

BACHELOR'S THESIS COMPUTING SCIENCE

Developing a fallback decryption method for Postguard

ESTHER SHI
S1021508

January 26, 2023

First supervisor/assessor:

Dr. Hanna Schraffenberger
h.schraffenberger@cs.ru.nl

Second assessor:

Prof. Dr. Bart Jacobs
b.jacobs@cs.ru.nl

Radboud University



Abstract

While the usage of email is ubiquitous, email encryption remains scarcely adopted for a number of reasons, which range from difficulty of key management to a negative user perception of encryption. This thesis is centered on Postguard, which offers end-to-end encryption (E2EE) for email via an add-on that is available for Thunderbird and Outlook. Postguard also provides a fallback, an alternative option to decrypt emails for those without the add-on. We extend the existing fallback by creating a functional prototype with additional features that improve usability. A usability inspection is performed on both the current and the prototype fallback. We examine the shortcomings of each fallback, and the improvements in the new fallback. Ultimately, we conclude that the new prototype fallback has useful additions, but still retains usability issues. To this end, we provide recommendations for future work that can further improve usability.

Keywords: email encryption, Postguard, usability study

Contents

1	Introduction	4
2	Preliminaries	6
2.1	A short overview of email and email encryption	6
2.1.1	Public-key cryptography	7
2.2	Why the adoption of email encryption is limited	9
2.3	Proposals for increasing email encryption adoption	14
2.4	Identity-based encryption (IBE)	15
2.5	IRMA	17
2.6	Postguard	20
2.7	Usability inspection	21
2.7.1	Cognitive walkthrough	21
2.7.2	Heuristic evaluation principles	26
3	Evaluation of the current fallback	31
3.1	Explanation of the current fallback	31
3.2	Cognitive walkthrough	35
3.3	Issues with the current fallback	37
3.4	Proposed solutions	40
3.5	Deciding on a prototype	44
4	Development of the prototype	45
4.1	Setup	45
4.2	Decryption Page	46
4.3	Email History Page	51
4.4	Settings Page	52
5	Usability inspection of the prototype	53
5.1	Usability inspection conclusions	59
6	Discussion	60
6.1	Current and new fallback	60
6.2	Limitations	61
6.3	Future work	61

7 Conclusion	62
A Cognitive walkthrough of current fallback	69
B Cognitive walkthrough of prototype fallback	75

Acknowledgements

I would like to thank Dr. Hanna Schraffenberger for her invaluable guidance, contributions, and support – without whom this thesis would not have been possible. Many thanks go to Prof. Dr. Bart Jacobs for providing his feedback and time. I would also like to extend my gratitude to Leon Botros, who was an immense help with the practical part of my thesis and contributed many suggestions. Another thank you to Onno de Gouw and Laura Kolijn for taking part in the usability inspection. Finally, thank you to Martan van der Straaten and Nadezhda Dobрева for proofreading.

Chapter 1

Introduction

Email is one of the most commonly used tools on the web. To give a sense of scale, around 4.2 billion people worldwide use email services, and 306.4 billion emails are sent out daily (Statista, 2022), 45.37% of which are spam emails (according to The Radicati Group Inc., cited in Moorthy (2022)). Despite being the universal standard for private communication, email traffic remains overwhelmingly unencrypted (Stransky et al., 2022).

To this end, Postguard¹, which is developed by a team at Radboud University, is introduced. Postguard is a tool that provides end-to-end encrypted email via an add-on, which can be installed for Thunderbird and Outlook. For people who can not or will not install the add-on, Postguard offers a fallback service for decryption. However, this substitute option has rudimentary features and is inconvenient to work with. In this thesis we develop a prototype fallback that includes more features and improves the user experience.

Our research question is as follows:

“How to provide a better usability experience for the Postguard fallback website?”

To support our research question, we have the following sub-questions:

- How good is the usability of the current fallback and where does it fall short?
- How good is the usability of the newly developed prototype fallback, and in which ways is this an improvement over the current fallback?

¹<https://postguard.eu/>

In the preliminaries, we shall first give the necessary background information for the thesis. Then, to answer our first sub-question: “How is the usability of the current fallback and where does it fall short?”, we inspect the current Postguard fallback website by applying two usability inspections: a heuristic evaluation and a cognitive walkthrough. Based on the results of these, we propose a list of features that we implement in a functional prototype. After the functional prototype is developed, we answer our second sub-question: “How is the usability of the newly developed prototype fallback, and in what ways is this an improvement over the current fallback?” using the same usability inspection methods as before. We are interested in whether our prototype actually improves the user experience. Finally, this leads us to answer our initial research question.

Chapter 2

Preliminaries

In this chapter, we present the preliminary knowledge needed for the rest of the thesis. First, we provide a short overview on the history of email and email encryption. Then, we present findings from previous research that clarify why the adoption of email encryption is so limited. Additionally, we write out the proposals for how this can be improved. Following that, we explain IRMA and Identity-Based Encryption, both of which are required as they provide the foundation for Postguard. For analyzing the Postguard fallback, we introduce two usability inspection methods.

2.1 A short overview of email and email encryption

The first email system was created in 1965 for IBM (Van Vleck, 2012). To send a message, one would use the MIT computer to leave a message for a recipient in their private ‘mail box’. This primitive system required the sender and recipient to have access to the same machine, which could be achieved through remote dial-up. Shortly afterward, in 1971, Ray Tomlinson would go on to develop email communication over ARPANET (Spicer and Tomlinson, 2016). ARPANET was a computer network developed by the United States Government for academic and research purposes. In order to send an email to a recipient, one had to denote the recipient’s username and the machine to send it to, specified in the now iconic format `username@computer`. In these early days, email was mostly in use amongst researchers and academics (Leiner et al., 1997). In an effort to streamline email, Simple Mail Transfer Protocol (SMTP) was developed in 1980. SMTP proposed specifications on the sending, receiving, and redirecting of email over servers. These specifications are published in Request for Comments (RFCs), where RFC 821¹ pertains to SMTP. Email, and the internet, garnered widespread

¹<https://www.ietf.org/rfc/rfc0821.txt>

use in the 1990s following the invention of the World Wide Web in 1989 (Roser, 2018). The concern for email encryption arose as internet use became mainstream. As a result, several implementations concerning email security were developed. Before we elaborate on these, we must first provide a short explanation on public-key cryptography.

2.1.1 Public-key cryptography

Public-key cryptography is an asymmetric key encryption system that makes use of a set of keys, a public key PK and a private key SK . Each user possesses a combination of these keys. The PK is openly shared with others while the SK remains known only to the user. Say Bob wants to send an encrypted message C to Alice. He uses Alice's public key PK_{Alice} to do this:

$$C = M + PK_{Alice}$$

If Alice wants to read Bob's encrypted message, she uses her own secret key to decrypt it:

$$M = C + SK_{Alice}$$

In practice, the process of encryption and decryption involves a complicated mathematical operation. However, for the sake of providing a simplified explanation, we depict this with the '+' operator.

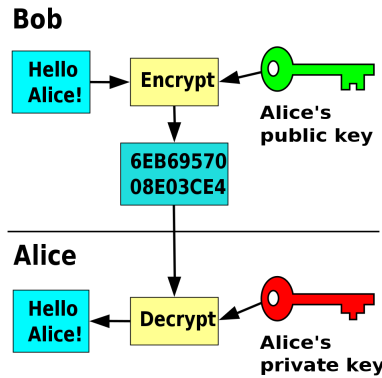


Figure 2.1: Public-key encryption scheme (from Public-key cryptography (2023))

In order to distribute the keys, public-key encryption needs a key management model. There are two distinct options for this: a **decentralized model** and a **centralized model** (Huang, 2019).

The decentralized model operates via a distributed system. One data encryption tool that uses this is Pretty Good Privacy (PGP), which operates via a concept known as the ‘Web of Trust’. In this decentralized approach, keys are verified among users, and the more verifications a key has by other users it is in contact with, the more reliable a key is regarded as. Thus, key management and the reliability that its owners are trustworthy are organized in a peer-to-peer fashion: users put their trust in the belief that their ‘network’ (composed of other users) verifies the legitimate keys. This naturally means that users need to actively extend their network to increase their collection of verified keys, this is extra work that can be seen as a disadvantage of the decentralized model.

The centralized model operates via a hierarchical system by using a concept known as ‘Certificate Authorities’ (CA). When a CA signs a key, this means that the key is legitimate and can be trusted by other users. There are multiple layers of CAs, where the trust of each CA lies in the trust of its superior CA. With this, users can verify the legitimacy of a key simply by checking the respective CA. This is the model that is used by Secure / Multipurpose Internet Mail Extensions (S/MIME). One disadvantage of the centralized model is that it is vulnerable to man-in-the-middle attacks (Huang, 2019). In a man-in-the-middle attack, a malicious actor positions themselves between two parties, both parties think they are directly communicating with each other, unaware that there is a third party listening in. In the case of a centralized model, it is also possible for a malicious actor to trick the CA into providing a rogue certificate. This is what happened with the DigiNotar incident, which resulted in significant damage (Arthur, 2017).

It must be noted that both PGP and S/MIME only encrypt the body of the email. Metadata, such as titles, senders, recipients, and dates, remain unencrypted. Both PGP and S/MIME provide **end-to-end encryption (E2EE)**, which means that data is encrypted at the source and only decrypted at the destination.

2.2 Why the adoption of email encryption is limited

Despite email being so ubiquitous, email encryption is rarely adopted. According to a study at Hannover University, of the 81 million emails they analyzed, only 0.06% were encrypted (Stransky et al., 2022). Yet, a study conducted by Reuter et al. (2021) found that 66% of the participants considered secure email as important or very important. If users are aware of the importance of encryption, then why is adoption still so low? Availability itself does not seem to be the issue, as many email clients support tools such as S/MIME and PGP, which enables the use of end-to-end encryption (E2EE) (Farrell, 2009). According to findings from multiple studies, the problems that hold people back can be summarized in the following points, which will be elaborated on below:

- Usability of key management and key migration: Whitten and Tygar (1999), Sheng et al. (2006), Garfinkel et al. (2005), Stransky et al. (2022)
- Misunderstanding of encryption principles: LaFleur and Chen (2014), Dechand et al. (2019), Renaud et al. (2014), Wu and Zappala (2018)
- Negative user perception of encryption: Dechand et al. (2019), Gaw et al. (2006), Wu and Zappala (2018), Ruoti et al. (2013), Atwater et al. (2015)
- Distrust of authority and companies in charge of encryption: Markoff (1996), Greenwald (2013), Sanger and Perlroth (2015), Pop (2022), Ritson (2018), Confessore (2018), Guild (2022), ProPublica (2021), Doffman (2021), Reporter (2018), Brody and Bergen (2019), Chin (2023), (Bai et al., 2017)

Usability of key management and key migration

The concept of key management is unintuitive to the regular computer user. In the paper ‘Why Johnny Can’t Encrypt’ by Whitten and Tygar (1999), 12 participants were tasked with using the encryption tool PGP5. It was demonstrated that getting people to understand and perform key management was difficult. Only 4 out of 12 participants were eventually able to sign and encrypt an email within the predetermined 90 minutes. Most participants failed to complete the tasks within this time. Whitten and Tygar (1999) noticed that participants had considerable difficulty with encrypting the email with the correct keys, decrypting the received emails, and one participant could not figure out how to encrypt at all. The researchers noted

that ‘failure to understand the public key model was widespread’. Participants did not properly understand the difference between a public and private key. Examples of this are participants only encrypting the email with their own public key, or sending out emails to the entire team using only one individual’s public key. Some participants also made catastrophic mistakes by sending the secret email in plain text. Another problem was that users were confused by the icons used to represent the private and public key, not understanding which represented which.

In a similar study conducted by Sheng et al. (2006) on PGP9, it was found that despite marginal improvements in key management, participants still had significant problems. Users were not able to successfully perform key verification. There was also a lack of user feedback, which denotes a usability shortcoming. For example, it was not clear to the user whether an outgoing email would be encrypted or signed. In a study by Garfinkel et al. (2005), 58% of participants reported being unaware that they would be unable to access their encrypted mail if their private key was lost.

Another problem with key management is the issue of key migration. If users want to access their encrypted emails on multiple devices, which is increasingly common in modern times, the decryption keys needed to unlock the emails must also be available on all the devices that the user has. Many people nowadays make use of multiple email clients. Because of the hassle of key migration, using different email clients decreased the likelihood of signing emails by 51.76% (Stransky et al., 2022).

Misunderstanding of encryption principles

Encryption is a foreign concept to the layman. This is apparent from multiple studies. LaFleur and Chen (2014) noted that 44% of their surveyed participants did not understand encryption, or felt no need to use it. Dechand et al. (2019) reported that most of its participants had heard of encryption, but people have misconceptions about the technical details. For example, participants believed that SMS is more secure than Whatsapp, for the sole reason that Whatsapp works via the internet (as opposed to SMS being cellular) (Dechand et al., 2019). Participants also overestimate hackers, and underestimate encryption, with the general belief that hackers are able to break it eventually (Dechand et al., 2019). In Renaud et al. (2014), participants reported to be aware of email privacy issues, but they were unaware that this is possible both on the mail server itself, and when the email is in transit.

In a study by Wu and Zappala (2018), participants were interviewed on their mental model of encryption. Wu and Zappala (2018) define a mental model as ‘the representation of your thought process of how something works’. Wu and Zappala (2018) made a number of observations.

Firstly, they observed that the mental model that people have of encryption is that of symmetric cryptography. Most participants understand encryption as using something much like a key to ‘scramble’ a piece of text, and also needing the same key to ‘descramble’ it. There were some participants who did not understand that encryption would transform the data, in turn describing that encryption was a way of hiding or obscuring a piece of information behind a barrier. These participants were largely unsure on the strength of encryption, giving widely varying estimates on how long they think it takes to break an encryption. Wu and Zappala (2018) noted that participants correlated encryption strength with the capabilities of hackers. For example, one participant thought encryption could not be that strong as she saw in popular movies that hackers could easily break into places. Wu and Zappala (2018) also noted that the existence of data breaches led participants to believe that it must not be that hard for hackers to access encrypted data.

Negative user perception of encryption

The aforementioned points have shown that misunderstanding of email encryption hampers its usage. However, this does not seem to be the only roadblock to widespread use. Studies also show that people view encryption negatively. Dechand et al. (2019) interviewed a number of employees at an anonymous company on their views of encryption. They report that, among other reasons, people do not trust encryption and do not feel protected by technology. These participants also do not feel that they are the target audience for email encryption. Additionally, while they understand in principle why encryption is useful in certain cases, such as the exchange of medical information, routine use of encryption is considered paranoid behavior (Gaw et al., 2006). Participants in another study also believed that personal use of encryption is reserved for illicit activity (Wu and Zappala, 2018). In both Dechand et al. (2019) and Gaw et al. (2006), participants feel no need to encrypt their email traffic. From these papers, participants express sentiments that external parties would not be interested in their correspondence. Even if external parties were to be reading along, participants said that they would not exchange information over email they were afraid of getting leaked. One interviewee from the paper of Gaw et al. (2006) had written in a pamphlet: “Some good advice: if you don’t want something you put in an e-mail to get into the wrong hands, don’t write it in the first place.” Encrypted emails were also seen as ‘more urgent’, prompting users to respond swiftly. Receiving encrypted emails with mundane information, such as a cooking recipe or a newsletter, is seen as unnecessary and even irritating (Gaw et al., 2006).

Email encryption as such is still seen as an ‘extra step’ in terms of caution (Gaw et al., 2006). Gaw et al. (2006) propose that ‘equating of encryption with confidentiality might disappear if encryption was invisible to the user’. However, automating and hiding the process of encryption might also result in confusion. Dechand et al. (2019) discussed Whatsapp, which has enabled end-to-end-encryption automatically for all users since 2016². In the interviews, participants reported not even being aware they were using Whatsapp’s encryption feature, claiming that they had no idea what the E2EE notification meant, or that they had to perform some action before they could ‘activate encryption’. Another paper by Ruoti et al. (2013) tested a number of participants on two email encryption systems: one which mostly obscured the process of encryption, and another which showed its encryption steps explicitly. It was found that users had greater trust in a system with manual encryption steps, and its added complexity did not hamper usability.

In Atwater et al. (2015), it was found that one third of participants trust a standalone encryption service over a service that provides its functionality via a browser extension that integrates into a mail client. Participants believe that ‘desktop applications are less likely to transmit their personal messages back to the developer of the software’ (Atwater et al., 2015). However, these participants eventually preferred the integrated service because of usability advantages.

Distrust of authority and companies in charge of encryption

There are valid reasons why there is distrust of authority and companies, especially of those based in the United States. This is because the United States and its US-based companies have garnered considerable controversy regarding data protection and privacy.

After Phil Zimmermann’s encryption software PGP was leaked to the public, he was investigated by the US federal government as his program was classified as ‘munition’. The investigation was eventually dropped (Markoff, 1996). The famous whistleblower Edward Snowden revealed in 2013 that the NSA had been surveilling US citizens without their knowledge or consent. Lavabit, the email service used by Snowden, opted to shut down instead of relinquishing its encryption keys to the US court (Greenwald, 2013). Levinson, the owner of Lavabit, issued a statement in which he said: “I would strongly recommend against anyone trusting their private data to a company with physical ties to the United States.” The National Security Agency (NSA), the US intelligence department, advocates against encryption, arguing that bad actors might use encrypted forms of communication to organize their attacks (Sanger and Perlroth, 2015).

²<https://faq.whatsapp.com/820124435853543>

Data protection laws in the United States are also more lax when compared to Europe. While Europe has adopted the GDPR, the United States tends to favor the company’s commercial interests when it comes to data regulation (Pop, 2022).

Meta and Google together are coined the ‘digital duopoly’, as combined they make up 84% of global digital media (Ritson, 2018). Both are also US-based companies. Whatsapp, which is owned by Meta, provides end-to-end encryption (E2EE) for its messaging service. To provide this feature, it acts as the central key distributor (Bai et al., 2017). Apple also provides EE2E for its messaging app³ using the same method. Acting as the central key distributor carries a risk that the companies themselves can carry out a man-in-the-middle attack on its users (Bai et al., 2017). In 2018, Facebook, also part of Meta, garnered controversy for allowing Cambridge Analytica to collect millions of users’ data without their consent (Confessore, 2018). This is only the most significant scandal in Facebook’s long list of data privacy breaches (Guild, 2022). Meta’s ownership of Whatsapp continues to receive widespread media scrutiny, with publications questioning its user monitoring process (ProPublica, 2021), and its data aggregation policies (Doffman, 2021).

In 2018, it was revealed that Google had been tracking the location data of an estimated 2 billion users even when explicitly told not to (Reporter, 2018). For this, Google paid a 392 million dollar fine. In 2019, Google’s YouTube was exposed for breaching child privacy laws (Brody and Bergen, 2019). It had been collecting data on children under the age of 13 without their parents’ consent. Google ended up paying a 170 million dollar fine.

Tangentially related, large companies are also often the target of data leaks. In a long list of history’s largest data breaches, many of the largest companies such as Yahoo, Microsoft, and Facebook, are included (Chin, 2023). One study conducted by Malwarebytes showed that distrust and privacy concerns of social media, such as Google and Facebook, were present amongst all age groups (Umawing, 2019). All of this combined evokes legitimate concern over the trustworthiness of these large companies being in charge of and managing encryption.

³<https://www.apple.com/privacy/features/>

2.3 Proposals for increasing email encryption adoption

Various studies made proposals they believe could increase the adoption of email encryption. These can be summarized as follows, and are explained below in short:

- Simplify key management: Stransky et al. (2022), Bai et al. (2017)
- Improve mental models of encryption: Dechand et al. (2019), Wu and Zappala (2018)
- Normalize encryption through widespread adoption: LaFleur and Chen (2014), Dechand et al. (2019)
- Improve usability: Smetters and Grinter (2002), Stransky et al. (2022)

Simplify key management

Since key management has proven to be a draconian effort, multiple studies propose to simplify the process. In Stransky et al. (2022), participants noted that they prefer to register their public keys on a public page and automatically retrieve keys of recipients, over the effort of manual key management. They also state that simplifying the key verification process by using a trust on first use (TOFU) approach could help. Bai et al. (2017) concluded in their research that participants preferred it when key distribution was taken care of for them.

Improve mental models of encryption

In order to get more people interested in encryption, the often faulty mental models that people have need to be improved (Dechand et al., 2019). Dechand et al. (2019) note that developers should consider the mental models that people have of encryption when creating a system. However, (Wu and Zappala, 2018) point out that changing the mental models that people have is not easy, and additionally, not sufficient to increase adoption. Wu and Zappala (2018) propose that ‘improved risk communication is necessary’. When explaining the importance of encryption, one must also focus on the *why*. They give as example that one participant noted they would understand the personal use of encryption as a means to protect oneself from a corrupt government. It was previously mentioned that (Wu and Zappala, 2018) noted that participants understood encryption strength in terms of the capabilities of hackers. Therefore, they recommend that it would help to frame encryption strength in terms of attacker capabilities.

Normalize encryption through widespread adoption

LaFleur and Chen (2014) believe that widespread adoption of encryption by companies will lead to other companies following in their footsteps. In the study of Dechand et al. (2019), participants claim that they would make use of encrypted messaging services if more people used them. However, one must note that this is also a circular issue: in order for more people or companies to adopt encryption, other people or companies must take the first step.

Improve usability

Improving the usability of encryption systems is also important. Smetters and Grinter (2002) propose that encryption systems should be re-designed from the ground up with usability in mind. For instance, Stransky et al. (2022) states that making use of color-coded messages for digitally signed messages made ‘users significantly less susceptible to social engineering attacks overall’.

2.4 Identity-based encryption (IBE)

Identity-based encryption (IBE) is an encryption scheme that mitigates the inconvenience of key management in public-key encryption (Identity-based encryption, 2022). See Figure 2.2 for a graphical representation of the scheme. Implementations of IBE make use of a trusted third party that controls a Private Key Generator (PKG). This PKG is in charge of distributing and computing the necessary keys using a Master Public Key (MPK) that it shares with users, and a Master Private Key (MSK) that is kept secret. In IBE, the user’s public key is known as an ‘identifier’ ID . This identifier is unique and can take on the form of any string, such as an email address.

In an example adapted from Youngblood (2005), Bob has a message M he wants to encrypt C and send to Alice. He first obtains a Master Public Key from the PKG (MPK_{PKG}). This MPK_{PKG} is combined with the identifier of the recipient, Alice (ID_{Alice}), to create the encrypted email C :

$$C = M + MPK_{PKG} + ID_{Alice}$$

When Alice receives C , she must first authenticate herself to the PKG that she is the owner of ID_{Alice} . After successful authentication, she receives her private key $SK_{ID_{Alice}}$, which she uses to decrypt the message into plaintext:

$$M = C + SK_{ID_{Alice}}$$

In practice, the process of encryption and decryption involves a complicated mathematical operation. However, for the sake of providing a simplified explanation, we depict this with the ‘+’ operator.

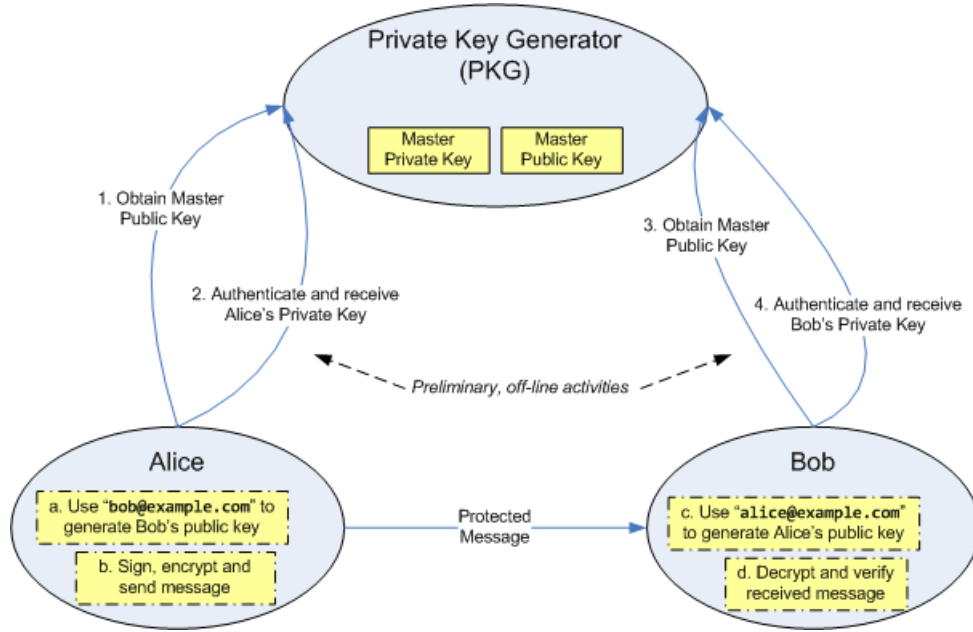


Figure 2.2: Identity-based encryption scheme (from Identity-based encryption (2022))

In IBE, the task of key management is delegated to the PKG, which centrally distributes keys upon request. As a consequence, if the trusted third party is compromised, so is the encrypted content. This is a risk of IBE.

2.5 IRMA

IRMA⁴, an acronym for ‘I Reveal My Attributes’, is an identity management platform for the purposes of authentication and digital signing. Originally developed at Radboud University, it is distributed via the Privacy by Design Foundation⁵ and SIDN⁶. The platform is available as a mobile application. IRMA can be regarded as a digital wallet that stores an individual’s personal information, which are referred to as ‘attributes’ (See Figure 2.3). Examples of these attributes are an email address, phone number, date of birth, or student identification. The attributes are provided by a number of verified instances known as ‘issuers’. In order to obtain an attribute, the user must first prove that they possess it. Authentication of possession is available, among others, through a verification code or link, or through DigiD (the identity management platform of The Netherlands). Currently, IRMA is focused on providing authentication for Dutch credentials, but there are plans for internationalization (IRMA in detail, n.d.).

IRMA is developed with two central principles in mind: privacy by design and data minimization (IRMA in detail, n.d.). Privacy by design stipulates that privacy is taken into account throughout the entire infrastructure of a product, and is therefore not treated as an afterthought. IRMA achieves this by informing the user about the attributes they possess and where they were obtained from. Upon authentication, users are shown explicitly which attributes they are sharing with external instances. Additionally, obtained attributes are only stored on the user’s phone. This transparency encourages users to be aware of their online privacy and gives them more control over their data. Data minimization instructs that only the strictly necessary information is exchanged, and nothing more.

⁴<https://irma.app/>

⁵<https://privacybydesign.foundation/>

⁶<https://www.sidn.nl/>

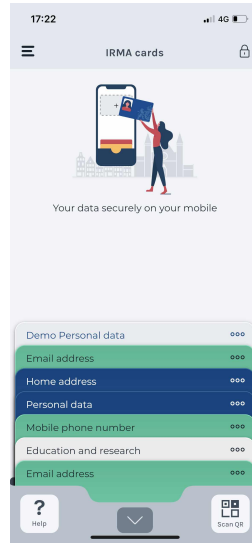


Figure 2.3: IRMA App with a number of attributes loaded in

To illustrate the benefits of using IRMA over conventional authentication methods, we provide an example. Imagine there is a website that requires all its users to be students. In conventional methods, the website will require a user to log in with their educational institute’s email. However, this means that the website gains more information than is strictly necessary – they not only know that the user is a student, but they also know the university of the student, and their name (as most such emails contain one’s full name). Considering the data minimization principle, we want to prevent this.

Now imagine the case where this website has adopted authentication via IRMA. The website requests that a user possesses the ‘student’ attribute. In order for the user to authenticate themselves, they scan a QR code displayed on the website (Figure 2.4). After scanning, the user is notified which attribute(s) they will be sharing with the website (Figure 2.5). In this case, the student agrees to share with the website that they indeed possess the ‘student’ attribute. The website receives the information that authentication was successful, but no additional information (Figure 2.6). We can now see that only strictly necessary information has been exchanged. A live demo of the above example and similar ones can be found on the Privacy by Design website⁷.

⁷<https://privacybydesign.foundation/demo/>



Figure 2.4: QR code that the user must scan

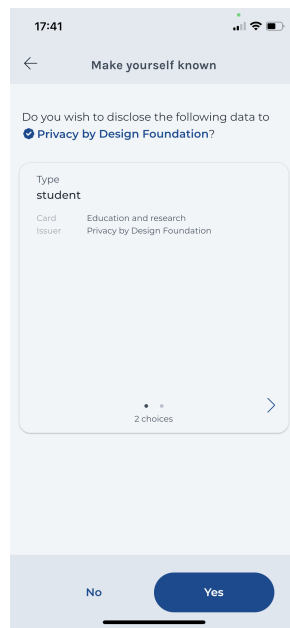


Figure 2.5: asking confirmation to user whether they want to disclose these attributes



Figure 2.6: Translation: Student authentication successful!

Next to authentication, IRMA also provides digital signatures. An individual can choose to digitally sign a product with certain attributes they possess. For example, if someone receives a digitally signed email from their doctor, they can scan the signature and confirm that the sender is indeed a certified medical specialist. However, it is important to note that the scope of the thesis is limited to the authentication aspect.

2.6 Postguard

Postguard provides end-to-end encryption (E2EE) for email and file sharing, and it is being developed by iHub⁸ at Radboud University. Our focus lies on its email encryption feature, which works via an add-on that is currently available for Thunderbird and Outlook. Postguard makes use of Identity-Based Encryption, thus Postguard also acts as the central key manager. As mentioned in Section 2.4, IBE requires users to authenticate themselves to prove that they are the rightful owner of an identity. Authentication is handled by IRMA.

Postguard provides the ability to make users authenticate with multiple attributes. Each unique combination of attributes requires separate authentication to the PKG, in order to obtain the right key. For example, when sending an encrypted email, the sender can mandate that its recipient is a student and/or have a certain date of birth, and/or have a certain name. When the recipient receives the email, they must prove that they possess all the required attribute(s) by authenticating themselves via IRMA.

Sometimes, an individual is unable or unwilling to install the add-on. This can be for various reasons, for example, they do not have Thunderbird or Outlook, or their job restricts them from using it. In this case, Postguard also provides a backup option called the ‘fallback’ that can be used to decrypt an email (Figure 2.7). The details of how the fallback works will be discussed in detail in Section 3.1.

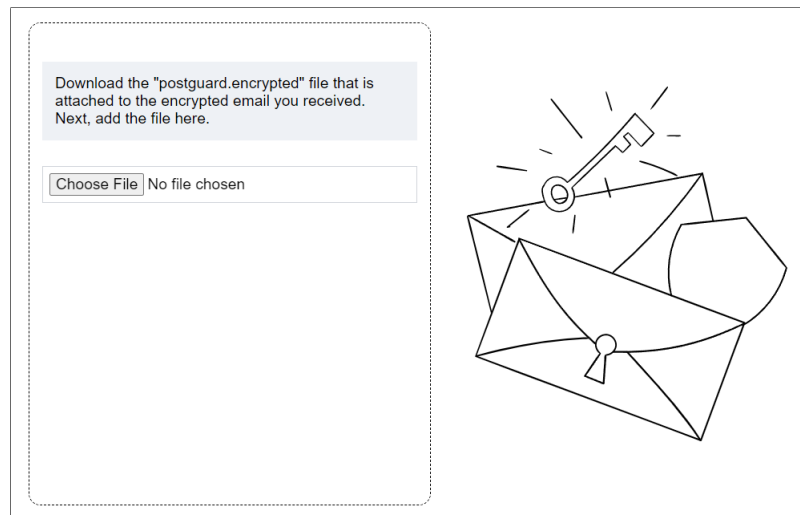


Figure 2.7: Postguard fallback website

⁸<https://ihub.ru.nl/>

2.7 Usability inspection

Below, we describe two methods for conducting a usability inspection. Usability describes whether a system is easy and intuitive for a user to navigate, whether the information presented is clear, and whether users are encouraged to perform the tasks in the way that is recommended (Kruger et al., 2020). Developing systems for usability is difficult. For example, it is easy for experts to make assumptions that the general user would not understand. It is also easy to forget that users have different preferences. Usability inspections can be conducted by different parties; they can be conducted by experts or end users. Ideally, we would like to perform inspections with both, but due to the time constraints of the thesis, we opted for experts. We introduce two usability inspection methods: the cognitive walkthrough and the heuristic evaluation principles. These aim to detect usability problems early on in the design process, so that amendments can be made more easily.

2.7.1 Cognitive walkthrough

A cognitive walkthrough is a technique where domain experts evaluate the usability of a system from the perspective of a new user (Polson et al., 1992). A domain expert is someone who is familiar with the system at hand. Examples of this can be a developer of the system, a UI/UX designer, or someone who is knowledgeable in the field of usability, design, or psychology. A new user is someone who encounters the product for the first time, and has no prior experience with similar products. This technique aims to evaluate whether such a user can, in an intuitive manner, navigate through the system and carry out the tasks required of them. A cognitive walkthrough focuses on the learnability of a system: “Learnability considers how easy it is for users to accomplish a task the first time they encounter the interface and how many repetitions it takes for them to become efficient at that task.” (Joyce, 2019). Those who participate in the cognitive walkthrough shall be referred to as ‘evaluators’.

Evaluator group

Instead of domain experts, our evaluators consist of three interns at Postguard (which includes the author of this thesis) who are familiar with both the technical and UI concepts of Postguard. Ideally, we would have liked to include the employees at Postguard, but because of resource and time constraints, this was not possible.

Setup

The cognitive walkthroughs were conducted on both the current fallback and the newly developed prototype fallback. These were conducted physically. It must be noted that inspection of both the current and prototype fallback was conducted in one session toward the end of the project. Due to scheduling constraints, it was more efficient to reserve one session for the evaluation so that all evaluators could be present. First, the concept of a cognitive walkthrough was explained to all evaluators. Then, the evaluators were given a physical sheet, called an evaluation form, seen in Figure 2.8. The specifics of the evaluation method are described further below.

User personas

In order to immerse the evaluators in the perspective of a new user, we employ user personas. User personas are fictionalized characters that have their own skill-sets, preferences, and behaviors (Harley, 2015). Individual traits and expertise greatly influence how a user understands the information that is presented to them. For example, someone with little knowledge of the Internet will have a much harder time navigating a technically dense system. Therefore, it is important that we take these different user perspectives into account, such that we do not make assumptions about knowledge and behavior.

Below are the two user personas we employed for the cognitive walkthrough. Names are randomly generated and faces are taken from <https://thispersondoesnotexist.com/>

Sofia Burton

Burton is a 39 year old Fiscal lawyer. She has one child and likes to spend her free time on practicing Judo and is the chair of the local Book Club.

Preferences: Rarely uses the Internet outside of browsing the news and checking her mail inbox.

Experience: Not experienced with the Internet.

Context: Uses Postguard because her friend sends her encrypted emails.



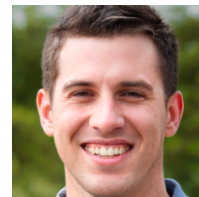
Richard Hastings

Hastings is a 22 year old Information Sciences student, with a special interest in digital privacy. In his free time he likes to play Dungeons and Dragons with his friends and does the occasional DJ-gig.

Preferences: Uses social media and plays online games.

Experience: Tech-savvy and experienced with the Internet.

Context: Uses Postguard for his internship.



We intend to diversify the user personas such that they would have different interests and specialties. For example, we can expect that Burton is less adept at using the internet than Hastings. Someone like Burton is also more likely to have difficulty with email encryption. For our cognitive walkthrough, we focus on the less technically apt persona, Burton, as it is important to us that the system is easy to use for a large public. In the case of Hastings, we can expect that this user has an easier time navigating complex interfaces, as they are more used to technology. However, we must note that the system must also be intuitive for someone who is good with technology, and not only focus on the technologically inept.

Evaluation method

To systematically evaluate the walkthrough, evaluators will review a number of tasks that users are expected to carry out with the system. Each task is broken down into the individual steps that must be carried out. An evaluation form is used to systematically review each step, see Figure 2.8. This form is taken from Salazar (2022), which is adapted from Wharton et al. (1994). We explain the questions in the form using examples from an email communication context.

The questions are as follows:

1. **Will users try to achieve the right result?**

Do users understand that the action (step) at hand is needed to reach their larger goal?

Example: if a user needs to finish sending an email, they need to understand they first need to click on the ‘send email’ button before their email is sent over the network. If this is not made clear to users, then they might be confused as to what the next step of their task is, or they might not even be aware that they have yet to perform another step to complete their task.

2. **Will users notice that the correct action is available?**

Is the interactive element that achieves the step visible or easily findable?

Example: if a user needs to finish sending an email, users need to be able to find the ‘send email’ button in the first place. If this is not clear, then users will not understand how to complete their task.

3. **Will users associate the correct action with the result they’re trying to achieve?**

Even if the correct action is clear to the user, is the result also in the form they expect? Or do they mistake it for something else?

Example: if a user needs to delete an email, they expect to find a delete button somewhere next to the email. However, in the case a button is unclear (for example, they don’t recognize the recycle bin symbol) or in the case the label is misleading (for example, not properly distinguishing between archiving an email or permanently deleting one), then users might be confused. This confusion can lead to users making mistakes. For example, they might accidentally archive an email they wanted to permanently delete.

4. After the action is performed, will users see that progress is made toward the goal?

Is enough feedback presented to the user so they know that they have made the right choice?

Example: if a user has clicked on the ‘send email’ button, they should be clearly informed that the email has indeed been sent. One example is by displaying a notification popup.

This evaluation form will be assessed for each individual step in the task. A step is only successful if all questions are answered with ‘Yes’. An action is only successful if all steps are answered with ‘Action success’.

Task				Action success		Action failure	
Action step							
Will the user try to achieve the right result?	yes <input type="checkbox"/>	from experience	the system tells them to	no <input type="checkbox"/>			
Will the user notice that the correct action is available?	yes <input type="checkbox"/>	from experience	they would see a call-to-action	no <input type="checkbox"/>			
Will the user associate the correct action with the effect they're trying to achieve?	yes <input type="checkbox"/>	from experience	a prompt/label matches action	no <input type="checkbox"/>			
After the action is performed, will the user see that progress is being made toward the goal?	yes <input type="checkbox"/>	from experience	there's a connection between the system response and user goal	no <input type="checkbox"/>			

Figure 2.8: Evaluation form for the cognitive walkthrough (from Salazar (2022))

It is important to note that the IRMA flow is included in the cognitive walkthrough for completion, but the focus of the thesis is not on the usability of IRMA. Feedback on that front will therefore be limited. Some steps in our task flow will also be skipped to prevent repetition. For example the steps such as opening or reading the email, setting up and authentication of IRMA, are only presented once.

2.7.2 Heuristic evaluation principles

A heuristic evaluation is, as defined by Nielsen and Molich (1990) “an informal method of usability analysis where a number of evaluators are presented with an interface design and asked to comment on it.” There are multiple methods for conducting a heuristic evaluation, we will be using the 10 principles developed by Nielsen (1994) and the design principles for actual security developed by Brandon et al. (2022). In our case, we use these heuristic evaluation principles in order to better identify and categorize the usability issues on both the current and new fallback websites.

Nielsen’s heuristics

When designing a system, Nielsen and Molich (1990) noted that developers had difficulty following the many design and usability guidelines that already existed. They proposed to condense and categorize these guidelines in a small set of principles that would be easier to evaluate.

Heuristic evaluations are preferably conducted by a small group of evaluators, ideally 5 (Nielsen and Landauer, 1993). Adding more evaluators did not significantly improve the number of problems found (Nielsen and Landauer, 1993). The list of heuristics we used and their descriptions are adapted from Nielsen (2020), which are in turn based on Nielsen (1994). We explain each heuristic using general examples.

1. Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Example: if a user needs to fill in a long form, they should be kept up to date on how many pages and/or questions are left until they are at the end of the form. This keeps users updated on how far they are in the process.

2. Match between system and the real world

The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, make information appear in a natural and logical order.

Example: users should not have to read industry specific terms, also known as jargon. An example of jargon might be the word ‘cache’. While it is easy for tech-savvy people to understand, the general public might not know what this word means.

3. **User control and freedom**

Users often choose system functions by mistake and will need a clearly marked ‘emergency exit’ to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Example: if a user has accidentally removed an email they did not want to remove, they should be easily able to restore it from their trash bin. The trash bin should be easy to find for the user and not hidden behind redirections or layers of pages.

4. **Consistency and standards**

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Example: it is important to stick to the same term on the same website. There might be multiple synonyms for the word ‘send’, such as send, compose, create, etc. A system should not use ‘send a message’ for sending a message on one page, but then use ‘compose a message’ on another page.

5. **Error prevention**

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Example: if users want to delete all their messages at once, they should be presented with a confirmation button, such as ‘Are you sure you want to delete all your messages?’ It is easy to make a mistake when clicking a button, so for important decisions it is better to ask to confirm.

6. **Recognition rather than recall**

Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Example: if a user needs a tutorial on how to navigate a site for the first time, it is better to break the tutorial down in individual steps instead of presenting them with a long list of explanation in one go. Whenever a user encounters a feature for the first time, a popup window appears to explain the feature to them. This results in an easier learning experience and does not overwhelm the user.

7. Flexibility and efficiency of use

Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Example: if an action is used often, a system can provide keyboard shortcuts.

8. Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Example: do not provide more images, colors, and/or styles than is needed. Too many decorations can overwhelm and distract a user.

9. Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Example: if a user misspells a word in a text box, they should be presented with a suggestion of the correct word.

10. Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Example: if a user is navigating a new social media site for the first time, they are provided with a step-by-step guide on what the different buttons, pages, and terms mean.

Design Principles for Actual Security

It must be noted that Nielsen’s set of heuristics apply to a broad range of systems and thus do not have security in mind specifically. For this reason, we use the 6 design principles for actual security developed by Brandon et al. (2022). They argue that some systems diminish actual security in favor of ease of use. Brandon et al. (2022) defines actual security as “the security provided by a system in practice, determined by (1) the security of the underlying technologies, and by (2) the extent to which users adopt the intended secure behavior”. We explain each principle using examples from an email communication context.

1. Consider Consequential Insecure Behavior

Users should be prevented from making insecure decisions, and enough information should be provided to guide users to make the most secure decisions. The system should also be sufficiently usable such that users will not try to create workarounds or avoid the system altogether.

Example: if a user needs to fill in their password every time they need to send an email, this could lead them to pick out a very simple password, just so that they can avoid having to type out a difficult password multiple times.

2. Enable the Intended Secure Behavior

The system should make the secure flow as seamless as possible and provide good feedforward that is intuitive to users.

Example: the secure behavior should not be hidden or hard to find for the user. If a user wants to encrypt their email, the ‘encrypt’ button should be easily visible to the user as they are writing the email. It should, for example, not be hidden behind multiple tabs.

3. Prevent Insecure Behavior

Good UI/UX design and restrictions should be employed to prevent users from making insecure decisions.

Example: using warning labels and color coding for messages and buttons.

4. Explain the Intended Secure Behavior

Users should be required to make as little effort as possible to understand what is expected of them. Adding explanations gives the user a feeling of control and makes them more likely to put in effort.

Example: if a user is writing an email, and they are presented with the option to encrypt or digital sign their message, the system should also explain to users what these terms mean. Otherwise, they are more likely to skip the option altogether because they do not know what the result is.

5. Reduce Social Obstacles and Resistance, and Create Trust

Users should be motivated to display secure behavior. Some ways to reduce social obstacles are to 1) build common ground, 2) address shared responsibility, and 3) make the security action a fun and cooperative activity through gamification.

Example: one way to create trust is to show to the user that an email has indeed been decrypted, for example by showing that only an encrypted ciphertext of the email is sent out.

6. Support Awareness, Recall, Remembering and Secure Habit Formation

Users should explicitly be made aware of security features so that they remember them. The secure behavior should include some friction and be distinct.

Example: it should be clear to a user when they have chosen to encrypt an email. For example, by giving encrypted emails a different color, symbol, or label, that distinguishes it from unencrypted emails.

Chapter 3

Evaluation of the current fallback

In this chapter, we first explain how the current Postguard fallback website works. Secondly, we perform a cognitive walkthrough. Thirdly, we inspect the issues the current fallback has with the help of the usability heuristics. We then propose a number of solutions that can be used to patch the aforementioned issues. From this, we decide upon a blueprint of the prototype which will be developed in Chapter 4.

3.1 Explanation of the current fallback

When a user receives an email that has been encrypted by Postguard, they are given two options to view the content. The first option is to make use of the Postguard add-on¹. This is also the option that is recommended by Postguard as it will make future encrypted correspondence easier. However, Postguard provides a fallback option for users who do not or can not install the add-on. The fallback is located on the Postguard website, thus it needs to be accessed via the browser. While the Postguard website itself is built in Svelte, a modern and high-performance frontend framework, the fallback is embedded as an iframe and written in Rust.

Firstly, an explanation of how the current fallback methods works:

1. The user receives an encrypted email.
2. The e-mail contains an attachment with the encrypted content, which they must download.
3. The attachment is selected on the fall-back website.

¹<https://postguard.eu/#addons>

4. If the email contains multiple recipients, the user is prompted to select their own email.
5. The user authenticates themselves by scanning a QR-code with the IRMA app.
6. If the authentication is successful, the attachment is decrypted and they can read the plaintext email.

When a user is sent a Postguard encrypted email, they are first given information on what Postguard is, and how to decrypt the email (see Figure 3.1).

The current design of the fallback consists of a label prompting users to download the ‘postguard.encrypted’ attachment they received in the email, and to add it in the filepicker box underneath. See Figure 3.2 below. It is important to note that at no point the contents of the file are uploaded to any server, even though the appearance of a file picker might suggest otherwise.

Once the encrypted file is added, the file is first checked for the number of recipients. If there are multiple recipients, the user is prompted to select which email belongs to them, as seen in Figure 3.3. This step is necessary as the fallback is unable to detect what the email of the user who is requesting the decryption is. Then, the user is prompted to authenticate themselves via the IRMA app. All the attributes must match precisely for successful authentication. After authentication, the decrypted email is displayed, as seen in Figure 3.4.

This is the extent of the features in the current fallback.

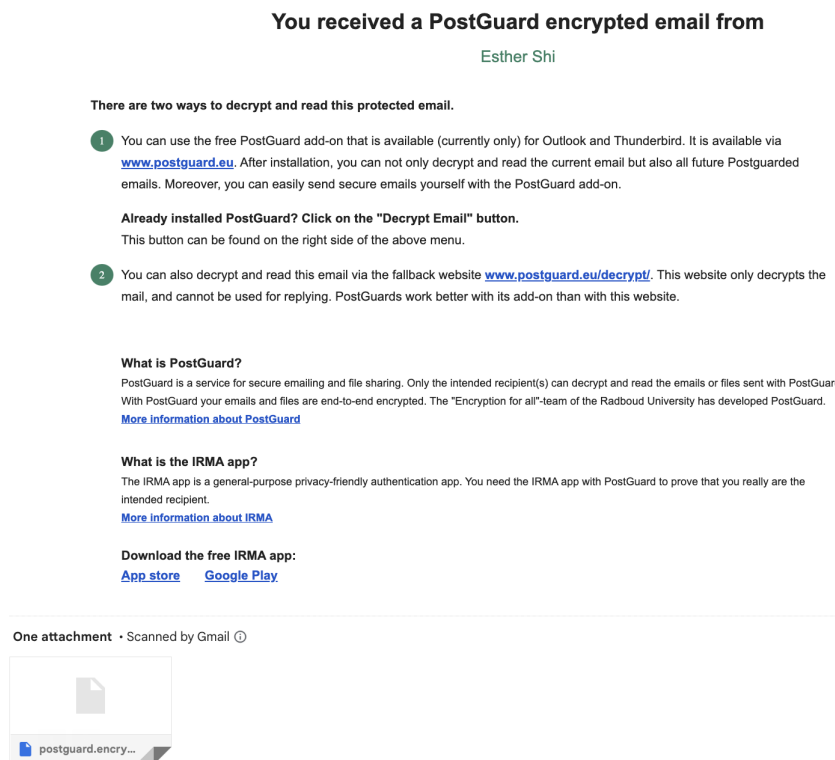


Figure 3.1: User receives an email from Postguard informing them they have been sent an encrypted email

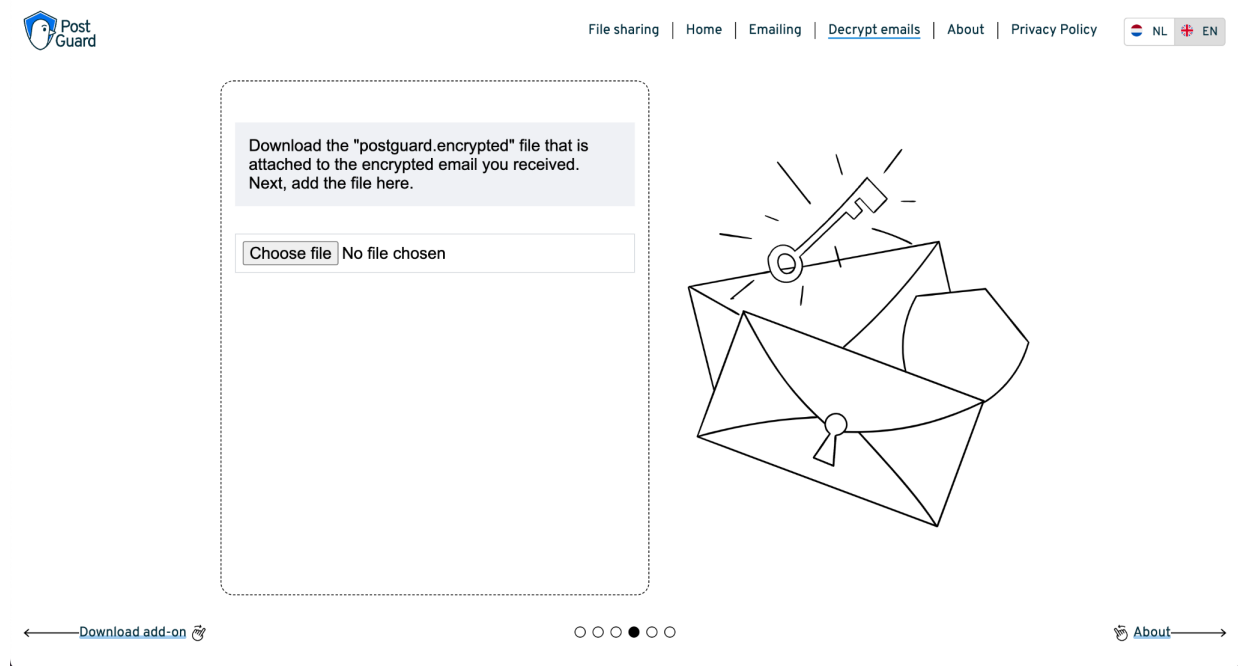


Figure 3.2: Current fallback page

Download the "postguard.encrypted" file that is attached to the encrypted email you received. Next, add the file here.

Choose file postguard-multiple.encrypted

This email has been encrypted for multiple recipients. Which recipient are you?

Select a value

Figure 3.3: If an email has multiple recipients, the user must select their own email

Message decrypted successfully

Choose file postguard-email1.encrypted

From esthrshi@gmail.com

To esthrshi@gmail.com

Date Sun Nov 13 21:01:31 2022

Subject this is a test

encrypted wooh

Figure 3.4: The decrypted email is displayed

3.2 Cognitive walkthrough

For the current fallback, we evaluate the following task:

Task 1: Download the encrypted file and decrypt it.

A visual guide to all the steps of this task can be found in Appendix A.

Task 1: Download the encrypted file and decrypt it

1. Download the encrypted file and decrypt it
2. Click on the link in the email for people without an add-on
3. Upload the postguard.encrypted file
4. Install the IRMA app from the App Store
5. Open the IRMA app
6. Navigate to ‘Adding cards’
7. Choose ‘email address’ and click on ‘Add’
8. Fill in your email address
9. Click on the confirmation link in the email you received
10. Scan the QR code with the IRMA app
11. Fill in the pairing code
12. Click on ‘Yes’
13. Scan the QR code on the fallback website
14. Click ‘Yes’
15. View decrypted email

After evaluating the steps, we conclude that all steps are successful. However, there are a number of usability remarks.

At step 3, the file picker insinuates that the file will be uploaded to some server. This is not the case. Nielsen (1994)’s 10th heuristic suggests that it is good to provide help and documentation to the user when needed. It is advised to add explanation for the user that all the decryption happens on the client-side, and the website will not be able to see the encrypted nor the decrypted file. The label of the file picker could also be refined. Instead of the label being named ‘Choose File’, it is advised to make the command more explicit by titling it ‘Choose file you want to decrypt’.

Additionally, if a user wishes to remove a file, an explicit remove file button should be present. This is in line with Nielsen (1994)'s 3rd heuristic: user control and freedom. If users make a mistake, they should be presented with a clear option to undo their choice.

After the user has picked their encrypted file, an IRMA QR code pops up. However, IRMA will be unfamiliar to a new user. Extra information about IRMA should be presented (again, Nielsen (1994)'s 10th heuristic: Help and documentation). The system wrongfully makes the assumption that they are familiar with the IRMA app. We are aware that information on IRMA is included in the email, but the user might have already forgotten this by the time they are on the site. It is possible to only display the information when the user needs it, for example, by adding an information symbol that the user can choose to click on.

3.3 Issues with the current fallback

In addition to the issues identified after the cognitive walkthrough, we also identified a number of bigger problems with the current fallback that hamper its usability. This was executed via an informal inspection by navigating the site. In order to improve on the current fallback we must first analyze its shortcomings. With the help of the usability heuristics, and through internal discussion with the Postguard development team, we identify a number of issues. It is important to note that for identifying the issues, we take into account the perspective of a user who must use the fallback regularly. With regularly, we define that they must decrypt multiple emails per day with the fallback.

We then label the issues with a priority-indicator, which can be one of {High, Medium, Low}. The weights of the priorities are defined as follows:

- High: this is an issue that makes the fallback near-unusable because the usability concerns make the process incredibly complex for the user.
- Medium: this is an issue that makes the process inconvenient for the user.
- Low: this is an issue that only slightly inconveniences the user, but it does not affect usability that much.

Issue 1: Downloading the attachment

The user receives an email with an attachment. They have to save this attachment somewhere on their device where they can find it (such as the downloads folder), and then select the attachment on the fallback website. This entire process is cumbersome. Downloading an attachment is particularly inconvenient on a mobile device, as they have a less sophisticated file manager compared to a computer. Prompting users to save an attachment that might be foreign to them breaks one of the ‘Design Principles for Actual Security’ by Brandon et al. (2022). Namely, it breaks ‘Design Principle 1: Consider Consequential Insecure Behavior’. The email wrongfully encourages users to download an attachment they are most likely unfamiliar with. Users should instead be warned to distrust unknown attachments, especially those with uncommon extensions such as ‘.encrypted’. Attachments are also often associated with malware. All of this combined makes downloading an attachment an issue that hampers both usability and security.

Issue 2: Re-authentication with IRMA

If you use Postguard via the add-on, the add-on caches the IRMA credentials for a certain period. This means that the user does not need to re-authenticate every time they open a new email, given that the requested credentials are identical and not more than one day has passed.

However, the fallback has no such caching feature, which means that the IRMA app is needed to authenticate every session. Performing the IRMA authentication step every time one needs to decrypt an email is cumbersome. As a consequence, this complication is so severe that it breaks ‘Design Principle 1: Consider Consequential Insecure Behavior’. Users would encourage others not to make use of Postguard’s encryption so they do not have to make the effort of using the fallback. In the worst case, people would prefer not to use encryption so as to not inconvenience their recipients, or their emails would simply not get read anymore.

Issue 3: Accessing previously decrypted emails

Once the email has been decrypted, the fallback provides no way to store it. This means that the user cannot read, search, reply to, or forward their previously decrypted emails. In short, most, if not all facets of a regular email correspondence become impossible. As a result, if the user wants to revisit their emails, they must store it somewhere on their device. Users are thus unintentionally led to save their plaintext emails in Word documents, or somewhere on the cloud, which could expose them to a data leak in case of a security breach. This shortcoming breaks Brandon’s ‘Design Principle 3: Prevent Insecure Behavior’. If users store their plaintext email correspondence in unsafe locations, this nullifies the purpose of encrypting them via Postguard in the first place.

Issue 4: Inability to reply to email

Another issue that becomes apparent is that once the email is decrypted, users are unable to directly reply. If they want to reply, they need to manually compose an email to the sender’s email address. As a result, most people would also copy-paste the contents of the decrypted email to provide context to the recipient. Users cannot respond with an encrypted email from their own email address as this requires the functionality of the add-on. This results in similar problems as presented in Issue 3, which is that people are encouraged to perform insecure behavior that nullifies the purpose of Postguard.

Issue 5: Inability to access emails on multiple devices

One last identified issue is that users cannot access their previously decrypted emails on multiple devices. Most people use at least both their computer and mobile phone for email communication, and it is inconvenient to be reliant upon one device.

Problem	Priority	Explanation
1: Downloading attachment	Medium	Having to download an attachment results in extra work. This is especially inconvenient on mobile devices.
2: Re-authentication with IRMA	High	Having to re-authenticate oneself every time one wants to read an email can quickly become a great annoyance.
3: Accessing previously decrypted emails	High	Not being able to read previous emails, or having to save them elsewhere, can be considered a quite significant overhead for the user and can also lead to insecure behavior. It is also a central feature of emails to be able to read them back.
4: Inability to reply to email	Low	While inconvenient, this is not a must-have for usability.
5: Inability to access emails on multiple devices	Low	While inconvenient, this is not a must-have for usability.

3.4 Proposed solutions

Now that we have evaluated the current fallback and detected the various issues, we move on to evaluating the potential solutions. These solutions are the combined effort of suggestions made by the supervisor Dr. Hanna Schraffenberger, the Postguard team, and the author of this thesis. Each solution solves a different subset of the aforementioned issues, and might present new issues of their own, which will be described.

Solution 1: Forward decrypted email after authentication

This is a proposal to solve Issue 1: Downloading the attachment. The idea of this solution is that once the user has authenticated themselves on the fallback, the decrypted content is sent via an email to the user. The decrypted content would need to be sent from an email address that is controlled by Postguard. However, this solution presents a glaring problem: in order for the user to receive the decrypted email, the plaintext content is sent over insecure means. This nullifies the purpose of using Postguard in the first place. Another disadvantage is that this adds overhead for Postguard, as it becomes responsible for sending out every decrypted email. To add, this solution has been implemented by TweedeGolf in an earlier prototype, but has been discarded because of the aforementioned disadvantages. Due to this, Solution 1 is not recommended.

Solution 2: Place encrypted email in URL

This is a proposal to solve Issue 1: Downloading the attachment. Postguard can include the encrypted email content in a URL, which will be presented to the user in the form of a link. The format of the URL is composed of a referrer to the fallback site, concatenated with the encrypted email in Base64 format:

```
postguard.eu/decrypt#encrypted={encrypted_email_in_base64}
```

In a URL, everything that is placed after the hash symbol # does not get sent to a server. This means that the contents of the encrypted email is processed entirely client-side, preserving privacy. By clicking on the link, the user is redirected to the fallback site, which detects the encrypted content, and can then be prompted to decrypt this after successful IRMA authentication.

However, there are two problems with this approach. Firstly, there are limitations to the maximum length that a URL can be. This presents a problem as large emails, such as those including attachments, can quickly result in a URL length that exceeds this maximum. It is advised that URLs

be no longer than 2,048 characters² if one wants them to be functional in most popular browsers. However, this advice **only holds for the section of the URL that is sent to servers**. As we mentioned before, the content of the URL placed after a hash symbol # is ignored by the server, giving us more leeway. Different browsers handle different such limits. By conducting an analysis of some of the most popular browsers, the following results were found:

- Chrome: 512KB
- Firefox: 512KB
- Safari: up to 10MB or even higher. It seems that Safari does not impose a fixed limit.

In the case of Safari, it must be noted that a URL of 10MB slows down the site enormously, resulting in a load time of more than 10 seconds. This is very slow for internet speed. Therefore, while technically possible, it is strongly discouraged to operate with URLs of this size.

The average size of an email is 75KB (Tschabitscher, 2021), thus it is viable to use the URL approach in most cases. We should also consider that the maximum email size that Gmail allows is 25MB. If an email is larger, Gmail will prompt you to make use of Google Drive to transfer your attachment instead³. It should be noted that Postguard also offers an encrypted file sharing solution⁴ for files up to 2GB, which should be plenty for most cases. It can thus be argued that, in the case that a user needs to include a large attachment, they should make use of Postguard's file sharing feature instead. It is also possible for Postguard to implement a feature that automatically converts a large attachment into Postguard's file sharing link, facilitating a good user experience. In short, while URL limitations present difficulty with attachments, there are workarounds that are feasible.

The second problem is, while this solution takes care of Issue 1, it introduces a security concern, namely encouraging users to click on links. According to 'Design Principle 1: Consider Consequential Insecure Behavior', users should not be taught insecure behavior. Especially in this case, when we have a very long link with unreadable encrypted content, which should alert the user. However, this disadvantage can be mitigated as Brandon et al. (2022) suggest to accompany links with a warning text. This text explains what the URL contains and cautions users to never just click on links without examining its content. While this does not completely make up for the insecure design, it can be considered a decent compromise, especially as it avoids the hassle of downloading an attachment.

²<https://www.sitemaps.org/protocol.html>

³<https://support.google.com/mail/answer/6584?hl=en&co=GENIE.Platform%3DDesktop&zipy=%2Cattachment-size-limit>

⁴<https://postguard.eu/#filesharing>

Solution 3: Include decryption module in e-mail

This is a proposal to solve Issue 1: Downloading the attachment.

Postguard can send the encrypted email and the decryption module as attachments. The decryption module is a few kilobytes in size. The user can download the decryption module and use this to decrypt the email.

However, the disadvantage is that the size of the decryption module is an overhead for email traffic, the module itself is only a few kilobytes, but this scales as large volumes of email are sent out, which makes this option less practical.

Solution 4: Copy paste content

This is a proposal to solve Issue 1: Downloading the attachment.

Instead of Postguard sending the encrypted email as an attachment, it can include it in the email body itself. Users will have to copy-paste this ciphertext from the email to the fallback website and authenticate as usual to read the content. There are some concerns when it comes to displaying the entire encrypted content for the user to see. Firstly is the issue of length. As emails can run very lengthy, so can the resulting encrypted content. This means that the body of the email can become very large, and the user would have to manually copy all of this text. While not infeasible, it does hamper usability. For those on mobile, copying large bodies of text is even more difficult. Secondly, the presence of encrypted text can intimidate a user. When the content is sent as a file, the encryption acts largely as a black box, the user needs only to download the file, they do not need to know or understand what is in it. In this approach, we must explain to the user that what they are copying is their email, but encrypted. One argument in favor of this approach is that it offers more transparency to the user. The study of Ruoti et al. (2013) supports this argument, as users preferred the encryption tool where the encryption steps were performed explicitly.

Solution 5: Cache emails and IRMA credentials on fallback

This is a proposal to solve Issue 2: Re-authentication with IRMA.

The fallback site can cache the IRMA authentication like the Postguard add-on does. As explained in Section 2.6 in the preliminaries, every combination of credentials is unique. When the site requests authentication via IRMA, a JSON Web Token (JWT) is returned in case of successful authentication. This JWT is valid for a certain period. Each combination of credential(s) and its respective token will then be stored in local storage for as long as it is valid. The next time a user requests to decrypt a file with a credential that has already been validated, the fallback can immediately decrypt the email without having to ask for authentication.

Implementing this solution takes care of Issue 3, users do not need to have their phone at hand at all times if they are on their desktop, and it reduces overhead. To address security concerns, we are aware that by caching the credentials, this might expose the user to data leaks in case a malicious actor gets access to the victim's device. This would mean that they are able to decrypt emails with credentials that are cached at the moment. However, it must be noted that the same security risk is present with Postguard's add-on, as it also caches these credentials. It is important to note that the JWT is stored in local storage, this makes the process relatively secure. Therefore, the efficiency of caching is a larger advantage over the security risk.

Solution 6: Caching on multiple devices using storage.sync

This is a proposal to solve Issue 5: Inability to access emails on multiple devices and Issue 3: Accessing previously decrypted emails.

A version of local storage for multiple devices is possible using storage.sync⁵. This enables the data to be cached and accessed on all browsers where the user is logged in. On Chrome this means that the user must be logged in on their Google account. On Firefox the user must be logged in and have their Add-ons enabled. It must be noted that while this solution improves ease-of-use, it also means that the emails and IRMA credentials will automatically be available on all devices the user is logged into their browsers. This is a small disadvantage for security.

Additionally, and more importantly, to provide this feature means that the data will be stored on a server. As this is in conflict with our goals, Solution 6 is not recommended.

⁵<https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/storage/sync>

3.5 Deciding on a prototype

In this section, we pick a subset of the solutions proposed in the previous section to implement for the prototype. Ideally, we would have liked to develop prototypes for multiple solutions, however, because of the scope of this thesis, we limit ourselves to implementing one solution for each issue.

Issue 1: Downloading the attachment

For solving Issue 1, we opt for ‘Solution 2: Place encrypted email in URL’. One benefit of this solution is its ease of use, that gives it an advantage over ‘Solution 4: Copy paste content’, which requires users to copy paste text themselves. Another benefit is that it does not increase the size of the email, like ‘Solution 3: Include decryption module in e-mail’ would. The disadvantage of using URLs is that there is a size limit, but we believe this limit to be sufficiently small that the benefits outweigh them.

Issue 2: Re-authentication with IRMA and Issue 3: Accessing previously decrypted emails

To solve Issue 2 and 3, we opt for ‘Solution 5: Cache emails and IRMA credentials on fallback’ by storing the credentials and emails in local storage.

Issue 4: Inability to reply to email

After some evaluation, we come to the conclusion that this issue cannot be solved in a proper way. This is because there is no way to compose an encrypted reply that is sent from the user’s email address via the fallback. We note that it is in some way possible to use a third-party email (one that Postguard is in charge of, in similar fashion to Solution 1) to send an encrypted email to the recipient. However, we want to prevent the usage of a method that requires a ‘middle-man’.

Issue 5: Inability to access emails on multiple devices

While it is technically possible to solve this issue as can be seen with Solution 6, we decide that we do not want to have the decrypted emails and JWT tokens stored on a server.

Chapter 4

Development of the prototype

In this chapter we explain how a functional prototype of the fallback was developed. The prototype is available on the Github repository¹.

4.1 Setup

The current fallback is built in Rust, which is embedded as an iframe on the Postguard website. We opted to build the prototype in Svelte² as this is the language that the Postguard website is written in, so for the goal of uniformity, the prototype will be in the same language. First, we recreated the same functionality that the current fallback has, namely, the ability to authenticate via IRMA and decrypt an email. Then, we implemented the features decided upon in Section 3.5:

- Cache the IRMA credentials so that the amount of times one has to authenticate is reduced
- Cache the decrypted emails so they can be re-accessed later
- Implement a method to decrypt an email via URL

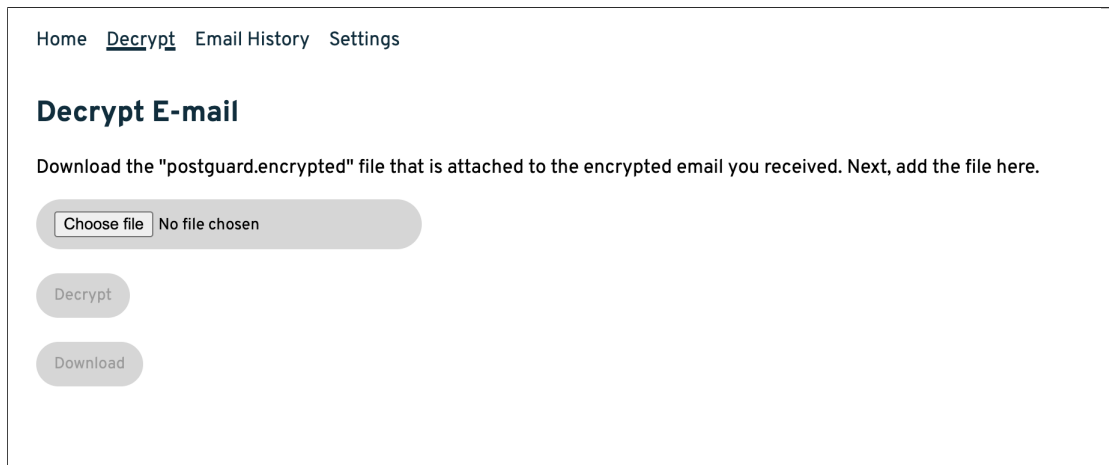
The caching feature makes use of local storage³ which allows you to store information that is only available client-side.

¹<https://github.com/esthrshi/postguard-fallback-thesis>

²<https://svelte.dev/>

³<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

4.2 Decryption Page



The screenshot shows a web interface for decrypting an email. At the top, there is a navigation bar with links: Home, Decrypt, Email History, and Settings. Below the navigation bar, the title "Decrypt E-mail" is displayed. A text instruction reads: "Download the 'postguard.encrypted' file that is attached to the encrypted email you received. Next, add the file here." Below this instruction is a file upload area with a button labeled "Choose file" and a status indicator "No file chosen". Below the file upload area are two buttons: "Decrypt" and "Download".

Figure 4.1: Decryption page

The decryption page allows users to decrypt an email via an attachment or via the URL. Decryption is possible by using the IRMAseal web assembly modules⁴, which takes care of the file decryption itself. Figure 4.2 shows the implemented decryption flow from start to finish. The current fallback contains steps 1, 2, 3, 6, and 7. The prototype adds the steps 4 and 5 for the IRMA caching in between. We explain all the steps below.

⁴<https://www.npmjs.com/package/@e4a/irmaseal-wasm-bindings>

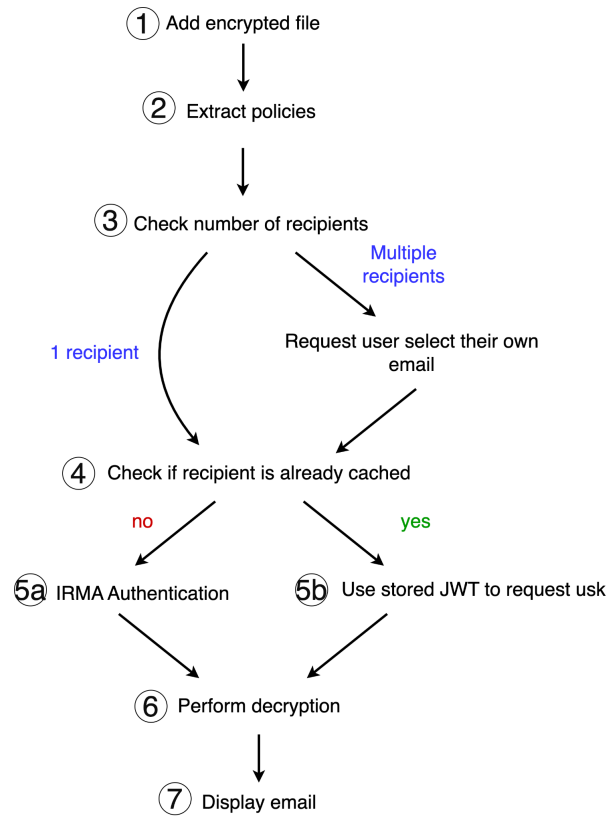


Figure 4.2: Implemented decryption flow

1: Add encrypted file

To decrypt a file, the user adds the ‘postguard.encrypted’ file attached to the Postguard email to the file picker. It must be noted that the file picker does not upload anything to a server. The entire process of decryption takes place client-side. The file is encoded in Base64.

We also offer the possibility of decryption via URL. To make this possible, the encrypted email is included as a parameter:

`postguard.eu/decrypt#encrypted={encrypted_email_in_base64}`

However, as mentioned before, this alternative has a size limit. The tested limit is 512KB on Chrome and Firefox. It is possible for the Postguard plugin to disable this option if they detect that an email content exceeds 512KB.

2: Extract policies

IRMAseal will then extract the policies from the file. Policies contain information about the recipient(s) of the email. The key(s) are the recipient's email address(es), together with a list of the requested attributes and their values.

```
bob@email.com:
  con:
    0:
      t: 'pbd.f.gemeente.personalData.surname'
      v: ''
    1:
      t: 'pbd.f.pbd.f.surfnet-2.id'
      v: 's123****'
    2:
      t: 'pbd.f.sidn-pbd.f.email.email'
      v: ''
    3:
      t: 'pbd.f.sidn-pbd.f.mobilenumber.mobilenumber'
      v: '+3161234****'
  ts: 1234567890
```

The format of these attributes are defined by IRMA⁵. Sometimes, a user can have multiple instances of the same attribute. For example, a user can have multiple phone numbers. In this case, a hint is visible and this will also be displayed to the user.

3: Check number of recipients

An email can be addressed to one or multiple recipients. In case the email has multiple recipients, the fallback needs to ask the user which recipient belongs to them. This is done via a dropdown menu.

4: Check if recipient is already cached

Firstly, the system checks whether the recipient's credentials are already cached. The requested attributes and their values must be exactly the same for a match.

If recipient is not cached – 4a: IRMA authentication

Before authentication, the key request must be composed. This consists of the requested credentials with their values removed. The validity key specifies how long the requested key should be valid for. Currently, the upper limit is set to one day by the Postguard plugin, but this limit can be adjusted by Postguard.

⁵<https://privacybydesign.foundation/attribute-index/>

```

con:
  0:
    t: 'pbdg.gemeente.personalData.surname'
  1:
    t: 'pbdg.pbdg.surfnet-2.id'
  2:
    t: 'pbdg.sidn-pbdg.email.email'
  3:
    t: 'pbdg.sidn-pbdg.mobilenumber.mobilenumber'
  validity: 25351

```

For authentication, a request is sent to IRMA, and users will have to authenticate their identity by scanning a QR code. After successful authentication, the server returns a JSON-Web Token (JWT). Using this JWT, the fallback makes a request to the Private Key Generator (PKG), located at <https://main.irmaseal-pkg.ihub.ru.nl>, to obtain a User Secret Key (USK).

If the user has enabled 'Cache my IRMA credentials' in Settings, then this JWT will be stored in local storage. It must be noted that each combination of attributes is unique and requires its own JWT. Additionally, the expiration date must be included to know when a JWT has expired. Therefore, a cache entry consists of the combination of attributes, its respective JWT, and its expiration date.

If recipient is cached – 5b: Use stored JWT to request usk

If the local storage has already cached a recipient with the exact same attributes, the JWT is taken from this cache. Using this JWT, the fallback performs the same request to the PKG as in Step 4a to obtain the USK.

6: Perform decryption

After the USK is obtained, the file is decrypted using the IRMAseal module. This results in a plaintext email in MIME format.

7: Display email

The returned plaintext email is thus parsed and displayed.

```

Date: Thu, 08 Dec 2022 19:03:02 GMT
MIME-Version: 1.0
To: bob@email.com
From: Alice <alice@email.com>
Subject: this is an encrypted email
Content-Type: text/html; charset=utf-8
This is the body of an email

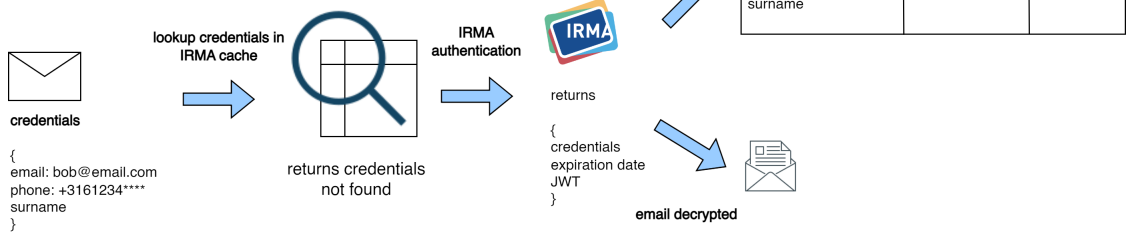
```

The process of how the IRMA credentials cache works is illustrated in Figure 4.3.

Before first decryption: IRMA cache is empty

credentials	expiration date	JWT

First decryption



Second decryption with same credentials

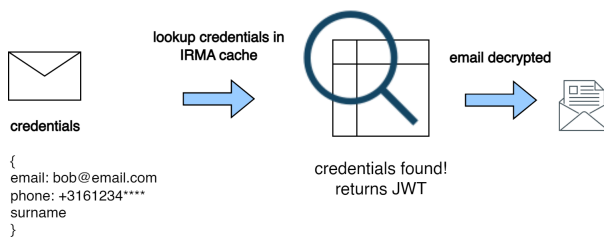


Figure 4.3: IRMA cache flow

4.3 Email History Page

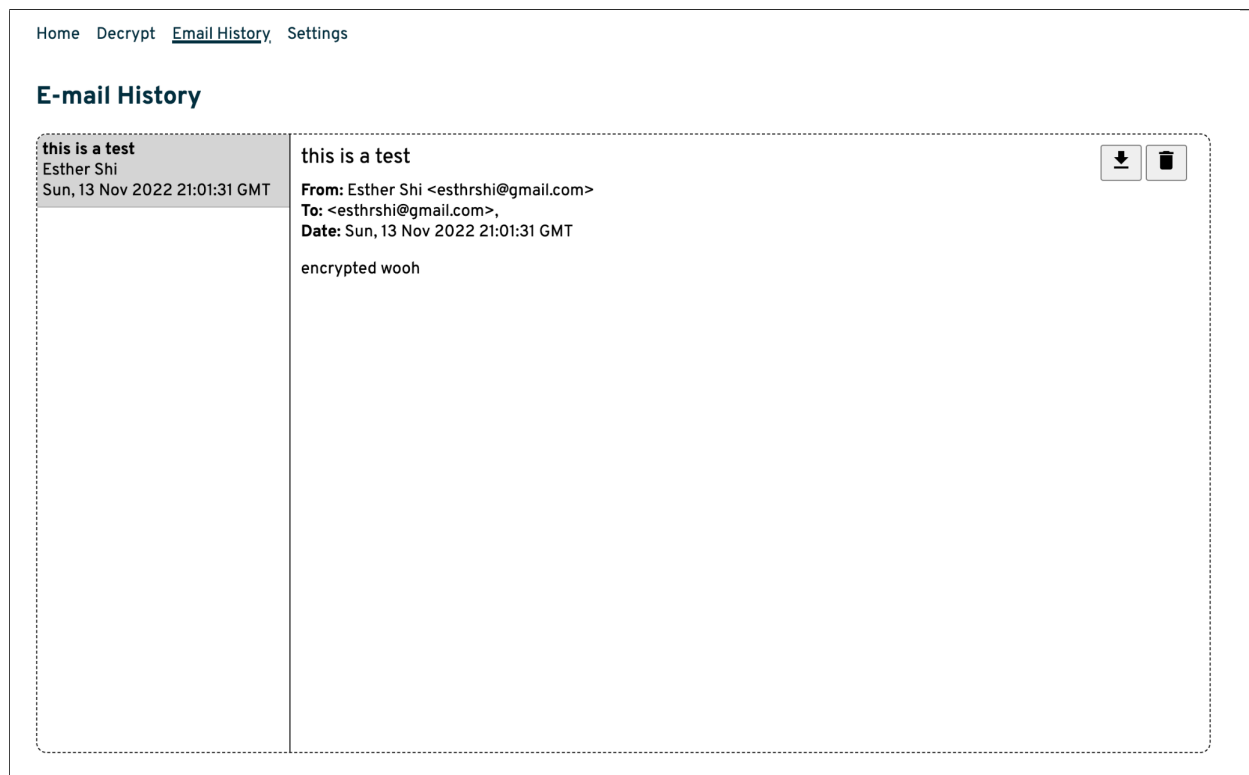


Figure 4.4: Email History page

Decrypted emails are stored under the page ‘Email History’ if the user has enabled ‘Cache my emails’ in Settings. All emails are stored in local storage. This page mimics a typical email client such as Outlook as we want to make the layout intuitive to the user.

In the left sidebar, users will see a list of their emails with the subject, sender, and date of each. If they want to view the body of an email, they must click on one of the items. This will open the entire email in the right body. Users are also able to download or delete an email. Downloading an email will give the user their email in .eml format.

Local storage unfortunately has a storage limit of only 10MB. We advise that emails should be stored without their attachments, as they can make the file very large. Instead, we encourage users to use Postguard’s file sharing solution⁶ for attachments.

⁶<https://postguard.eu/#filesharing>

4.4 Settings Page

[Home](#) [Decrypt](#) [Email History](#) [Settings](#)


Settings

All IRMA credentials and decrypted e-mails are cached locally in the user's browser, no information is sent to a server.

Caching

- ☒ Cache my emails
- ☒ Cache my IRMA credentials

IRMA Credentials

Credentials	Expiry date	
esthrshi@gmail.com	Tue, 20 Dec 2022 04:00:00 GMT	

[Delete all IRMA credentials](#)

Email History

[Delete all cached emails](#)

Figure 4.5: Email History page

On the Settings page, users can choose whether they want their emails and IRMA credentials cached. They can also view all the IRMA credentials that have been cached, and have the option to delete them individually. Users can also choose to delete all of their emails and IRMA credentials at once.

Chapter 5

Usability inspection of the prototype

In this chapter we describe the usability inspection on the newly created prototype that was carried out using a cognitive walkthrough. We also incorporate the usability heuristics in our feedback of the walkthrough. A visual guide to all the tasks can be found in Appendix B. We conclude with a highlight of our findings.

For the fallback prototype, we evaluate the following tasks:

- Task 1: Download the encrypted file and decrypt it
- Task 2: Turn on IRMA- and email caching in settings
- Task 3: Decrypt another email (same credentials as previous one) with the URL
- Task 4: Decrypt another email with multiple recipients and different credentials as previous one
- Task 5: View a cached email and download it
- Task 6: Remove all cached emails and all IRMA attributes

Aside from Task 1 being the same as the one in the current fallback, Tasks 2-6 are newly added. This is because the added features in the prototype allow for more extensive testing using a cognitive walkthrough.

Task 1: Download the encrypted file and decrypt it

1. Add the postguard.encrypted file
2. Click on ‘Decrypt’
3. Perform IRMA authentication
4. View email

All steps are successful but there are some usability concerns for the decryption page.

Step 1: Add the postguard.encrypted file - SUCCESS

Firstly, the grayed out buttons, made to signify that the button is inactive and thus cannot be clicked on, can be hard to read for those with visual impairments. Instead, it might be more effective to change the color of an active button to green.

The confusion with the file picker from the current prototype remains. It must be made explicit that users are not uploading their encrypted content anywhere.

Step 2: Click on ‘Decrypt’ - SUCCESS

No remarks.

Step 3: Perform IRMA authentication - SUCCESS

At step 3, the same issue from the current prototype regarding the IRMA authentication remains. The fallback does not provide enough information to the user about what IRMA is (and why they need it).

Step 4: View email - SUCCESS

The download button does not clearly communicate that it will download a file with the email content. Additionally, the design of the button is also identical to the preceding ‘Decrypt’ button, this gives users the impression that somehow it is a button they must press to finish their task. One suggestion is to provide additional information in a label, such as “Want to download this email to your device? Click on the download button below.” in order to stress it is an optional decision.

Task 2: Turn on email- and IRMA caching in settings

1. Go to ‘Settings’
2. Check ‘Cache my emails’ and ‘Cache my IRMA credentials’

Step 1: Go to ‘Settings’ - SUCCESS

No remarks.

Step 2: Check ‘Cache my emails’ and ‘Cache my IRMA credentials’ - FAIL

Step 2 presents some glaring usability problems, especially on the topic of jargon. On the evaluation form, the following question “Will the user associate the correct action with the effect they’re trying to achieve?” was answered with NO. It was concluded that users will have difficulty understanding what caching is, this is therefore seen as jargon. Nielsen (1994)’s second heuristic “Match between system and the real world” stipulates that familiar phrases should be used. It is better to use a more general term such as ‘Remember my emails’. This conveys more or less the same message as caching does, but it saves the user from jargon, and also saves the interface from adding too much text that could make the page look bloated.

It is also beneficial to add an information block to explain that all information is stored on the user’s device – no information is sent over servers. This could be written out in a more user-friendly way such as ‘No one but you can see this information’. By describing the feature in terms of security, it immediately communicates to the user that this information is secure in a way they can understand.

Furthermore, it is also important to add why caching the IRMA credentials makes the user’s life easier. Extra information such as this can be included in an information window next to the setting.

The evaluation question “After the action is performed, will users see that progress is made toward the goal?” also resulted in a NO. When the checkbox changes state, the settings are updated instantly. However, from a user perspective this is unclear, as users expect to see either a save button (to save their choices), or a notification to announce that their settings have been updated. Nielsen (1994)’s first heuristic also suggests this: “Visibility of system status”, which specifies that users should be kept informed of the system status. This lack of feedback makes it unclear for the user. It is therefore necessary to either add a save button or a notification popup that appears whenever the user has made a change.

Task 3: Decrypt another email (same credentials as previous one) with the URL

1. Click on the link in the email for people without an add-on (URL version)
2. Click on ‘Decrypt’

Step 1: Click on the link in the email for people without an add-on (URL version) - SUCCESS

At Step 1 there are some problems with the interface. Firstly, the hyperlink is unclear to the user. It is best to rewrite ‘Please click on this link’ to ‘Please click on the following link’ followed by a hyperlink of ‘Decrypt this email’. Instead of using a hyperlink, it might be more intuitive to use a button. Secondly, it is also important to explain to the user that the button or link contains the entire encrypted email, so users are informed what link they are clicking on. Nielsen (1994)’s tenth heuristic “Help and documentation” also stipulates that users should be presented with extra information when needed.

Step 2: Click on ‘Decrypt’ - FAIL

The evaluation question “Will the user try to achieve the right result?” yielded NO. The first problem is that the label ‘Found an encrypted email which can be decrypted!’ is confusing to the user. The user might not understand that the website has parsed the contents of the URL and therefore has detected the encrypted content. To the user, they’ve simply clicked on a link they were told would show them their email. Secondly, the word ‘detected’ scares users, as detected is often associated with viruses. The text should be rewritten to include active language, explanation should be offered to the user, and it should include a call-to-action to make explicit what the user needs to do to read the email. An example: “The link you clicked on includes an encrypted email. You can read this email after you have proven yourself to be the intended recipient. To do this, please click on ‘Decrypt’.”

Task 4: Decrypt another email with multiple recipients and different credentials from the previous one

1. Select the email address that belongs to the recipient
2. Click on ‘Decrypt’

Step 1: Select the email address that belongs to the recipient - FAIL

The evaluation question “Will users associate the correct action with the effect they’re trying to achieve?” received a NO. This step is confusing as users do not understand the need for selecting their own email. Users are accustomed to their own email client, which automatically knows what their email is, so why do they need to select their own email here again? The system should explain that the fallback website does not know what the user’s email is because an email can have multiple recipients, and the user is completely anonymous to the system. This fits with Nielsen (1994)’s tenth heuristic “Help and documentation”.

Step 2: Click on ‘Decrypt’ - SUCCESS

While Step 2 succeeds, the displayed hints are confusing to the user. There should be an explanation on why these hints are necessary, namely, to let the user know which attribute they need to authenticate. Additionally, the word ‘credential’ can be seen as jargon. It is better to replace it with a more general term such as ‘detail’.

Task 5: View a cached email and download it

1. Go to ‘Email History’
2. Click on an email
3. Download the selected email

Step 1: Go to ‘Email History’ - SUCCESS

Step 1 succeeds but the label ‘Email History’ is unclear. It is better to rename it to ‘Mail Inbox’.

Step 2: Click on an email - SUCCESS

No remarks.

Step 3: Download the selected email - FAIL

Step 3 fails. For the evaluation question, “Will the user try to achieve the right result?” the answer was NO. The download and delete buttons are not familiar to everyone, especially those who are less used to technology. It is recommended to add a text label together with the symbol.

Task 6: Remove all cached emails and all IRMA attributes

1. Go to ‘Settings’
2. Click on ‘Delete all IRMA credentials’ and ‘Delete all cached emails’

Step 1: Go to ‘Settings’ - SUCCESS

No remarks.

Step 2: Click on ‘Delete all IRMA credentials’ and ‘Delete all cached emails’ - SUCCESS

No remarks.

While both steps pass, there are some general usability remarks. The buttons for deleting all IRMA credentials and cached emails should be colored red. The choice to delete everything has vast consequences and the gray colored button does not convey enough caution. When all IRMA credentials are deleted, feedback is immediately displayed as the table with credentials will become empty. However, this feedback is missing with the ‘Delete all cached emails’ button. It is recommended to show a label with the amount of emails that are currently saved. On deletion, this amount is changed to 0, to communicate to the user that all their emails have indeed been deleted. This is also stipulated by Nielsen (1994)’s first heuristic “Visibility of system status”.

5.1 Usability inspection conclusions

Our cognitive walkthrough has resulted in useful feedback which is summarized here. We noticed that usability concerns still remained, largely on the front of providing sufficient help and documentation, and also providing sufficient system feedback.

Firstly, the website should make use of notifications to announce when the user has performed a task, such as when the user has made a change to the settings, or when an email has been deleted. This gives users explicit feedback on their actions.

Secondly, users should be provided with more documentation on what certain concepts are. For example, they should receive explanation on IRMA when requested. Also, when the system prompts the user to select their own email in the case there are multiple recipients, they should be provided with information why this is necessary. Providing sufficient explanation also results in more transparency from the system.

Thirdly, the website would benefit from more user-friendly language. Terms such as ‘caching’ and ‘attributes’ should be replaced in favor of words that are easier to understand for the layman. It is also important to provide labels for buttons and symbols that are not immediately clear to the user, such as the download and delete buttons.

Lastly, while the website has minimalist design, which is in line with Nielsen (1994) 8th heuristic “aesthetic and minimalist design”, it could make use of more colors. The buttons should be colored to signify their importance. A delete button is usually red for instance. This communicates to the user that they need to pay extra attention on certain choices.

Chapter 6

Discussion

6.1 Current and new fallback

From our evaluation of the current fallback, it was found that despite providing the necessary feature of email decryption, there were significant shortcomings that made usage of the fallback inconvenient. We took these shortcomings in consideration while developing the functional prototype. The hassle of downloading an attachment was taken care of partially by implementing a link that contains the encrypted content in a URL. We also reduced the amount of IRMA authentication requests by caching the credentials for a limited period. Additionally, we implemented a way for users to re-read their decrypted emails. After performing a usability inspection on the prototype, it was concluded that users would largely have a decent time navigating the interface, but that usability issues still remained.

The usability issues largely boiled down to a lack of elaboration on concepts that could appear foreign to the user. For example, the implemented URL-feature has the possibility of confusing the user, as it replaces the process of downloading an encrypted attachment. The user is not sufficiently informed of the fact that the encrypted content is passed as a parameter in the URL. The prototype could also benefit from adopting more user-friendly language. For example, jargon such as ‘caching’ or ‘credentials’ should not be used as they are rather technical and risk confusing the user. We believe that after implementing the suggestions proposed during the usability inspection, the added functionality of the prototype has practical usage for the Postguard fallback.

6.2 Limitations

Our first limitation concerns that of the cognitive walkthrough. While our walkthrough was conducted by interns at Postguard, we would have liked to have experts, such as the developers, or the UI/UX designers, at Postguard present for the walkthrough too. We believe that their expertise would have led to additional useful feedback that was possibly missed by the interns. Additionally, since the usability inspection was only performed by those internally involved at Postguard, we must take into account that positive bias exists, as those involved in a product itself are more likely to have a positive view of the product.

To mitigate the risk of positive bias we would have ideally liked to conduct a qualitative and quantitative user test. However, this was not possible due to time constraints. This user test would serve the purpose of surveying the general public on the functional prototype, and thus gather more representative feedback on its features and issues.

Ideally, we would also like to implement the feedback from the usability inspections into the prototype. After implementing the feedback, it would have been beneficial to conduct another iteration of the usability inspection. However, because of the limited scope of the thesis and the limited time available, there was only room for one iteration. An additional iteration would have provided useful feedback and may have discovered new issues of its own.

6.3 Future work

For future work, we like to explore the limitations we previously discussed. Ideally, we would like to conduct a user test, implemented the suggestions from the usability inspections and the user test, and perform another usability inspection to measure the improvements. We believe this would result in useful feedback that could be incorporated into the fallback.

While Postguard makes use of identity-based encryption (IBE), there is also another encryption scheme known as attribute-based encryption (ABE) (Venema et al., 2022). IBE relies on the workings of a stringified identifier as explained in Section 2.4, which means that with each combination of attributes, a different IRMA authentication is required. With ABE it is possible to re-combine attributes client-side, thus reducing the need for IRMA re-authentication. For future work, it would be interesting to examine the specific advantages and disadvantages of ABE, and examine whether Postguard would benefit from an ABE encryption scheme.

Chapter 7

Conclusion

In this thesis, we have conducted a usability inspection on Postguard's fallback website and based on our findings, developed a functional prototype that has added features. We remark that it is a considerable effort to combine usability with security. We also conducted a usability inspection on the prototype. In conclusion, we hope that the Postguard fallback will be implemented on the website and serve as a stepping stone for users to consider adopting email encryption.

Bibliography

- Charles Arthur. DigiNotar SSL certificate hack amounts to cyberwar, says expert, 2 2017. URL <https://www.theguardian.com/technology/2011/sep/05/diginotar-certificate-hack-cyberwar>.
- Erinn Atwater, Cecylia Bocovich, Urs Hengartner, Edward Lank, and Ian Goldberg. Leading Johnny to Water: Designing for Usability and Trust. *Symposium On Usable Privacy and Security*, pages 69–88, 1 2015. URL https://cs.uwaterloo.ca/~uhengart/publications/soups15_1.pdf.
- Wei Bai, Doowon Kim, Moses Namara, Yichen Qian, Patrick Gage Kelley, and Michelle L. Mazurek. Balancing Security and Usability in Encrypted Email. *IEEE Internet Computing*, 21(3):30–38, 5 2017. doi: 10.1109/mic.2017.57. URL <http://dx.doi.org/10.1109/mic.2017.57>.
- Merel Brandon, Hanna Kathrin Schraffenberger, Wouter Sluis-Thiescheffer, Thea van der Geest, Daniel Ostkamp, and Bart Jacobs. Design Principles for Actual Security. *Adjunct Proceedings of the 2022 Nordic Human-Computer Interaction Conference*, 10 2022. doi: 10.1145/3547522.3547684. URL <http://dx.doi.org/10.1145/3547522.3547684>.
- Ben Brody and Mark Bergen. Google to Pay \$170 Million for YouTube Child Privacy Breaches, 9 2019. URL <https://www.bloomberg.com/news/articles/2019-09-04/google-to-pay-170-million-for-youtube-child-privacy-breaches>.
- Kyle Chin. Biggest Data Breaches in US History [Updated 2023] — UpGuard, 1 2023. URL <https://www.upguard.com/blog/biggest-data-breaches-us>.
- Nicholas Confessore. Cambridge Analytica and Facebook: The Scandal and the Fallout So Far, 11 2018. URL <https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html>.
- Sergej Dechand, Alena Naiakshina, Anastasia Danilova, and Matthew Smith. In Encryption We Don’t Trust: The Effect of End-to-End Encryption to the Masses on User Perception. *2019 IEEE European Symposium*

- on Security and Privacy (*EuroSecP*), 6 2019. doi: 10.1109/eurosp.2019.00037. URL <http://dx.doi.org/10.1109/eurosp.2019.00037>.
- Zak Doffman. Why You Should Quit WhatsApp As Critical New Update Confirmed, 3 2021. URL <https://www.forbes.com/sites/zakdoffman/2021/03/06/stop-using-whatsapp-after-facebook-apple-imessage-signal-and-telegram-privacy-backlash/>.
- Stephen Farrell. Why Don't We Encrypt Our Email? *IEEE Internet Computing*, 13(1):82–85, 1 2009. doi: 10.1109/mic.2009.25. URL <http://dx.doi.org/10.1109/mic.2009.25>.
- Simson L. Garfinkel, David Margrave, Jeffrey I. Schiller, Erik Nordlander, and Robert C. Miller. How to make secure email easier to use. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 4 2005. doi: 10.1145/1054972.1055069. URL <http://dx.doi.org/10.1145/1054972.1055069>.
- Shirley Gaw, Edward W. Felten, and Patricia Fernandez-Kelly. Secrecy, flagging, and paranoia. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 4 2006. doi: 10.1145/1124772.1124862. URL <http://dx.doi.org/10.1145/1124772.1124862>.
- Glenn Greenwald. Email service used by Snowden shuts itself down, warns against using US-based companies, 8 2013. URL <https://www.theguardian.com/commentisfree/2013/aug/09/lavabit-shutdown-snowden-silicon-valley>.
- Team Guild. A timeline of trouble: Facebook's privacy record and regulatory fines, 7 2022. URL <https://guild.co/blog/complete-list-timeline-of-facebook-scandals/>.
- Aurora Harley. Personas Make Users Memorable for Product Team Members, 2 2015. URL <https://www.nngroup.com/articles/persona/>.
- Yahsin Huang. Decentralized Public Key Infrastructure (DPKI): What is it and why does it matter?, 5 2019. URL <https://hackernoon.com/decentralized-public-key-infrastructure-dpki-what-is-it-and-why-does-it-matter-babee9d88579>.
- Identity-based encryption. Identity-based encryption, 5 2022. URL https://en.wikipedia.org/wiki/Identity-based_encryption.
- IRMA in detail, n.d. URL <https://privacybydesign.foundation/irma-explanation/>.
- Alita Joyce. How to Measure Learnability of a User Interface, 10 2019. URL <https://www.nngroup.com/articles/measure-learnability/>.

- Rendani Kruger, Jacques Brosens, and Marie Hattingh. A Methodology to Compare the Usability of Information Systems. *Lecture Notes in Computer Science*, pages 452–463, 2020. doi: 10.1007/978-3-030-45002-1\{-}39. URL http://dx.doi.org/10.1007/978-3-030-45002-1_39.
- Kendal Stephens LaFleur and Lei Chen. Email Encryption: Discovering Reasons Behind its Lack of Acceptance. *Proceedings of the International Conference on Security and Management*, pages 124–129, 1 2014. URL <http://works.bepress.com/lei-chen/60/>.
- Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Lawrence G. Roberts, and Stephen S. Wolff. The past and future history of the Internet. *Communications of the ACM*, 40(2):102–108, 2 1997. doi: 10.1145/253671.253741. URL <http://dx.doi.org/10.1145/253671.253741>.
- John Markoff. Data-Secrecy Export Case Dropped by U.S., 1 1996. URL <https://www.nytimes.com/1996/01/12/business/data-secrecy-export-case-dropped-by-us.html>.
- Jyothiikaa Moorthy. 23 Email Spam Statistics to Know in 2022, 8 2022. URL <https://www.mailmodo.com/guides/email-spam-statistics/>.
- Jakob Nielsen. Enhancing the explanatory power of usability heuristics. *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*, 1994. doi: 10.1145/191666.191729. URL <http://dx.doi.org/10.1145/191666.191729>.
- Jakob Nielsen. 10 Usability Heuristics for User Interface Design, 11 2020. URL <https://www.nngroup.com/articles/ten-usability-heuristics/>.
- Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '93*, 1993. doi: 10.1145/169059.169166. URL <http://dx.doi.org/10.1145/169059.169166>.
- Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*, 1990. doi: 10.1145/97243.97281. URL <http://dx.doi.org/10.1145/97243.97281>.
- Peter G. Polson, Clayton Lewis, John Rieman, and Cathleen Wharton. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, 36(5):741–773, 5 1992. doi: 10.1016/0020-7373(92)90039-n. URL [http://dx.doi.org/10.1016/0020-7373\(92\)90039-n](http://dx.doi.org/10.1016/0020-7373(92)90039-n).

- Cristina Pop. EU vs US: What Are the Differences Between Their Data Privacy Laws?, 9 2022. URL <https://www.endpointprotector.com/blog/eu-vs-us-what-are-the-differences-between-their-data-privacy-laws/>.
- ProPublica. How Facebook Undermines Privacy Protections for Its 2 Billion WhatsApp Users, 10 2021. URL <https://www.propublica.org/article/how-facebook-undermines-privacy-protections-for-its-2-billion-whatsapp-users>.
- Public-key cryptography. Public-key cryptography, 1 2023. URL https://en.wikipedia.org/wiki/Public-key_cryptography.
- Karen Renaud, Melanie Volkamer, and Arne Renkema-Padmos. Why Doesn't Jane Protect Her Privacy? *Privacy Enhancing Technologies*, pages 244–262, 2014. doi: 10.1007/978-3-319-08506-7_{-}13. URL http://dx.doi.org/10.1007/978-3-319-08506-7_13.
- Guardian Staff Reporter. Google records your location even when you tell it not to, 8 2018. URL <https://www.theguardian.com/technology/2018/aug/13/google-location-tracking-android-iphone-mobile>.
- Adrian Reuter, Ahmed Abdelmaksoud, Karima Boudaoud, and Marco Winckler. Usability of End-to-End Encryption in E-Mail Communication. *Frontiers in Big Data*, 4, 7 2021. doi: 10.3389/fdata.2021.568284. URL <http://dx.doi.org/10.3389/fdata.2021.568284>.
- Mark Ritson. Mark Ritson: Why you should fear the ‘digital duopoly’ in 2018, 10 2018. URL <https://www.marketingweek.com/ritson-digital-duopoly-2018/>.
- Max Roser. The Internet’s history has just begun, 10 2018. URL <https://ourworldindata.org/internet-history-just-begun>.
- Scott Ruoti, Nathan Kim, Ben Burgon, Timothy van der Horst, and Kent Seamons. Confused Johnny. *Proceedings of the Ninth Symposium on Usable Privacy and Security*, 7 2013. doi: 10.1145/2501604.2501609. URL <http://dx.doi.org/10.1145/2501604.2501609>.
- Kim Salazar. How to Conduct a Cognitive Walkthrough Workshop, 4 2022. URL <https://www.nngroup.com/articles/cognitive-walkthrough-workshop/>.
- David Sanger and Nicole Perlroth. Encrypted Messaging Apps Face New Scrutiny Over Possible Role in Paris Attacks, 11 2015. URL <https://www.nytimes.com/2015/11/17/world/europe/encrypted-messaging-apps-face-new-scrutiny-over-possible-role-in-paris-attacks.html>.

- S Sheng, C Koranda, J Hyland, and L Broderick. Proceedings of the Second Symposium on Usable Privacy and Security. 2006.
- D. K. Smetters and R. E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. *Proceedings of the 2002 workshop on New security paradigms*, 9 2002. doi: 10.1145/844102.844117. URL <http://dx.doi.org/10.1145/844102.844117>.
- Dag Spicer and Raymond Tomlinson. Interviews: Raymond Tomlinson: Email Pioneer, Part 1. *IEEE Annals of the History of Computing*, 38(2): 72–79, 2016. doi: 10.1353/ahc.2016.0024. URL <http://dx.doi.org/10.1353/ahc.2016.0024>.
- Statista. Number of e-mails per day worldwide 2017-2025, 9 2022. URL <https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/>.
- Christian Stransky, Oliver Wiese, Volker Roth, Yasemin Acar, and Sascha Fahl. 27 Years and 81 Million Opportunities Later: Investigating the Use of Email Encryption for an Entire University. *2022 IEEE Symposium on Security and Privacy (SP)*, 5 2022. doi: 10.1109/sp46214.2022.9833755. URL <http://dx.doi.org/10.1109/sp46214.2022.9833755>.
- Heinz Tschabitscher. Why Are Email Files so Large?, 6 2021. URL <https://www.lifewire.com/what-is-the-average-size-of-an-email-message-1171208>.
- Jovi Umawing. Labs survey finds privacy concerns, distrust of social media rampant with all age groups, 3 2019. URL <https://www.malwarebytes.com/blog/news/2019/03/labs-survey-finds-privacy-concerns-distrust-of-social-media-rampant-with-all-age-groups>.
- T. Van Vleck. Electronic Mail and Text Messaging in CTSS, 1965-1973. *IEEE Annals of the History of Computing*, 34(1):4–6, 1 2012. doi: 10.1109/mahc.2012.6. URL <http://dx.doi.org/10.1109/mahc.2012.6>.
- Marloes Venema, Greg Alpár, and Jaap-Henk Hoepman. Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. *Designs, Codes and Cryptography*, 91(1):165–220, 9 2022. doi: 10.1007/s10623-022-01093-5. URL <http://dx.doi.org/10.1007/s10623-022-01093-5>.
- Cathleen Wharton, John Rieman, Clayton Lewis, and Peter G. Polson. The cognitive walkthrough method: a practitioner’s guide. *John Wiley Sons, Inc. eBooks*, pages 105–140, 6 1994. URL <https://dl.acm.org/citation.cfm?id=189214>.

- Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. *USENIX Security Symposium*, page 14, 8 1999. URL <http://css.csail.mit.edu/6.858/2014/readings/johnny.pdf>.
- Justin C.Y. Wu and Daniel Zappala. When is a Tree Really a Truck? Exploring Mental Models of Encryption. *Symposium On Usable Privacy and Security*, pages 395–409, 1 2018. URL <https://www.usenix.org/system/files/conference/soups2018/soups2018-wu.pdf>.
- Carl Youngblood. An Introduction to Identity-based Cryptography. Technical report, 3 2005. URL https://courses.cs.washington.edu/courses/csep590/06wi/finalprojects/youngblood_csep590tu_final_paper.pdf.

Appendix A

Cognitive walkthrough of current fallback

Task 1: Download the encrypted file and decrypt it

1. Download the encrypted file and decrypt it
2. Click on the link in the email for people without an add-on
3. Upload the postguard.encrypted file
4. Install the IRMA app from the App Store
5. Open the IRMA app
6. Navigate to 'Adding cards'
7. Choose 'email address' and click on 'Add'
8. Fill in your email address
9. Click on the confirmation link in the email you received
10. Scan the QR code with the IRMA app
11. Fill in the pairing code
12. Click on 'Yes'
13. Scan the QR code on the fallback website
14. Click 'Yes'
15. View decrypted email

What is the IRMA app?

The IRMA app is a general-purpose privacy-friendly authentication app. You need the IRMA app with PostGuard to prove that you really are the intended recipient.

[More information about IRMA](#)

Download the free IRMA app:

[App store](#) [Google Play](#)

One attachment • Scanned by Gmail ⓘ



Step 1

You received a PostGuard encrypted email from

Esther Shi

There are two ways to decrypt and read this protected email.

- 1 You can use the free PostGuard add-on that is available (currently only) for Outlook and Thunderbird. It is available via www.postguard.eu. After installation, you can not only decrypt and read the current email but also all future PostGuarded emails. Moreover, you can easily send secure emails yourself with the PostGuard add-on.

Already installed PostGuard? Click on the "Decrypt Email" button.

This button can be found on the right side of the above menu.

- 2 You can also decrypt and read this email via the fallback website www.postguard.eu/decrypt/. This website only decrypts the mail, and cannot be used for replying. PostGuards work better with its add-on than with this website.

Step 2



File sharing | Home | Emailing | [Decrypt emails](#) | About | Privacy Policy

Download the "postguard.encrypted" file that is attached to the encrypted email you received. Next, add the file here.

Choose file | postguard.encrypted



← Download add-on ⓘ

○○○●○○

Step 3

22:00

< Search



IRMA authentication

Privacy by Design Foundation

OPEN



45 RATINGS

3.0

★★★★☆

AGE

4+

Years Old

CHART

No. 27

Utilities

DEV

Privacy

What's New

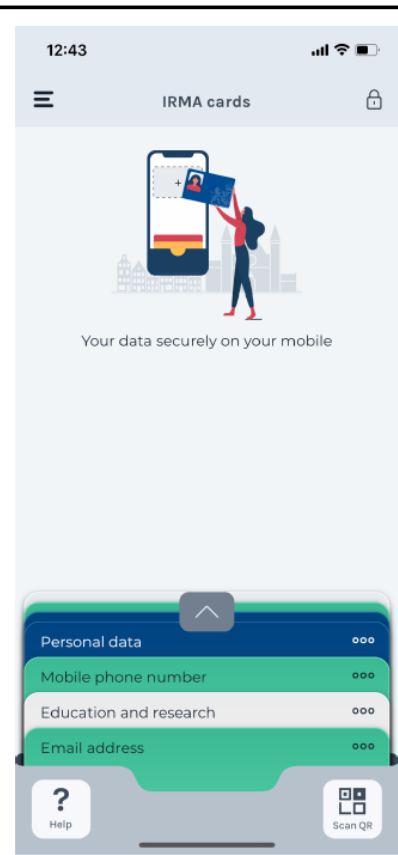
[Version History](#)

Version 6.3.2

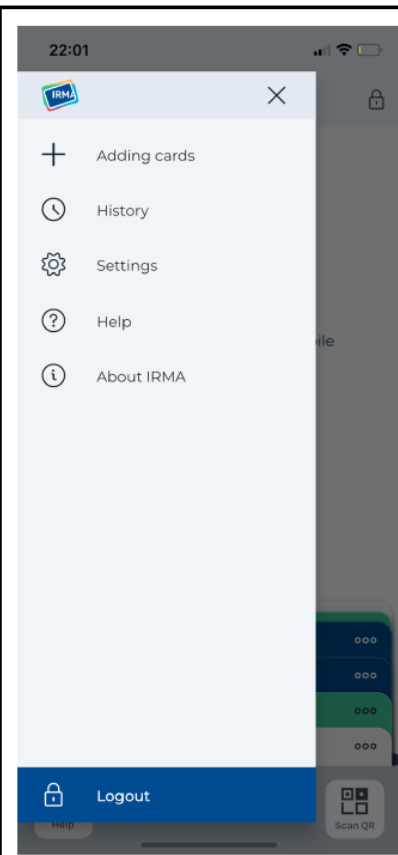
2mo ago

- Fixed that app crashed on iOS12 devices due to

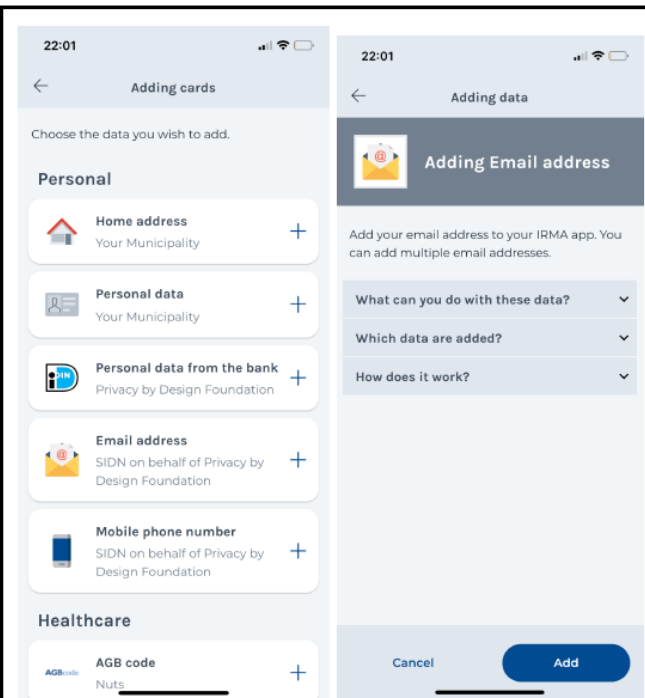
Step 4



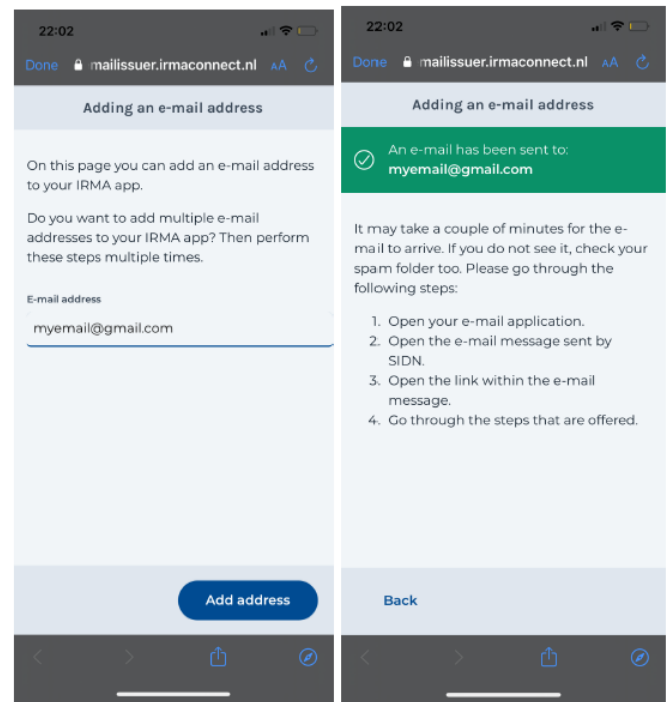
Step 5



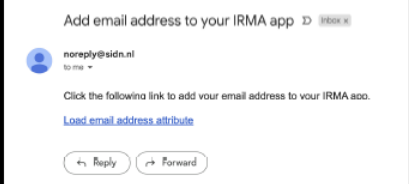
Step 6



Step 7



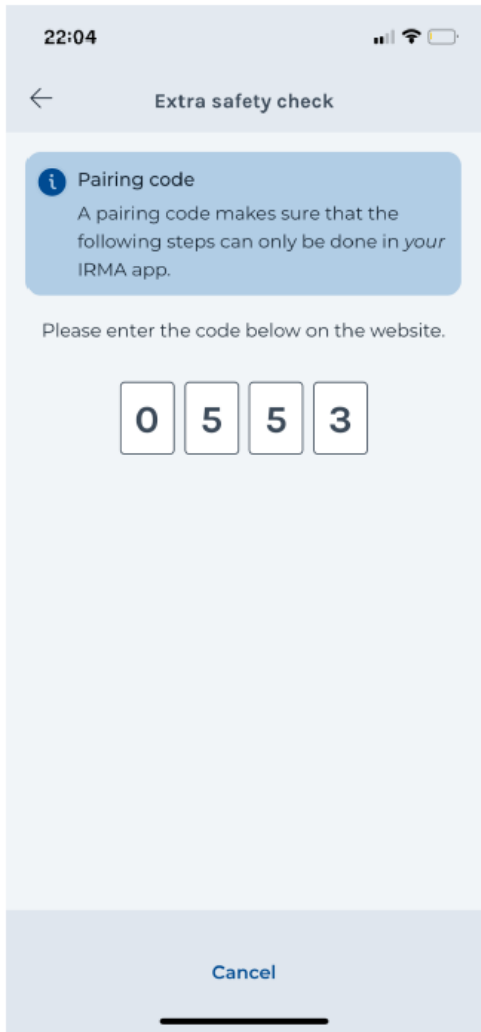
Step 8



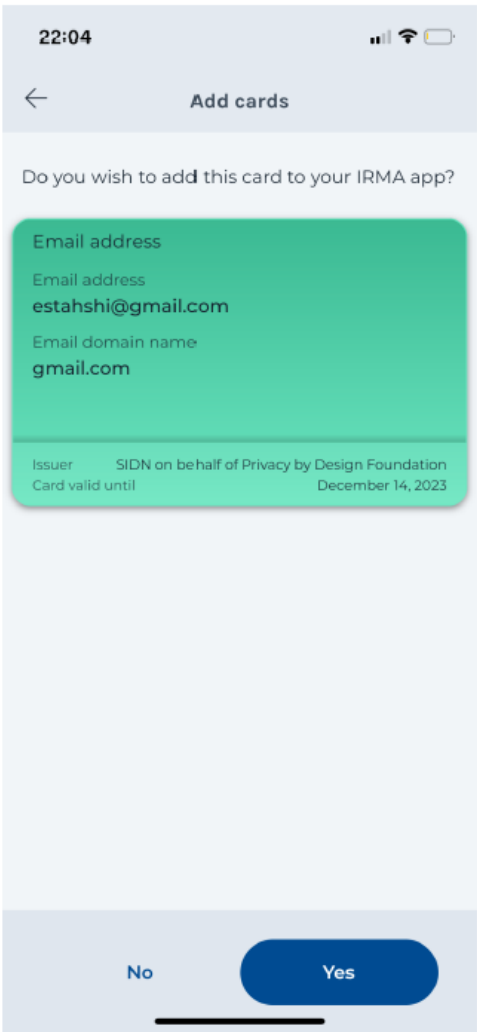
Step 9



Step 10



Step 11



Step 12



Step 13

10:45

←

Make yourself known

Do you wish to disclose the following data to ihub.ru.nl?

Email address

esthrshi@gmail.com

Card

Issuer

Email address

SIDN on behalf of Privacy by Design ...

No

Yes

Step 14

Message decrypted successfully

Choose file

postguard-email1.encrypted

From

esthrshi@gmail.com

To

esthrshi@gmail.com

Date

Sun Nov 13 21:01:31 2022

Subject

this is a test

encrypted wooh

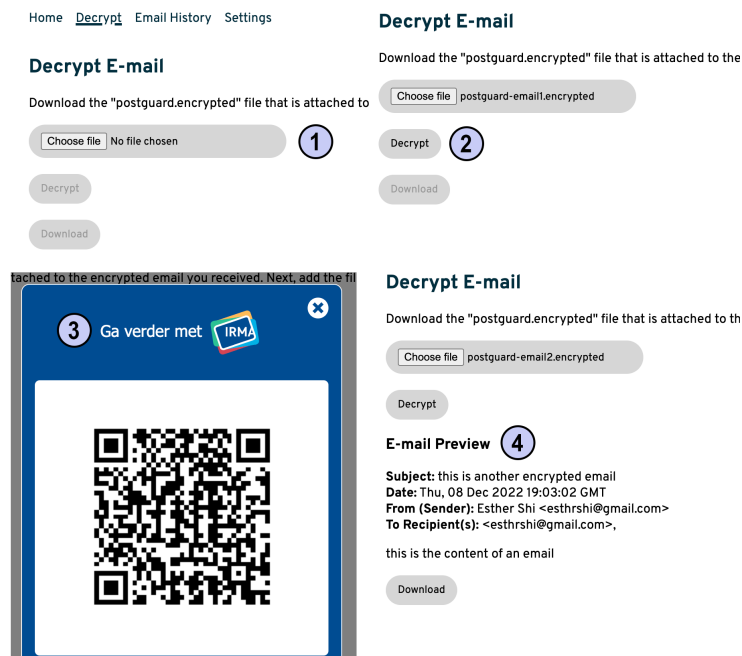
Step 15

Appendix B

Cognitive walkthrough of prototype fallback

Task 1: Download the encrypted file and decrypt it

1. Add the postguard.encrypted file
2. Click on 'Decrypt'
3. Perform IRMA authentication
4. View email



Task 2: Turn on email- and IRMA caching in settings

1. Go to 'Settings'
2. Check 'Cache my emails' and 'Cache my IRMA credentials'

Home Decrypt Email History Settings ①

Settings


All IRMA credentials and decrypted e-mails are cached locally in the user's browser, no information is sent to a server.

Caching

☒ Cache my emails ②

☒ Cache my IRMA credentials ③

IRMA Credentials

Credentials	Expiry date
esthrshi@gmail.com	Tue, 20 Dec 2022 04:00:00 GMT 

Delete all IRMA credentials

Email History

Delete all cached emails

Task 3: Decrypt another email (same credentials as previous one) with the URL

1. Click on the link in the email for people without an add-on (URL version)
2. Click on 'Decrypt'

You received a PostGuard encrypted email from

Esther Shi

There are two ways to decrypt and read this protected email.

- 1 You can use the free PostGuard add-on that is available (currently only) for Outlook and Thunderbird. It is available via www.postguard.eu. After installation, you can not only decrypt and read the current email but also all future Postguarded emails. Moreover, you can easily send secure emails yourself with the PostGuard add-on.

Already installed PostGuard? Click on the "Decrypt Email" button.

This button can be found on the right side of the above menu.

- 2 Please click on [this](#) link to decrypt the e-mail via the Postguard fallback site.

Step 1

Home [Decrypt](#) Email History Settings

Decrypt E-mail

Encrypted file detected in URL

Decrypt

Download

Step 2

Task 4: Decrypt another email with multiple recipients and different credentials as previous one

1. Select the email address that belongs to the recipient
2. Click on 'Decrypt'

Home [Decrypt](#) Email History Settings

Decrypt E-mail

Download the "postguard.encrypted" file that is attached to the encrypted email you received. Next, add the file here.

postguard-everything.encrypted

Please select which email belongs to you:

1

You selected

Please select which email belongs to you:

estahshi@gmail.com
✓ esther.shi@ru.nl
esthrshi@gmail.com
You selected esther.shi@ru.nl

Home [Decrypt](#) Email History Settings

Decrypt E-mail

Download the "postguard.encrypted" file that is attached to the encrypted email you received. Next, add the file here.

postguard-everything.encrypted

Please select which email belongs to you:

esther.shi@ru.nl

You selected esther.shi@ru.nl

Your credentials:

Mobile number: +3164094****

2

Task 5: View a cached email and download it

1. Go to 'Email History'
2. Click on an email
3. Download the selected email

Home Decrypt Email History Settings

E-mail History

test all creds
Esther Shi
Tue, 22 Nov 2022 11:08:10 GMT



this is a test
Esther Shi
Sun, 13 Nov 2022 21:01:31 GMT

this is a test
Esther Shi
Sun, 13 Nov 2022 21:01:31 GMT

this is a test
Esther Shi
Sun, 13 Nov 2022 21:01:31 GMT

this is a test
Esther Shi
Sun, 13 Nov 2022 21:01:31 GMT

test all creds
From: Esther Shi <esthrshi@gmail.com>
To: <esthrshi@gmail.com>,
Date: Tue, 22 Nov 2022 11:08:10 GMT
ajbfldfjb sdvafdf a mdsfv,sf eeeee



Task 6: Remove all cached emails and all IRMA attributes

1. Go to 'Settings'
2. Click on 'Delete all IRMA credentials' and 'Delete all cached emails'

[Home](#) [Decrypt](#) [Email History](#) [Settings](#)

1

Settings

All IRMA credentials and decrypted e-mails are cached locally in the user's browser, no information is sent to a server.

Caching

- ☒ Cache my emails
- ☒ Cache my IRMA credentials

IRMA Credentials

Credentials	Expiry date
esthrshi@gmail.com	Tue, 20 Dec 2022 04:00:00 GMT 

Delete all IRMA credentials

2

Email History

Delete all cached emails

4

127.0.0.1:5173 says

Are you sure you want to delete all IRMA credentials? This action is permanent!

Cancel

OK

3

127.0.0.1:5173 says

Are you sure you want to delete all emails? This action is permanent!

Cancel

OK

5