protocol but under a more permissive license. As stated by a Wire developer, "The Signal Protocol is open source under the GPL that means you can't integrate it into a commercial product and thats why Whisper Systems were getting large licensing agreements from Facebook and Google and WhatsApp to integrate without opening up all of their source code."

High-risk users tended to understand that open-source was necessary for trust, as their friends who were security trainers said this to them: "All security experts whom I trust all use Signal, and we must use something that is secure and easy-going and that we can use all together so we decided to use that and we just hope it is safe. I think they looked at the source code, I did not but I have to trust them." Yet high-risk trainers recognized that an easy-to-use interface with cutting-edge features (including new emojis) mattered: "You can say OK we verified this application, it's legit, it does what it says. But the user interface part is key in reaching the audience. Features, looking nice, be easy to use. This is what you need to have success with users." Rather than look at code themselves, high-risk relied on 'word-of-mouth' from other high-risk users in terms of code quality.

VI. Conclusions

In order to design protocols appropriately, the intentions and needs of users need to be brought into greater co-ordination with that of developers. However, this is not as trivial as there are distinct classes of users, ranging from high-risk and low-risk users. In addition, a subset of both high-risk and low-risk users end up being trainers that train other users. Although we did not have time to thoroughly explore the entire space of possible levels of risks and levels of expertise as our interview process is still underway, it does appear high-risk and low-risk users have very different threat models, with high-risk users generally being concerned over physical device compromise and other active attacks with low-risk users being more concerned over passive attacks and server-seizures. High-risk users defined their threat model against a local active adversary, often the police or secret agencies of their government or a nearby hostile government, rather than a global passive adversary such as the NSA. In contrast, developers usually view their threat model as the NSA, a global powerful adversary, despite the lack of attention to privacy properties like metadata collection in secure messaging protocols. While developers created their applications for high-risk users, they were in general concerned mostly with the particulars of messaging protocols, and not threats such as device seizures, although the move to ephemeral messaging in Signal shows growing awareness of this threat model.

As for our first initial thesis (Developer-user disconnect), that the properties of protocols, particularly in terms of cutting-edge design choices, are not understood by users is correct but subtle. Users do want confidentiality and group messaging. Yet in other properties there is a disconnect, as users also want privacy protection, do not need repudiation, and do not care about decentralization and licensing. While key management is important to keep simple, no users needed cryptographic repudiability of a conversation. Instead, they are more interested in privacy and even anonymization to resist metadata collection. Still, the problems are subtle in points where the application does not line up with user expectations. For example, users often believe Signal protects metadata and keeps their conversations anonymous. For example, even though Signal does not log metadata (outside the last time a user installed the application and the time of last connection), a real-world adversary could simply watch the encrypted messages going in and out of the Signal server in order to capture the social graph of users. More easily, a captured device would reveal the phone numbers of all other Signal contacts. Although some applications such as Ricochet do achieve protection against this kind of adversary that high-risk users worry about, high-risk users are in general much more aware of Signal and find it easy to use, while the anonymized Ricochet application was unknown amongst the users we interviewed. Some issues that are important for developers, such as standards or decentralization, are not in general important to users. Licensing (and having code open-source) is equally important to developers. Low-risk users tend not to care, but high-risk users do prefer open-source code, although they do not inspect it themselves.

In terms of our second initial thesis (High-Risk User Problem), high-risk users have different properties they want and behavior than low-risk users. High-risk users have well-defined (if implicit) threat models, prefer open-source, are concerned over device seizure, and are concerned over privacy, including not just metadata collection but having phone numbers from secure messaging applications such as Signal being leaked or captured. Therefore, we are left with the curious situation of high-risk users in Ukraine preferring Cryptocat, which suffered from serious security vulnerabilities in early versions but did not require phone numbers like Signal. However, high-risk users are not homogeneous, as the social and geopolitical differences between high-risk users lead to vastly different eco-systems of applications. Therefore, for future work we need to explore the differences between different high-risk groups of users across a wider variety of countries to see what generalizable patterns emerge. Note that the same likely holds for low-risk users, as low-risk technological enthusiasts are likely very different than business users, and further work needs to be done studying the variety of low-risk users as well.

As for the third thesis (Security Trainings Differ by Risk), training formats vary due to local context, and the geopolitical dimension will play an important part for trainers' choice of tools. Trainings in low-risk and high-risk contexts will have different organizational formats. While low-risk trainings devote time to focus on privacy (e.g. suggesting privacy-respecting search-engines, open-source and decentralized alternatives to mainstream services, and focus on ad blocking), high-risk trainings are more security-oriented and include a long component on operational security in addition to secure messaging: Two-factor authentication, passphrase requirements, password managers, phishing attacks, and hard-disk encryption.

Open-source and licensing choices are less covered in high-risk trainings, as high-risk users do not always associate open-source with security. Open-source is seen as a less important criteria in the context of an immediate physical threat, as when a proprietary but "efficient" and "easy to explain" solution exists, trainers will give priority to it. For example, in Ukraine WhatsApp is the most recommended application, because it is considered to be easy to install. Trainers consider WhatsApp's proprietary license and collaboration with Facebook in terms of metadata less important than the immediate security it can give to the users. The primary task in high-risk contexts with low-knowledge users is to help them quickly abandon unencrypted tools as well as tools that collaborate with their adversaries.

In contrast, low-risk trainings address the topic of decentralization and technical choices of architectures in general, training users in Signal but also Jabber and PGP. High-risk trainers pointed out to the inherent complexity of decentralized solutions, and their "experimental" character that makes them "unreliable" in high-risk contexts. High-risk trainers who recommend PGP and Jabber (XMPP) are not hopeful they will be used. Low-risk trainers eschew more proprietary solutions such as WhatsApp and VPNs, preferring instead that people use open source tools like Tor and Signal.

For future work, we plan to at least double the number of interviews, including more high-risk users in different locations (such as the Middle East and South America, as most of the high-risk users in this study is concentrated in Ukraine and Russia). We will continue to test to see how various security and privacy properties such as the collection of geolocation via IP addresses, deniability, and various post-compromise security situations around forward secrecy lead users to react, and therefore if user beliefs align with the reality of the protocol and its implementation. Further interviews will focus on the history of protocol design choices by developers, the authenticity of software updates, and transcript consistency in group messaging. The number of application developers will increase in particular, and we will focus more on high-risk users from areas outside Ukraine and Russia like the Middle East, and have more low-risk and non-trainer users to assure enough contrast with high-risk users and trainers to achieve possibly clear statistical significance in inter-group differences. We hope to further develop more work around the social construction of trust in secure messaging, looking at the possible biases of trainers, as well as the influence of social pressure and social media on adoption of specific tools (e.g. influence of press articles regarding various secure IM and email clients on the usages of these tools). The next "post e-mail" protocol for secure messaging needs to be aligned with real high-risk user needs, and cryptographic protocols need to be designed with real-world threat models.

Addressing this disconnect will require both more communication between developers and users, as well as more protocol development to reflect user expectations. One developer, Elijah from LEAP, put it this way: "My hope for the future is that the people, the users, humans imagine the same thing that the machines are imagining. I don't want people to dream in binary, but what I mean is that trust relationships that we are imagining match the actual trust relationship that actually exist. [...] When people are installing the app they are trusting their device, and I want to make it clear on what things they are trusting their providers, and on what things they're trusting people who make the application. When people are downloading the app, I want to foreground these issues, to make it clear. That is my goal." However, most developers are still far from this goal, and much work has to be done to make explicit the threat models of users, in particular high-risk users, to developers, and so to adequately prioritize features for development for secure messaging.

Acknowledgment

This work was funded by the European Unions Horizon 2020 Framework Programme for Research and Innovation (H2020-ICT-2015, ICT-10-2015) under grant agreement number 688722.

References

[1] Yasemin Acar, Sascha Fahl, and Michelle L Mazurek. You are not your developer, either: A research agenda for usable security and privacy research beyond end users. In *Cybersecurity Development (SecDev), IEEE*, pages 3–8. IEEE, 2016.

[2] Anne Adams and Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[3] Chris Alexander and Ian Goldberg. Improved user authentication in Off-The-Record messaging. In *Proceedings of the Workshop on Privacy in Electronic Society*, pages 41–47. ACM, 2007.

[4] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-The-Record communication, or, why not to use PGP. In *Proceedings of the Workshop on Privacy in the Electronic Society*, pages 77–84. ACM, 2004.

[5] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. On post-compromise security. In *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*, pages 164–178. IEEE, 2016.

[6] Harry Collins and Robert Evans. The third wave of science studies: Studies of expertise and experience. *Social Studies of Science*, 32(2):235–296, 2002.

[7] Don Davis. Defective Sign & Encrypt in S/MIME, PKCS# 7, MOSS, PEM, PGP, and XML. In *USENIX Annual Technical Conference*, pages 65–78, 2001.

[8] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Secure off-the-record messaging. In *Proceedings of the Workshop on Privacy in the Electronic Society*, pages 81–89. ACM, 2005.

[9] Sascha Fahl, Marian Harbach, Thomas Muders, Matthew Smith, and Uwe Sander. Helping Johnny 2.0 to encrypt his Facebook conversations. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 11. ACM, 2012.

[10] M. Foucault and J. Pearson. *Fearless Speech*. Semiotexte, distributed by MIT Press, Cambridge, MA, 2001.

[11] Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz. How secure is TextSecure? In *European Symposium on Security and Privacy (EuroS&P)*, pages 457–472. IEEE, 2016.

[12] Peter Galison. *Image and logic: A material culture of microphysics*. The University of Chicago Press, Chicago, United States, 1997.

[13] Simson L Garfinkel and Robert C Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook express. In *Proceedings of the Symposium on Usable Privacy and Security*, pages 13–24. ACM, 2005.

[14] Matthew Green and Matthew Smith. Developers are Not the Enemy!: The Need for Usable Security APIs. *IEEE Security & Privacy*, 14(5):40–46, 2016.

[15] Jakob Jakobsen and Claudio Orlandi. On the CCA (in)security of MTProto. In *Proceedings of the Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 113–116. ACM, 2016.

[16] Audun Jøsang and Gunnar Sanderud. Security in mobile communications: Challenges and opportunities. In *Proceedings of the Australasian Information Security Workshop*, pages 43–48. Australian Computer Society, Inc., 2003.

[17] Morgan Meyer. The rise of the knowledge broker. *Science Communication*, 31(1):118–127, 2010.

[18] Nelly Oudshoorn and Trevor Pinch. *How users matter: The co-construction of users and technology*. MIT Press, Cambridge, United States, 2005.

[19] Svenja Schröder, Markus Huber, David Wind, and Christoph Rottermanner. When Signal hits the Fan: On the Usability and Security of State-of-the-Art Secure Mobile Messaging. In *European Workshop on Usable Security*. IEEE, 2016.

[20] Ryan Stedman, Kayo Yoshida, and Ian Goldberg. A user study of Off-The-Record messaging. In *Proceedings of the Symposium on Usable Privacy and Security*, pages 95–104. ACM, 2008.

[21] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure Messaging. In *IEEE Symposium on Security and Privacy (SP)*, pages 232–249. IEEE, 2015.

[22] Alma Whitten and J Doug Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Usenix Security*, 1999.

[23] Mary Ellen Zurko and Richard Simon. User-centered security. In *Proceedings of the Workshop on New Security Paradigms*, pages 27–33. ACM, 1996.

VII. Appendix A: Questionnaire

All interviews were semi-structured, with open questions. In every case, a set of context-specific questions may also be added. For developers, based on our preliminary reading of the documentation and analysis of the website and UI of the tool, we formulated specific questions regarding the tool. For users, country-specific questions were added. The questions asked to all those interviewed were:

- How have you chosen the name?
- How have you come up with the technical solution?
- Were you inspired by some other projects or was it created from scratch?
- Do you have a threat model? List of properties?
- How do you come to decide what components of the software secured using a cryptographic or privacy-enhancing protocol and whats not?
- How do you come to decide what kinds of user data you need to store or use?
- Have you chosen centralized architecture, and are you thinking of moving towards decentralization?

- Do you support the transfer of large files?
- Do you support repudiation? Do you let users archive or search their messages?
- What kinds of groups does your protocol support?
- What kinds of metadata do you collect, and why? Do you use tools (ranging from programming languages and cryptographic libraries to development environments) in the same field as the one you are developing? Which ones and why?
- Every developer is also a user. You, as a user, what kind of difficulties do you experience with technologies you depend on, for example, with cryptographic libraries?

We also ask a number of social questions to developers:

- How many people are in your team?
- How do you share responsibilities and tasks?
- Whos allowed to make changes?
- In addition to software development, is there an operational component to your work that includes security (such as hosting servers)?
- Whats your choice of licensing? Is the protocol you use standardized, working towards a standardization or do you prefer not to standardize the protocol?
- How do you sustain yourself financially?
- What is your business model of running the any infrastructure, such as servers?
- Which other projects from the field are you collaborating with?
- How do you communicate with these projects?
- Has your protocols ever been reimplemented by other projects?
- What is your opinion on the existing academic work in the field?
- Do you collaborate with researchers?
- If you use research, what do you read? Blogs, papers?
- What conferences or gatherings do you organize or convene with developers and/or users in the field?
- How do you explain your politics (such as data collection) to your users?
- Do you get in touch with your users or information security trainers? If yes, how do you gather feedback? Do you know who your users are?

For users, the following questions were asked:

- Can you tell us about the moment when you installed your first encrypting tool?
- What was this tool? Why have you decided to use it?
- Were you satisfied with this tool? Has it helped?
- Did you install it by yourself or did someone helped you?
- Was it easy to use? Since then, which other privacy-enhancing technologies have you tried, if any?
- If you stopped using some of these tools, can you explain when and why have you abandoned them?
- Have you tried PGP? If yes, have you installed it by yourself or has someone helped you? Was it easy to install? Where did you learn to install it?
- What is your "privacy kit" for today? Describe it, of which tools it consists?
- Why have you chosen these tools?
- How do you use it in your profession/activism?
- Can you define "who is your enemy"?
- What would happen to you if your enemy got your messages?
- Do you worry about any data these tools store, and what data?
- Do you know if these tools store your list of contacts on their servers? Do you worry these servers could be monitored, or seized?
- What do you worry about more, your device being seized or the server?
- Do you want the ability to be able to move your data between servers?
- Are you more concerned over your old messages being read or new messages being read?
- Do you want to search through or archive your old messages?
- How often do you send large files as attachments?
- Do you want your messages to disappear? Do you know if they disappear on your device or on the server?
- What features are missing on secure messaging application?

Also, there is an evaluation of frequency of use of secure messaging as well as (GPG or S/MIME) encrypted mail.

A number of questions attempting to evaluate the understanding of users around cryptography:

- Do you know what an end-to-end encryption?
- How could you explain it?
- What is a key, what is a private and a public key?
- How do you usually get someones public key?
- Does it seem to be some third party does it for you, and if so, who exactly finds the other users for you?
- If you find another person's key, do you do it in person or searching on a server or do you mail them your key

(and if so, through a different application or the same one you are using? Do you verify keys?

- What do you do when your software tells you something is wrong with a key?
- What is, according to you, the most secure and trusted way to exchange and update keys?
- Do you know if [application they use] is centralized or decentralized? Does being centralized change something for you?
- Do you trust the centralized server?
- Do you trust the people behind it? What is the worse thing that could happen to them?
- What is decentralization? How can you explain it?
- Can you please make a schematic drawing to explain me how, according to you, an encrypted message goes to its destination?
- How does it get decrypted?

We tell users to, if they want, draw the scheme for the tools you use (Signal, PGP, other...)?

The current usage of users is captured as well via questions:

- Do you use encryption for all of your communications online?
- If not, what kinds of communication are unencrypted?
- Do you use secure messaging both for mobile and desktop?
- Do you also use it for audio and video?
- Do you encounter any difficulties in shifting from one support to another (mobile to desktop and vice versa)?
- If multiple secure messaging applications are used, how do you coordinate and shift among the various messengers that you have?
- What is the mail client that you use in your everyday life? Is it encrypted?
- If not, do you think you need to encrypt it and do you know solutions for doing so?
- Do you use Facebook? Do you think you need to protect your communications on Facebook? Have you tried to protect them?
- Do you use Twitter?
- Do you use encrypted cloud or storage? If not, do you think you need this kind of solution?
- Do you use any means to protect your IP address? Do you know the existing solutions?
- Do your family members use encryption? If yes, for what kinds of communication? (mails, instant messaging?)
- Do your close friends use encryption? If yes, for what kinds of communication?
- How do you contact your family members or friends who do not use encryption?
- Is it a problem for you? Have you tried to speak to your family and friends something about privacy? How did you explain the problem? Did they understand?
- in which ways does the fact of using secure messaging impact your online identity?
- Do you feel that different parts of your online identity are linked to cryptographic keys?
- If you have to, how do you manage these keys?

For trainers, a number of additional questions are asked in addition to the ones asked users:

- How did you become familiar with encryption?
- Do you believe there is a difference between security and privacy?
- Can you tell me a little bit about the first security seminar you have been to?
- Where was it held, who organized it? What did you think of it?
- What is the first event you organized or participated at as a coach? Why did you decide to organize it or participate in it?
- Who is the audience of your events? (journalists, activists, students, lay people, tech experts)
- How do you inform people about your events? Are the events public or private?
- What do you usually do at the seminar (or cryptoparty, workshop, etc.)?
- If your event turns around showing and explaining different tools, which tools do you normally promote?
- Why do you choose these tools? (What are your criteria for a “good” and a “bad” secure messaging application?
- What are your criteria for a “good” and a “bad” encrypted mail client?
- Do you have any experience of being disappointed by a tool or abandoning a tool? Why has it happened?
- Do you think that people understand your explanations of security, encryption, privacy, and the like?
- If not everyone understands, what are, according to you, the most difficult points to explain?
- How do you explain public and private keys to the public?
- How do you explain centralization and decentralization to the public?