



UNIVERSITY OF GOTHENBURG

Designing a User Interface For an End-to-end Secure Messaging System

Bachelor of Science Thesis in the Programme Software Engineering & Management

JON KRISTENSEN

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2012

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Designing a User Interface For an End-to-end Secure Messaging System

Jon Kristensen

© Jon Kristensen June 2012.

Examiner: Helena Holmström Olsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2012

Designing a User Interface For an End-to-end Secure Messaging System

Jon Kristensen

June 1, 2012

Abstract

The main goal of the report is to develop a user interface prototype for an easy-to-use, extendable, end-to-end secure, and privacy-aware web messaging application. We investigate cryptography and usability for such an application in the context of JavaScript and XMPP (Extendable Messaging and Presence Protocol), and develop a set of suitable high-level software design suggestions allowing for private communication. Based on these suggestions, we proceed to develop and test the usability of a user interface prototype.

1 Introduction

Text messaging software is an important Internet communication tool (Ollmann, 2004); basic use of online messaging includes identifying people who are online and to exchange information in near-realtime. O'Sullivan (2006) reported that 200 million people use text messaging while at work. An analysis of text messaging in the financial sector considers text messaging software found in 70% of enterprises to be a risk due to its vulnerabilities (Murphy, 2003), one of which is information leaks due to insufficient security capabilities. Meanwhile, software applications that are executed through web browsers have become popular due to the ubiquity of web browsers, the convenience of using web browsers as clients, and the possibility to update web applications without repeatedly installing software updates locally on possibly millions of computers.¹

¹http://en.wikipedia.org/wiki/Web_application. Accessed on the 26th of March, 2012.

Auguste Kerckhoffs realized already back in the 19th century that a secure communication system must be easy to use and require neither mental strain nor the knowledge of a long series of rules to observe (Kerckhoffs, 1883), yet easy-to-use, secure, and privacy-aware software solutions remains largely inaccessible today. Messaging on the web is generally centralized in its nature, and there is a strong trend of free advertisement services, which datamines and analyzes their users at the cost of their privacy; in April, 2012, Facebook reported having over 900 million active users².

In order to try to fill this gap, we have started a project aimed at creating an easy-to-use, uncentralized, free and open source software web application which enables privacy-aware Internet messaging. Since it has been argued that usability should come before security in order for a secure application to become successful (Gutmann & Grigg, 2005), we have chosen the following research question: *What would an easy-to-use user interface for a secure chat application look like?* This report provides a high-level communication and security design for one such application, investigates what implications the design has on the user interface, and offers the rationale for the construction of a user interface prototype, as well as the collected results of usability testing of the prototype.

The report is structured as follows. In the next section, we present the necessary background for the reader to understand the rest of the report. In section 3, we present the research question and method. Section 4 is concerned with discussing the different cryptographic and communicative

²<http://www.pcmag.com/article2/0,2817,2403410,00.asp>. Accessed on 18th of May, 2012.

design decisions made. Section 5 maps the design decisions to user interface requirements, while Section 6 walks the reader through the user interface rationale. Section 7 summarizes the results from the usability testing. Section 8 mentions elaborates around related works, and Section 9 concludes the report.

2 Background

This section elaborates on the technical background to help prepare the reader for the following sections. We start by presenting some background knowledge related to security, the XMPP messaging protocol, and usability.

2.1 Security

There are two main classes of modern cryptography: *symmetric-key cryptography* and *public-key cryptography* (Delfs & Knebl, 2007). Symmetric-key cryptography uses the same cryptographic key for both encryption and decryption. Essentially, the symmetric key is shared between the parties and is used to maintain a private information link. One drawback of symmetric-key cryptography is that the keys needs to be exchanged over a secure channel. Public-key cryptography (also called *asymmetric cryptography*) allows exchanging keys over an insecure medium, but leaves the problem of *Authentication*—verifying the identity of peers. Other typical security properties include *Confidentiality* (communication is not understandable to external entities), *Integrity* (any alterations to the communication must be detected by the system), and *Replay Protection* (guarantees that copies of previous communications are identifiable by the system). These properties will be used as a starting-point for our security requirements.

2.2 XMPP

XMPP (Extensible Messaging and Presence Protocol), previously called Jabber, is an decentralized open-standard communications protocol (Saint-Andre, 2011). XMPP was initially designed for instant messaging, but has since then been generalized and extended with hundreds

(possibly thousands) of extensions³. XMPP has been widely deployed across the Internet⁴. The message primitives of XMPP, *stanzas*, are *Message* (a push mechanism), *Presence* (a publish-subscribe mechanism), and *Info/Query* (a request-response mechanism).

2.3 Usability

Yee (2002) offers guidelines for usable security. Some of these guidelines are especially applicable: *The path of least resistance* (the natural way to do a task should also be the safest), *Explicit authorization* (the user needs to understand that his or her actions implies granting of authority when they do), *Visibility* (the interface should let the user easily review any active authority relationships that could affect decisions), *Revokability* (allowing the user to revoke authority, when possible), and *Clarity*.

On web usability, Krug (2000) argues that web sites should be self-evident (or at least self-explanatory), and try to minimize the cognitive workload (thought) necessary to use the application. This can be done through simple language, making buttons obviously clickable, and making the system smarter (minimizing the choices for the user). Krug also elaborates around user behaviour. It is mentioned that users generally do not read pages, rather they mostly skim through them; users tend to focus on words and phrases that seems to match the task or interest at hand, or words hardwires to our nervous systems. Also, users often *satisfice*—choosing the first reasonable option. It is stated that users generally do not care about how things work, and if they discover something that gets the job done, they stick to it. Visually, Krug recommends a clear visible hierarchy (prominancy related to how important it is, large, bold, colours, etc.), while avoiding visual noise. E.B. White’s seventeenth rule in *The Element of Style*—“Omit needless words”—is quoted; “happy talk” and unnecessary instructions should be removed. The navigation is recommended to be global and predictable (with mi-

³<http://xmpp.org/xmpp-protocols/xmpp-extensions/>. Accessed on the 13th of April, 2012.

⁴<http://xmpp.org/xsf/press/2003-09-22.shtml>. Accessed on the 13th of April, 2012.

nor exceptions), and to include a home button. A site ID or logo should be visible at all times. The home page should state identity and mission. Taglines, a welcome burb, teases, timely content, registration and sign-in information should all be available from the homepage.

On the other hand, Victor (2006) suggests a shift away from thinking of usability in terms of interaction, and instead elaborates around software as something that is used for learning (*information software*), creating (*manipulation software*) and communicating (where communication software is treated simply as a combination of information software and manipulation software). The fields of graphic design and industrial design are suggested as preferable methods for designing applications for these two classes of software. According to Victor, manipulation software should be available, understandable, and comfortable (which infers that good manipulation software should provide good visualization as well, which involves graphics design). Manipulation software displays a model; it shows not only what can be done, but also what the model is. The design of manipulation is being concluded as “unbelievably difficult”, and best avoided, if possible. On the other hand, information software is used for learning (find information or answers, compare, etc.). The essence of information software is not functionality, but presentations. Victor argues that it is not about what the software does, but what information is relevant for the user, what questions he or she will ask, and what decisions he or she is trying to make. Software is flexible in how it displays data, which creates a unique possibility for *context-sensitive* information graphics, which incorporates who the user is, and what the user wants to learn at the moment, and displays the subset of data that the user is interested in at the moment. Context can be inferred from the environment (current state of the world), history, and interaction (input from the user).

3 Research method

In order to answer our research question (*What would an easy-to-use user interface for a secure chat application look like?*), we have to identify

which kind of cryptographic and communication approaches to use (which affects the user interface) as well as designing a user interface prototype. When the prototype is finished, we will perform usability testing sessions to gather empirical and observational data which can be used to guide us through possible extensions to the user interface.

Usability testing is an iterative process (Krug, 2000). We did start with performing early usability testing with simple web mock-ups and even pen and paper sketches. The results of this testing helped to develop the prototype, and to make way for the broader, more in-depth testing that is the subject of the usability testing performed in this report. Also, we would like to note that even after the usability test is finished, we plan to perform additional tests on the modifications made. Finally, we will meet as many subjects as seems necessary, until it appears that no new issues or suggestions are reported. Refer to Section 7 for more information about on how we conducted our usability testing.

There is an overlap between our method and that of design research (Vaus, 2001), a method for creating products, services, and systems that respond to human needs. We try to generate utility value for end-users of our system by “emphatically” collecting and mapping their experiences and needs. However, as our data collection is quite limited, so is our synthesis.

4 System design

This section presents the high-level system design to be realized through the user interface we have developed. We start by presenting the general system requirements, after which two sections elaborating around the communication and security aspects of our system follows.

4.1 General requirements

The goal is to produce a secure text chat client. Moreover, we have set up the following requirements for the system:

End-to-end security No one besides the two end-users communicating (not even the server routing the messages) should be able

to read or manipulate the conversation. The communication also needs to be otherwise secure, depending on what security features found appropriate.

Extendability While the system will be a text chat client, the solution must be extendable and allow for other types of communication uses, to allow the system to expand into other areas later.

Uncentralized The system must not rely on one specific server, and users must be allowed to be spread out over a federation of servers. Malicious servers must not be able to cause harm to the rest of the network.

Cross-platform To the extent that it is possible, the solution should be made available on different operating systems, as well as smart phones. However, the first platform to be targeted is the web.

Relatively compartmentalized There are plans to expand the system into a social network with lots of additional features. The text messenger is just one of many planned features. Also, the chat functionality should be possible to use by itself.

Free and open source software The system must be (permissively) free and open source software, and should not rely on any proprietary technologies. This is because we want to encourage administrators to set up the system, and allow programmers to verify that the system is safe.

User friendly The system must be easy to use, to be able to compete for the same users as other popular messengers and social networks. End-users of the system should not have to configure any kind of services, browser settings, firewalls, and so on.

No plug-ins The system must not rely on Java, Flash, ActiveX, or any other kind of browser plug-in which would have to be installed separately. JavaScript, however, which is not only supported in all major web browsers, but also on Android, GNOME 3, Windows 8, iOS, and so on, can be assumed.

4.2 Communication

XMPP is the only protocol identified that fulfills the above mentioned requirements. Other decentralized communication protocols found are PSYC and IRC. PSYC could be ruled out quickly as it is not mature enough and lack proper web interoperability. While IRC is decentralized and offers Secure DCC for secure client-to-client communications, it requires servers to be trusted, is not extendable, does not offer a proper request-response mechanism, and lacks in web interoperability (Oikarinen, 1993). Thus, XMPP serves as the basis for communication between users of our system.

As the Transport Layer Security protocol is available in the core specification of XMPP, connections between XMPP clients and servers may easily be secured (Saint-Andre, 2011). However, the solutions available for securing the communication between two clients—end-to-end encryption—are not as accessible. General requirements for end-to-end encryption has been developed (P. Saint-Andre, 2010). Several XMPP protocol extensions has been suggested over the years for end-to-end cryptography (Saint-Andre, 2004; Muldowney, 2006; Paterson, Saint-Andre & Smith, 2007; Meyer & Saint-Andre, 2009; Miller, 2012). However, the most adopted extension seems to be a simple implementation of the Off-the-Record (abbreviated OTR) protocol (Borisov, Goldberg & Brewer, 2004). Not only is the cryptographic flexibility of version 2 of OTR is limited, but there are several other drawbacks with its current usage:

1. Only the body element of the message stanza (a subset of one of the three XMPP communication primitives) is encrypted, leaving the Info/Query and Presence messaging capabilities of XMPP completely unprotected
2. It does not work well if a user is logged into an account with multiple devices (*resources* in XMPP terminology), because the OTR session key cannot be negotiated for multiple endpoints at the same time
3. There is no namespace to, for example, announcing client support for OTR using the Service Discovery (Hildebrand, et. al., 2008)

extension, enabling XMPP clients to identify the availability of OTR without querying the receiving entity with a message

BOSH, or Bidirectional-streams Over Synchronous HTTP, enables XMPP to be used over HTTP (Paterson et al., 2010; Paterson & Saint-Andre, 2010).

We are not investigating the low-level protocol issues further, as they are unlikely to have an impact on the user interface.

4.3 Security

While researching the security aspects, we found that the future of cryptography is uncertain in many ways, especially in what concerns asymmetric cryptography. If computer performance continues to increase, chances are that current practical key lengths will become too small. Furthermore, no one knows what the future holds: The Diffie-Hellman problem (Bao, Deng & Zhu, 2003) could be solved. There could be a breakthrough in integrated circuits. A quantum computer could be realized. The only cipher that seems likely to (at least in theory) guarantee future-proof security is a one-time pad. However, one time pads has serious drawbacks for practical use: they need to be perfectly random, distributed to peers prior to use, and as long as the messages exchanged. It also has to be kept secret and properly disposed.

There are a number of research papers estimating the likely security of given key sizes (Smart et al., 2010; Lenstra, 2004; Barker et al., 2011; Orman & Hoffman, 2004; Lenstra & Verheul, 2000).⁵

Looking at the XMPP end-to-end security requirements mentioned above (Saint-Andre, 2010), we find a set of new protocol security requirements:

Trust Allows users to establish trust in each others credentials, such as (self-signed) certificates, a shared secrets, or pre-shared keys. Note that the protocol must not require a trust model that is external to the users, such as a Certificate Authority or a web-of-trust.

⁵A convenient comparison of the security provided by different key sizes can be done on <http://www.keylength.com/en/compare/>.

Perfect Forward Secrecy Communications

must not be revealed if any of the long-lived (private) keys are compromised. It must also be possible to periodically change the decryption (public) keys.

Authentication The parties must possess a credential which only they are expected to have. One example of a such credential is identity coherence through time.

Upgradability To allow for new and improved versions of the protocol, as well as new encryption algorithms.

Identity Protection The authentication credentials should not be bound to the XMPP address (“JID”), so that the peer is not limited to a specific XMPP account, and so that keys cannot be bound to a specific XMPP account.

Two other relevant properties that are mentioned is “*Usability*” (easily deployable, “no more effort than a one-time out-of-band verification of a string up to eight alphanumeric characters”) and *Efficiency* (good performance in CPU and network constrained environments).

One other security feature that is desirable in private communication is *Forgeability* (Borisov, Goldberg & Brewer, 2004). Forgeability means that an adversary gaining access to the messages cannot prove who sent the message, or that the messages has not been faked or altered. In fact, even the adversary has the power to do so. Malleable encryption enables forgeability of transcripts, as well as repudiation of contents and plausible deniability.

A lot can be learned from the Off-the-Record protocol, as invented by Borisov, Goldberg & Brewer (2004):

1. Digital signatures, an element of asymmetric cryptography, can be used to authenticate the author of a message. However, it would go against the repudiability property of our messages to sign them. Instead, the signatures are only used to establish connection between the peers. This enables long-term authentication.
2. Asymmetric cryptography (like RSA) is often used as a hybrid cryptosystem, in which case

symmetric key cryptography (such as AES) is used on top of the asymmetric layer. Establishing a shared secret in this way allows for both deniability and performance (Sun, Du & Chen, 2011). The Diffie-Hellman key exchange algorithm is an effective way that can be used to produce such a secret (Diffie & Hellman, 1976). Being cheap, the Diffie-Hellman exchange allows the peers to re-key often, which enables Perfect Forward Secrecy (as long as keys are forgotten). Re-keying also helps to protect against replay attacks, as the old keys will no longer be valid.⁶

3. By deriving Message Authentication Codes (MACs) from the shared secret (perhaps by using a one-way hash function), we can protect the integrity of the messages while at the same time allowing for forgeability.

It's important to note that, since the initial version, Off-the-Record has been extended (Goldberg, Off-the-Record Development Team, 2005) through a couple of protocol enhancements due to the discovery of certain vulnerabilities (Raimondo, Gennaro & Krawczyk, 2005; Bonneau & Morrison, 2006). The latest version, OTRv2⁷, uses the Socialist Millionaires' Protocol to verify that the routing server is not able to perform a man-in-the-middle attack, while avoiding the need to manually compare cryptographic fingerprints through an outside channel. While it does require a password/passphrase, it can be relatively simple (natural language can be used); the adversary will get exactly one guess.

It should be noted that there are standardized alternatives to these techniques with comparable requirements, such as ECIES (Shoup, 2001). Furthermore, algorithms can be chained for added security.

The report does not take the possibility of a malicious web server into consideration.

⁶Alternatively, timestamps or IDs can also be used to protect against replay attacks.

⁷<http://www.cypherpunks.ca/otr/Protocol-v2-3.1.0.html>. Accessed on the 13th of April, 2012.

5 User interface implications

Below is a discussion about different ways the above findings will affect the user interface. This is part of our contribution. We will discuss authentication, the concept of trust, and a possibly obtrusive requirement for generating entropy sources.

5.1 Authentication

XMPP supports SASL (Simple Authentication and Security Layer), to which several authentication methods are available. SCRAM-SHA-1 (Salted Challenge Response Authentication Mechanism) is the only authentication method that is mandatory-to-implement in the core specification of XMPP (Saint-Andre, 2011). It does not enable adversaries with access to authentication databases to extract the password and impersonate the user (Newman et al., 2010). However, using passwords is a trade-off between usability and security; while it is well known that users tend to choose weak passwords, forcing users to use a more secure method (such as authenticating using certificates through SASL-EXTERNAL) would likely result in a usability catastrophe.

Note that authentication is not required in all cases. SASL-ANONYMOUS can be used to allow users to communicate without authentication. (Furthermore, trace information may be used to track users between browser sessions if the server supports it (Saint-Andre, 2009), and it is found to be appropriate.) However, seeing remembering contacts is a "must have" feature for most messaging use cases.

5.2 Establishment of trust

In order to prevent the server to impersonate both users and perform a man-in-the-middle attack, some kind of out-of-band authentication unfortunately seems to be necessary. As mentioned above, one option would be to have a simple shared password that the routing server can be assumed not to be able to guess on one attempt (per authentication).

Another (more secure) way would be to manually verify the signature fingerprint. For exam-

ple, users could print their fingerprint on business cards or a similar item.

A third way would be to distribute files containing the signature file or hash. Note that HTML file uploads cannot be used, as the server should not be able to manipulate the signature/certificate. Fortunately, HTML5 provides two files API (Application Programming Interface)s that can be used to both read (Ranganathan & Sicking, 2011) and write (Uhrhane et al., 2012) local files.

5.3 Entropy sources

For added security around the random entropy sources, we want to make the random byte sequence less predictable. Cryptocat⁸ requires the user to type randomly on his or her keyboard for a couple of seconds upon establishing a session. Doing this may or may not be necessary for us later. So far in our implementation, we believe that we could unobtrusively record any mouse movement, keyboard input (release times, input characters), browser-settings, local time, and so on, while the user signs in. The Fortuna RNG algorithm (Schneier, Ferguson & Kohno, 2010) is one way to combine these sources.

5.4 Third-party APIs

Note that no external JavaScript should be allowed, as any script can override the random generator functions and make the cryptography deterministic and insecure. This prohibits us to use certain third-party functionality, such as certain APIs of semantic web applications and social networks.

6 The user interface

This sections describes how we applied the above mentioned usability guidelines and security features to our user interface. This is part of our contribution.

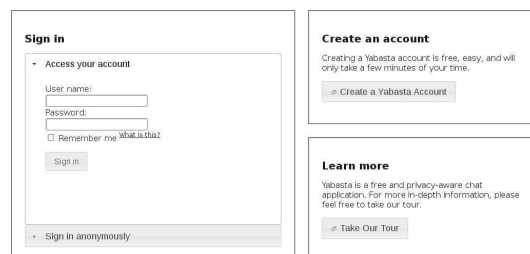
⁸<https://crypto.cat/>. Accessed the 19th of May.

6.1 Home page

At the home page (the very first page that is shown), new users might want to learn about the site or create an account. Regular users signs in, or are automatically redirected to the “inside” of the site if their session is remembered.

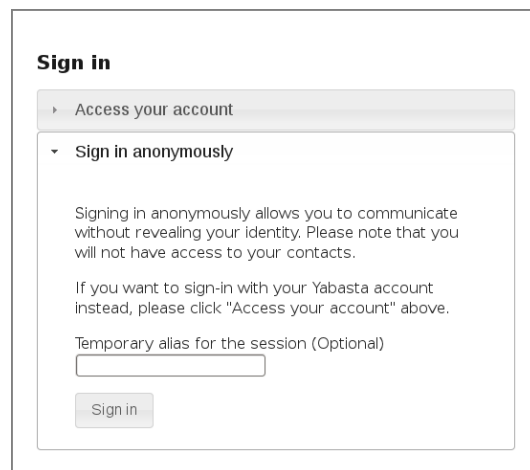
Also, as mentioned above, new users will need to be aware of the fact that their communication will never be entirely secure. We discuss this during both the tour as well as during the creation of an account.

All of these actions have been made available:



The screenshot shows two main sections. The 'Sign in' section has an accordion menu with 'Access your account' selected, containing fields for 'User name:', 'Password:', and a 'Remember me (what is this?)' checkbox, followed by a 'Sign in' button. Below it is a 'Sign in anonymously' button. The 'Create an account' section has a sub-header, a short paragraph, and a 'Create a Yabasta Account' button. The 'Learn more' section has a sub-header, a paragraph, and a 'Take Our Tour' button.

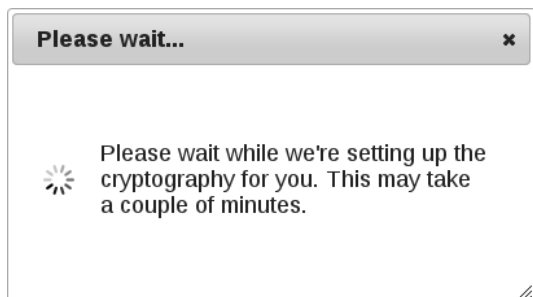
For users that wants to sign in anonymously, an accordion widget offers an alternative sign in form.



The screenshot shows the 'Sign in' section with an accordion menu where 'Sign in anonymously' is selected. The content includes a paragraph explaining that signing in anonymously allows communication without revealing identity, but users won't have access to contacts. It also offers an option to sign in with a Yabasta account. There is a text input field for a 'Temporary alias for the session (Optional)' and a 'Sign in' button.

Upon signing in, a new key pair and a certificate may need to be calculated. Since this may take a while, and users often are impatient, we provide a dialog to assure the user that the application has not stopped working. Also, we utilize

asynchronous JavaScript execution not to make it appear that the browser has crashed.



6.2 Contacts list

Looking at the contact list, there are a number of questions the user could ask:

- Who are my contacts?
- Who are on-line?
- What contacts has been trusted?

The proposed contact list answers these questions, and more.



It is alphabetically sorted so that the user easily can look up a specific contact. We show on-line contacts first, as they are most likely to be relevant to the user. Initiating a chat is as straightforward as clicking on a contact. We show a checkmark-like icon for secure contacts, and another warning triangle-like icon for insecure contacts. Hovering over the icon will display the necessary information, which is otherwise hidden from view.



We do enable a context menu of contact-specific actions, accessible through right-clicking. However, since not everyone has the possibility to right-click, and since right-click may not be an obvious possibility (since right-clicking in browsers usually has a different meaning), we have a gear icon as an alternative for accessing this context menu. As we currently mostly focus on the security features, the list of actions displayed only offers to options.

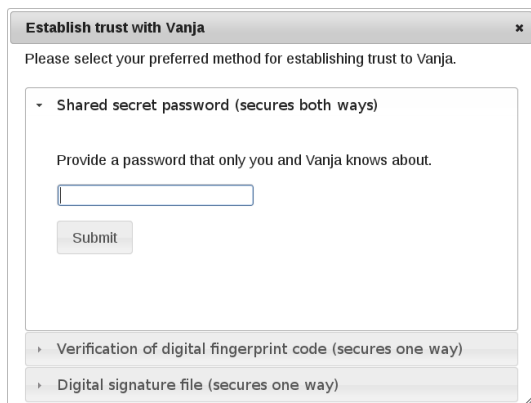


Contacts can be permanent or anonymous; in the unlikely case that the user has any anonymous contacts, they are shown first, as they are more likely to be interacted with than your regular contacts, and because they could be missed if they are put in the bottom.

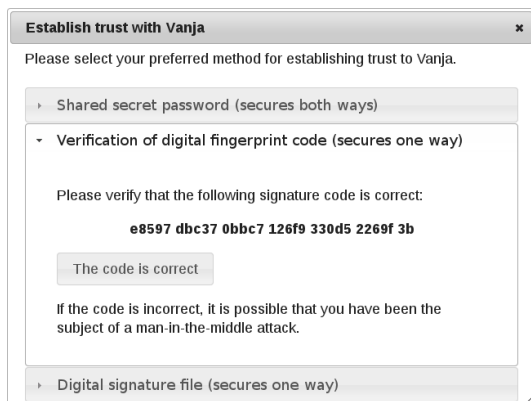
6.3 Trust

Upon selecting “Establish trust” (or as part of the process of adding a contact), the user is offered to provide a credential to establish trust to the contact. There are three options available. This time, we use an accordion that is not expanded, in order not to be biased towards one option, or to confuse the user.

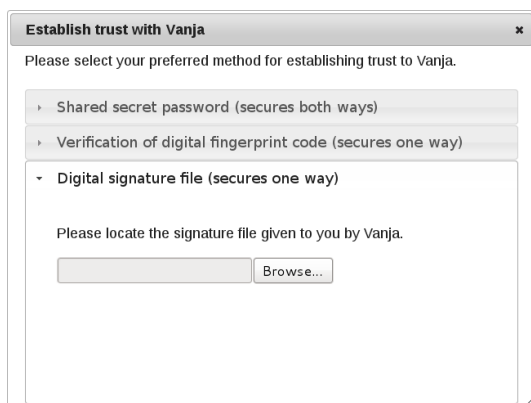
The first method, providing a shared secret password, is unique in that it authorizes the users both ways.



The second method allows the user to verify a signature code (perhaps from a business card).



The third option simply allows the user to use a local file.



The own signature code is always present at the top of the page.

7 Usability testing results

The usability testing was carried out with one user at the time being shown a user interface and asked to either figure out what it is, or to try to use it to perform the key tasks of the system. In total, four people participated in the tests, which were conducted over four separate sessions. We avoided discussing the site beforehand, and only told the users that it was an application for securely chatting. We took notes during the sessions, and tried to probe for the thoughts of the tester. We also tried out all the tests before the usability testing started. Finally, the test was followed up by a semi-structured interview where the thoughts of the interviewee on the security aspects of the application were being discussed.

The results from the four usability testing sessions follows. First off, we discuss the general layout of the page. Secondly, we discuss the testers thoughts on the overall security concept.

7.1 Layout

The home page was generally appreciated. However, the testers reported that the “Sign in” box was the first thing that they saw, being that they look at a page from left to right (as well as top to bottom). One tester said that removing the border around the box would make it more distinguishable, being the main alternative for most users. One tester would have preferred to instead have the “Sign in” box to the right. Also, the “What is Yabasta?”⁹ box was suggested to be moved to the top by two of the testers.

The “Learn more” heading did not catch many of the users attention, and one user suggested that it could be more on point, like “What is Yabasta?”. The text under the heading could be shortened, and could prepare users for the fact that trust will have to be established by saying something like “Once setup with your friends, Yabasta enables secure communication...”. Also, if the heading is renamed, the “Take Our Tour” button could be rephrased to “Learn more”, one tester suggested.

The option to sign in anonymously was considered visible enough across all the testers except for one, which thought that it could be more distin-

⁹Yabasta is the working name of the system.

guishable, perhaps by using an icon or a different colour or font. The temporary alias label was not distinguished from the rest of the text, according to two of the four testers.

There appeared to be few problems with the contact list. The icons (online status, checkmarks and warning icon) was very noticeable, and the testers, upon not understanding what the checkmark and warning icons meant, immediately hovered the mouse pointer over the icon and read (and appreciated) the tooltip text (though the online status text could have been shorter). However, some of the testers pointed out that the warning icon could instruct the user to (left-)click on it to read more about or configure the trust. Two of the testers wanted to single-click for initiating a conversation with a contact, while two users initially tried to double-click.

While the users considered the “Establish trust” dialog to clearly display three options, there were some conceptual issues (see below).

7.2 The security concept

The concept of “trust” was a little confusing to the testers. The users suspected that it was a necessity for secure communication, but would have preferred the term to be more on-point. The word “identity” seemed to be key; one user suggested labeling a trusted contact “Identity verified”; another user suggested “Identity confirmed”. One tester thought the term was confusing without having any suggestion, and one tester thought the term “trust” was sufficient.

The difference between using a shared secret password and verifying a signature code was not obvious to any of the users. “Is this the password I use to login?”, one of the testers asked. Also, three of the testers could not understand how they would know what the signature code was, and how they could determine whether or not it was correct.

Overall, there was a consensus that while our three methods of establishing trust were decent, they need more work.

One user suggested that the code, being a hex-based hash, could be made shorter by utilizing more letters of the alphabet.

8 Related works

To the best of our knowledge, nobody has developed a user interface or research around it for an end-to-end secure application with requirements like ours. However, there is two secure messaging applications that we have taken a look at: Pidgin-OTR and Cryptocat.

Pidgin-OTR is a plug-in for the free cross-platform native instant messenger Pidgin. It was usability tested by Stedman, Yoshida & Goldberg (2008). The study showed the importance of users using it correctly (also mentioned in the secure usability requirements of Yee (2002)). It also showed that there is a limitation in the shared secret method; some users stated that the shared secret method was too difficult to use, and that it might deter users from using OTR. Establishing a shared secrets also proved to be especially difficult for contacts that are not friends.

Cryptocat is a free and open source web application. Cryptocat seems to have an elaborate secure model (Kobeissi, 2012), but it does not offer permanent accounts or contact lists. It also does not offer the extendable features of XMPP, or have a permissive open source software license.

Jøsang et al. (2007) argues that, while some authors state that theoretical security does not have to be compromised if usability aspects are considered in the beginning of the system development life cycle, some security building blocks are inherently unsuitable for designing user friendly security solutions. They argue that sometimes it is necessary to invent radially new security building blocks in order to archive secure and user friendly systems. (They also list a number of useful *security action* and *security conclusion* usability principles that can be incorporated into a risk assessment process.) It can be argued that our three identity verification methods (password, signature, and certificate file) are simply not user friendly enough, and that we will need to take a step back and re-think the security of those parts of our system.

For more formal usability testing, the Heuristic Evaluation usability test method (Molich & Nielsen, 1990) can be used. It utilizes a small set of evaluators to examine a system and compare it to predefined usability principles. Another formal approach is the Cognitive Walkthrough method

(Preece et al., 1994), in which the system is analyzed and basic tasks of the user is described as (use case) scenarios. These scenarios contain what the user knows, what goals and subgoals the user has, and what his or her motivation is. We considered our system to be simple enough in order to motivate these methods.

A case study for usability in secure e-mail communication reached the conclusion that the best way (second to a face-to-face meet) to exchange a signature today (not necessarily in a couple of years) would be to do so over a phone call (Kapadia, 2007).

9 Conclusions

We started of designing a user interface prototype for private messaging web application, where our two primary quality attributes were security and usability. Our security model was designed to not only cover the features that are usually implied for secure systems, but also deniability—the impossibility for anyone to prove that a user has written any given message. (However, it should be mentioned that security is a very complicated subject, and that there no such thing as a completely secure system. Also, we do not take the possibility of a malicious web server into consideration.)

There was one particular challenge: We had problems finding a way to easily allow users of our systems to verify each others identities. Our solution consisted of three different methods: A shared secret (a password that can be relatively simple), verification of a signature string (that may be public), and usage of a certificate file.

We also found that we needed a more secure entropy (“randomness source”) for JavaScript, and that remote JavaScript file inclusion should be prohibited in order to not allow third parties to override the JavaScript randomness functions.

We moved on to conduct usability testing for this prototype, in which we received valuable feedback. The testers were somewhat satisfied with our identity verification methods, but more research is necessary to make them even easier to use. This is consistent with the conclusions of a usability study of a somewhat similar system, Pidgin-OTR.

Future research could include finding a more

usable method for contact identification, investigating how encryption keys could be backed up or moved between devices, as well as identifying a more secure (yet user-friendly) alternative to (the relatively weak) sign-in passwords. Another subject for further research could be the (low-level) design an XMPP extension dealing with deniable end-to-end communication.

References

- Barker, E., et al., 2011. *Recommendation for Key Management – Part 1*. NIST Special Publication 800-57.
- Bonneau, J., Morrison, A., 2006. *Finite-State Security Analysis of OTR Version 2*.
- Delfs, H., Knebl H., 2007. *Introduction to Cryptography—Principles and Applications*. Second Edition. Springer.
- Diffie W., Hellman, M., 1977. *New Directions in Cryptography*. IEEE Transactions on Information Theory.
- Gutmann, P., Grigg, I., 2005. *Security Usability*. CryptoCorner.
- Jøsang et al., 2007. *Security Usability Principles for Vulnerability Analysis and Risk Assessment*. 23rd Annual Computer Security Applications Conference.
- Kapadia, A., 2007. *A Case (Study) For Usability in Secure Email Communication*. Secure Systems.
- Kerckhoffs, A., 1883. *La Cryptographie Militaire*. J. des Sciences Militaires, vol. IX, Jan. 1883.
- Kobeissi, N., 2012. *Cryptocat Protocol Specification*. Version 1.4, 07/05/2012. <<https://project.cryptocat/documents/spec/spec-rev1.4.pdf>> [Accessed 18 April, 2012]
- Krug, S, 2000. *Don't Make Me Think! A Common Sense Approach to Web Usability*.

- Lenstra, A., K., 2004. *Key Lengths Contribution to The Handbook of Information Security*.
- Meyer, D., Saint-Andre, P., 2009. *XTLS: End-to-End Encryption for the Extensible Messaging and Presence Protocol (XMPP) Using Transport Layer Security (TLS)*. Network Working Group. Available at: <<http://tools.ietf.org/html/draft-meyer-xmpp-e2e-encryption-02>> [Accessed 17 April, 2012]
- Miller, M., 2012. *End-to-End Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)*. Internet Engineering Task Force. Available at <<http://tools.ietf.org/html/draft-miller-xmpp-e2e-00>> [Accessed 13 April, 2012]
- Molich, R., Nielsen, J., 1990. *Improving a Human-Computer Dialogue*. Communications of the ACM, March 1990.
- Muldowney, T., 2006. XEP-0027: *Current Jabber OpenPGP Usage*. XMPP Standards Foundation. <<http://xmpp.org/extensions/xep-0027.html>> [Accessed 17 April, 2012]
- Murphy, D., 2003. *Instant message security—Analysis of Cerulean Studios’ Trillian Application*. SANS Institute.
- Nikita, B., Goldberg, I., Brewer, S., 2004. *Off-the-Record Communication, or, Why Not To Use PGP*. WPES’04.
- Oikarinen, J., 1993. *RFC 1459: Internet Relay Chat Protocol*. Network Working Group. Available at: <<https://tools.ietf.org/html/rfc1459>>. [Accessed 21 April, 2012]
- Ollman, G., 2004. *Securing against the threat of instant messengers*. Network Security, Vol. 2004, March 2004.
- Orman, H., Hoffman, P., 2004. *RFC 3766: Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*.
- O’Sullivan, S., 2006. *Instant messaging vs. instant compromise*. Network Security, Vol. 2006, Issue 7, July 2006.
- Paterson, I., Saint-Andre, P., Smith, D., 2007. *XEP-0116: Encrypted Session Negotiation*. XMPP Standards Foundation. <<http://xmpp.org/extensions/xep-0116.html>> [Accessed 17 April, 2012]
- Preece, J. et al., 1994. *Human-Computer Interaction*. Addison-Wesley.
- Raimondo, M., D., Gennaro, R., Krawczyk, H., 2005. *Secure Off-the-Record Messaging*. WPES’05.
- Ranganathan, A., Sicking, J., 2012. *File API – W3C Editor’s Draft 7 May 2012*. <<http://dev.w3.org/2006/webapi/FileAPI/>> [Accessed 14 April, 2012]
- Saint-Andre, P., 2004. *End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)*. The XMPP Standards Foundation. Available at: <<http://xmpp.org/rfcs/rfc3923.html>> [Accessed 13 April, 2012]
- Saint-Andre, P., 2009. *XEP-0175: Best Practices for Use of SASL ANONYMOUS*. XMPP Standards Foundation. <<http://xmpp.org/extensions/xep-0175.html>> [Accessed 11 May, 2012]
- Saint-Andre, P., 2010. *Requirements for End-to-End Encryption in the Extensible Messaging and Presence Protocol (XMPP)*. Internet Engineering Task Force. Available at: <<http://tools.ietf.org/id/draft-ietf-xmpp-e2e-requirements-01.txt>> [Accessed 17 May, 2012]
- Saint-Andre, P., 2011. *RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core*. Internet Engineering Task Force. Available at: <<http://tools.ietf.org/html/rfc6120>> [Accessed 13 April, 2012]
- Schneier, B., Ferguson, N., Kohno, T., 2010. *Cryptography Engineering*. 1st ed. John Wiley & Sons.
- Shoup, V., 2001. *A Proposal for an ISO Standard for Public Key Encryption (version 2.1)*.

- Smart, N., 2011. *ECRYPT II Yearly Report on Algorithms and Keysizes (2010-2011)*. Revision 1.0 30.
- Stedman, R., Yoshida, K., Goldberg, I., 2008. *A User Study of Off-the-Record Messaging*. Symposium On Usable Privacy and Security (SOUPS) 2008, July 23–25, 2008.
- Uhrhane, E., et al., 2012. *File API: Writer*. *W3C Working Draft 17 April 2012*. <<http://www.w3.org/TR/file-writer-api/>> [Accessed 14 April, 2012]
- Vaus, D., A., 2001. *Research design in social research*. Volume 10. SAGE.
- Verheul, E., R., Lenstra, A., K., 2000. *Selecting Cryptographic Key Sizes*.
- Victor, B., 2000. *Magic Ink – Information Software and the Graphical Interface*.
- Yee, K. P., 2002. *User Interaction Design for Secure Systems*. Proc. 4th Int'l Conf. Information and Communications Security. Springer-Verlag.