PostGuard: Towards easy and secure email communication

Leon Botros
Radboud University
Nijmegen, The Netherlands
l.botros@cs.ru.nl

Merel Brandon
Radboud University
Nijmegen, The Netherlands
merel.brandon@ru.nl

Bart Jacobs
Radboud University
Nijmegen, The Netherlands
bart@cs.ru.nl

Daniel Ostkamp
Radboud University
Nijmegen, The Netherlands
daniel.ostkamp@ru.nl

Hanna Schraffenberger
Radboud University
Nijmegen, The Netherlands
hanna.schraffenberger@ru.nl

Marloes Venema
Radboud University
Nijmegen, The Netherlands
marloes.venema@ru.nl

ABSTRACT

This paper presents PostGuard: a secure email encryption solution for communication in the private and public sectors. The novelty of PostGuard lies in the combination of identity-based encryption with a digital identity wallet. Within this setting, senders specify who should be able to read/decrypt their emails in terms of the recipient's attributes (e.g., their name and/or email address). Subsequently, recipients use the digital identity wallet app Yivi to prove that they possess these attributes to decrypt the mails. Thus, PostGuard reduces decryption to authentication. The underlying mental model is: to see confidential information, you need to prove that you are the intended recipient. The main contribution of this paper is the working prototype of PostGuard for Outlook and Thunderbird. This paper describes the concept, setup, implementation, and design of PostGuard and discusses current limitations and plans for future work.

CCS CONCEPTS

• **Security and privacy → Usability in security and privacy**.

KEYWORDS

Email encryption, user experience design, cryptography, attribute-based identity management, Yivi, IBE, ABE

ACM Reference Format:
Leon Botros, Merel Brandon, Bart Jacobs, Daniel Ostkamp, Hanna Schraffenberger, and Marloes Venema. 2023. PostGuard: Towards easy and secure email communication. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3544549.3585622

1 INTRODUCTION

A major cause of privacy breaches is human error [19]. For instance, looking at two years of data breaches in the UK, Ingham [16] has reported that 88% of incidents were not the result of malicious actors but rather caused by human mistakes. A particularly common error was sending sensitive data to the wrong recipient, explaining 37% of the reported breaches. As the frequency of such incidents shows, users need better means to make sure sensitive data can be read only by the *intended recipients*. In recent years, this need to protect sensitive data from falling into the wrong hands has furthermore been reinforced by stringent requirements – for instance, for electronic communication between doctors and patients — in data protection regulations (like the GDPR) and in the light of systematic mass surveillance activities of internationally operating intelligence agencies.

From a purely technical perspective, a solution for secure communication exists in the form of public key cryptography [10]. However, cryptography (and especially encryption) also imposes the problem of key management: how to make sure that people can easily (and securely) obtain and maintain cryptographic keys for encryption and decryption. From a stern security perspective, the preferred method in public key cryptography is that users themselves generate private-public key pairs on their own devices and link their public key to their identity, typically via a Certificate Authority or via a public key server. This is so complicated that it works only for specialists and has been a show-stopper for wide-scale adoption [14, 29].

The PostGuard project addresses the fundamental societal problem that email encryption as a technique has been available for decades but has never been widely adopted, in large part because of key management challenges. PostGuard aims to overcome prevailing usability issues of email encryption solutions by (1) leveraging the inherent usability benefits of Identity-Based Encryption (IBE) and (2) utilizing attribute-based identity management (and, in particular, the existing, separate identity wallet app Yivi) for authentication within the IBE system. In our setting, users do not have to manage encryption keys themselves. Instead, to encrypt an email, the sender specifies who should be able to read/decrypt their emails in terms of the recipient's attributes (e.g., their name, email address, and/or mobile number) via a dedicated access management interface (see Figure 3). To decrypt the email, the recipient subsequently uses the identity management app Yivi to prove that they possess these attributes and thus are the intended recipient (see Figure 4). Key management and encryption happen under the hood and rely on a Trusted Third Party (TTP) (see 3.1 for details).

We are currently developing PostGuard with support from several stakeholders in the public and private sectors. The main contribution of this paper is the working prototype of an email encryption tool for Outlook and Thunderbird. To the best of our knowledge,

CHI EA '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9422-2/23/04.
https://doi.org/10.1145/3544549.3585622

this prototype is the first that combines identity-based encryption with a digital identity wallet. With the proposal of the European Commission for a European Digital Identity Wallet [11][1], such mobile wallet apps have become particularly relevant for the HCI community. We are currently conducting a usability study of PostGuard, and plan to report the results in a follow-up paper.

2 BACKGROUND AND RELATED WORK

More than 40 years ago, Diffie and Hellman [10] paved the way for secure electronic communication with the invention of public key cryptography. By now, various tools for the encryption of electronic documents exist. Encrypting emails, for instance, is part of the S/MIME standard, which is supported in popular email clients such as Outlook and Apple Mail. Similarly, the security tool PGP ("pretty good privacy") allows for encrypting emails and other types of files. However, these tools are barely used in practice [32].

The poor adoption of secure communication tools has many reasons, such as incorrect mental models of secure communication and a poor understanding of the fundamental concept of end-to-end encryption (E2EE) among potential users [2]. Furthermore, many potential users do not trust encryption [9], are unaware that encryption tools exist [23] or do not know that sensitive communication (like sending passport copies) should not happen via standard, unsecured email. In addition, email encryption tools have repeatedly been shown to suffer from a variety of usability and user interface issues [e.g., 12, 14, 18, 23, 31, 34] that can get in the way of (correct) usage. One of the biggest challenges concerning usability is how to facilitate *usable key management* [29].

2.1 The key management problem

To encrypt emails, users need a public/private key pair. Obtaining and managing these keys is complex, time-consuming, and error-prone [17, 27], even more so because keys also need to be linked to their legitimate owners. PGP, for instance, expects users to verify, sign and thus endorse the keys of other users, thereby establishing a so-called "Web of Trust" [33].[2] In contrast, S/MIME, requires users to obtain certificates from trusted certificate authorities that verify the user's identity and bind their keys to them — a complex and time-consuming process that drives away users [14].[3]

Because of the usability problems with key management, researchers have started to explore hiding key management details from users [29] and automating, e.g., the generation, uploading, and downloading of necessary keys [e.g., 4, 5, 17]. Furthermore, compromises between usability and security, such as trusting keys on first use but detecting and flagging subsequent changes ("key continuity management"), have been studied [14, 15]. However, with these new developments, new worries have started to emerge. Specifically, it has been found that hiding key management from users can inhibit them from understanding how the system works, increasing the risk of mistakes and causing a lack of trust in the system [e.g., 17, 24, 28]. Furthermore, even though key management automation brings benefits, most usability issues around key management remain unaddressed [29]. Open issues include, e.g., key verification (making sure the keys belong to the right person), key revocation (revoking lost or compromised keys and moving on to new keys), managing more than one key pair (e.g., for multiple email addresses), backup of keys, synchronization of keys between different devices, and users' understanding of key-based systems [29].

2.2 The mental model mismatch

Key management introduces a complexity that – judging from existing research into peoples' mental models of encryption – does not align with preconceived notions that average users have about encryption/decryption.

Insights into how users expect encryption to work can be found in the work by Wu and Zappala [35], who (via interviews and a diagramming exercise) identified four mental models of encryption. While these vary in complexity, all identified models see encryption as a form 'access control'. The most simple model entirely reduces encryption to access control and, e.g., does not include any notions of data being transformed or keys being used. In contrast, more complex models include the notion of a 'shared secret' or key that is needed. However, these models coincide well with a *symmetric* encryption model, where (rather than dealing with public/private key pairs) one shared secret/key is used for encryption and decryption. These observations lead Wu and Zappala [35, p. 404] to conclude that "getting users to understand public and private keys–even from a functional perspective–seems an uphill battle" and suggest that the "[...] common perception of encryption as access control can be useful in the right contexts. Because it was shared by all our participants, even those with the most simple mental models, it can serve as a lowest common denominator model off which to build, and is likely a useful and intuitive abstraction in certain use cases".

More evidence for an 'access control' mental model is provided by Abu-Salma et al. [1], who found that half to two-thirds of their 125 respondents in a survey had partially correct mental models of E2EE – in particular "that it prevents third-party access and/or limits access to just the sender and recipient" (p. 7).[4]

While it would be rushed to assume that the general population shares an 'access control' mental model at large,[5] we believe the approach based on access control is worth pursuing in the context of email encryption. Thus, PostGuard reduces encryption to identifying who should be able to access the data and consequently reduces decryption to authentication. To this end, PostGuard builds on two technologies that (when combined), can achieve this well: identity-based encryption and attribute-based identity management.

2.3 Identity-based encryption

An important technical breakthrough towards usable encryption has been Identity-Based Encryption (IBE) – originally proposed by Shamir [30] but matured since Boneh and Franklin [7] and Boneh

[1] Yivi (formerly known as IRMA) has been a prime example for these plans.

[2] This can happen, e.g., at so-called physical 'Keysigning Parties' where users can check each other's identities and keys physically.

[3] What is more, self-signed certificates and certificates by untrusted/unknown authorities leave users with the difficult decision of whether or not to trust the certificates [20].

[4] However, somewhat contradictory, only one-quarter of the respondents believed that no one aside from the sender and recipient could access the communications with their hypothetical E2EE tool.

[5] Based on interviews with 60 participants, Abu-Salma et al. [2] report somewhat different mental models. Participants, e.g., mistake encryption with the encoding of data, describe it as something that makes conversations 'invisible' or that transforms a message into random text or as a special language.

and Boyen [6]. Here, a public key can be derived automatically using a master public key together with some publicly available identifier of the intended recipient, like an email address or a national identification number. Deriving these end-user public keys can happen by the email client, making sure that an end-user only needs to know the recipient's email address to send an encrypted message. The owner of the target email address (the recipient of the encrypted message) can obtain the corresponding private key from a trusted key server, acting as a Trusted Third Party (TTP) that hosts the Private-Key Generator (PKG). This is a major step forward since people no longer have to generate keypairs themselves.

2.4 Attribute-based identity management

The use of IBE in itself only solves half of the key management problem, since people still have to be able to reliably prove their identity to the TTP to claim their private key needed to decrypt messages. This is where the mobile app Yivi [22] (formerly known as IRMA [21]) comes in – as an open source platform for attribute-based authentication.[6] This app takes the form of a digital wallet, which can be filled with verified (and signed) information about the user (e.g., name, address, date of birth, phone number and email address), by obtaining these from trusted issuers (e.g., a municipality).[7] Users can then use this verified information, so-called *attributes*, to reliably prove that a name, address, date of birth, phone number, email address, etc., is theirs. After doing so, they can obtain the corresponding private key from the TTP to decrypt messages that have been encrypted for them using the public keys corresponding to these attributes.

The PostGuard project thus leverages Yivi to reduce decryption to authentication. The underlying mental model is: in order to see confidential information, you need to prove that you really are the intended recipient. This is the core idea behind PostGuard – and it also sets PostGuard apart from existing IBE systems, such as PWN [28] and its' two successors PWN 2.0 [25] and MessageGuard [26]. Both PWN and PWN 2.0 use a form of email-based authentication [13], where users do not have to do anything to authenticate themselves – access to their emails is all they need. MessageGuard modularly supports different key management schemes. Its IBE setting also leverages email-based authentication but requires users to first create an account before encrypting and decrypting emails.

3 POSTGUARD

In the following, we describe the general setup, the implementation, and the design of PostGuard.

3.1 General setup and flow

The PostGuard setup and session flow are depicted in Figure 1, describing a scenario where Alice sends an encrypted email to Bob, who subsequently decrypts the message. We assume that Bob already has the Yivi app (filled with the necessary attributes) and that both Alice and Bob have installed the PostGuard add-on in advance. (However, Bob can also install the tools upon receiving encrypted emails.) We furthermore assume that the TTP runs the PKG and a Yivi server.

During the PKGs initial setup phase, it generates a *master public key (mpk)* and *master secret key (msk)* ⓪ . When users subsequently install PostGuard, the add-on obtains the *mpk* from the PKG without user interaction ① . To send an encrypted mail, Alice specifies the identity of the recipient ② . Her PostGuard add-on uses the *mpk* and this identity of the recipient to encrypt the email message, which is then sent to Bob ③ . When Bob receives the mail, his PostGuard add-on requests the *user secret key (usk)* from the PKG ④ . The PKG subsequently initiates a disclosure session at the Yivi server that enables Bob to prove his identity ⑤ . The session information is passed on to Bob, via a QR code displayed by his add-on ⑥ . Bob scans the QR code with the Yivi app and discloses the attributes that fit the specified identity to the Yivi server. (With this, Bob authenticates himself as the intended recipient) ⑦ . The PKG detects whether Bob successfully disclosed the requested attributes to the Yivi server ⑧ . If so, it uses the *msk* and Bob's identity to extract the *user secret key (usk)* for the specified identity and sends the resulting *usk* to Bob's PostGuard add-on ⑨ . Finally, the PostGuard add-on decrypts the ciphertext using the *usk* and displays the plain text to Bob ⑩ .

3.2 Implementation

We have implemented a proof of concept that illustrates that this setup is practical. Our implementation is modular and consists of several components. For end-users, we have developed a *Thunderbird add-on* and an *Outlook add-on* to encrypt and decrypt emails. In addition, we offer a *fallback website* (see https://postguard.eu) that allows users to decrypt emails without an add-on. At the core, we have developed libraries for encrypting data streams of arbitrary length for an identity using IBE. A key component within any IBE system is the *Private Key Generator*, a server that is responsible for providing keys to clients. Even though the PKG uses Yivi as the primary authentication method since it closely integrates with IBE, we enable the community to implement other authentication modules which might suit their needs. The core libraries and PKG were developed in Rust. As a result, we can compile the client library to WebAssembly (WASM), which can run in a sandboxed environment in most modern browsers, providing a performant and easy-to-use interface for client-side encryption in web applications. All components are open source and available on GitHub (https://github.com/encryption4all).

By putting these components together and hosting our own Yivi server, we have a basic infrastructure for sending and receiving encrypted emails using PostGuard. Running it only requires keeping these servers (and plugins) active and up to date. However, development is ongoing, and the current implementation still has limitations (see section 4).

[6] The Yivi app was originally called *IRMA* and grew out of earlier research at Radboud University. It is now being rolled out via the spin-off foundation Privacy by Design (https://privacybydesign.foundation) and SIDN (https://www.sidn.nl), two cooperating non-profit foundations. The app is a precursor of the newly proposed 'European Digital Identity Wallet'.

[7] For Yivi, this information is only stored locally on the user's phone. It is not shared with the foundations behind Yivi and is not stored 'in the cloud'.

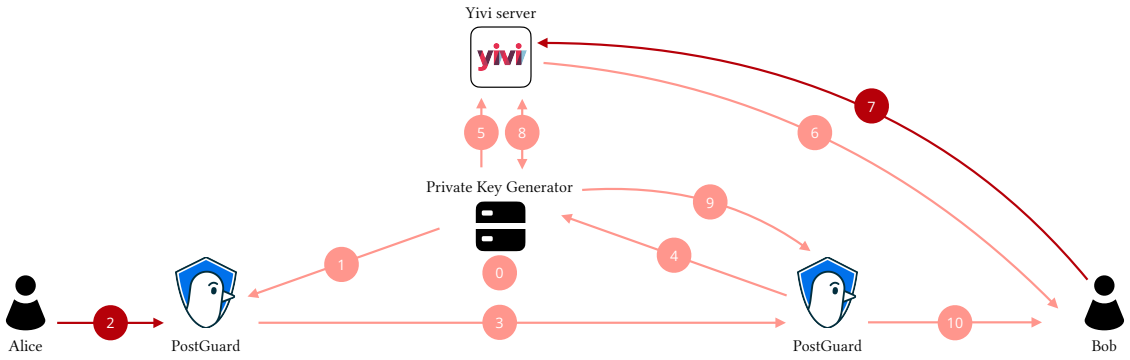

Figure 1: General setup and flow. User actions are depicted in dark red. Automated actions are depicted in a lighter red.

3.3 Design

PostGuard is designed iteratively, using input from five project partners in the domains of health, education, government and cybersecurity. In addition, four external *user experience experts* and three external *behavioural science/design for behavioural change experts* have inspected an earlier clickable prototype of PostGuard (and other email encryption tools) with a heuristic evaluation, as part of our effort to formulate design principles for actual security [8]. Consequently, we have used their feedback on PostGuard and the resulting general principles to guide the design of PostGuard. (Feedback from potential users is currently obtained with usability tests.)

In order to allow users to integrate encryption into their existing email workflows, we offer add-ons for popular email clients. By now, we have a first functional prototype for Thunderbird (which is customizable and allows us to design and implement the tool as envisioned) and Outlook (which comes with more constraints but is much more commonly used). The two add-ons work slightly differently when it comes to encrypting emails. In the following, we focus on the Thunderbird version, which resembles our vision more closely.

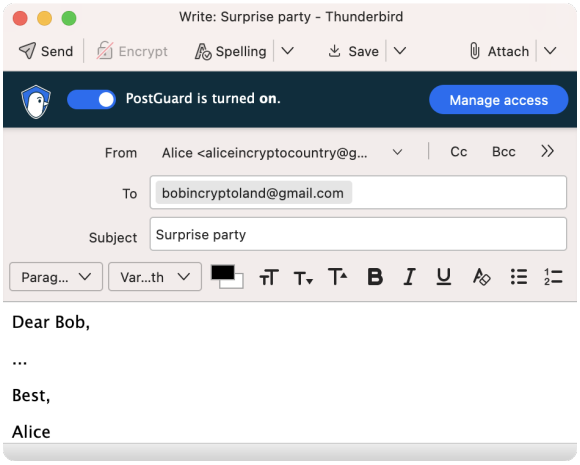

Figure 2: PostGuard in Thunderbird (switched on).

Encryption. When composing a mail with Thunderbird, PostGuard encryption can be toggled on and off in the PostGuard bar that has been added to the compose window (see Figure 2). When encryption is turned on, PostGuard automatically uses the recipient's email address as the identity for encryption. This means that aside from toggling on encryption, the sender has to take no special action to send an encrypted email.

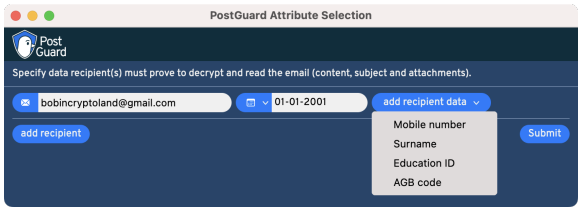

Figure 3: Attribute selection process.

If desired, the sender can optionally choose *additional attributes* of the recipient and, e.g., specify the recipient's name and/or birthday (or other properties) in addition to the email address. This can be done in the access management window (see Figure 3), which opens when a user clicks 'manage access' in the top right of the PostGuard bar. For each recipient, the email address is automatically added as an attribute for encryption. The sender then can select and specify additional properties/attributes of the recipients. This can prevent data breaches when an email is accidentally sent to the wrong person (see section 1) or when a mailbox is accessed by someone who is not the recipient. In such cases, the unintended recipient cannot disclose the additional attributes and is not be able to decrypt the mail (see below).

Decryption. Figure 4 illustrates the decryption process. When the recipient chooses to decrypt an encrypted mail, a pop-up from PostGuard opens with a QR code (1). Next, the recipient needs to open their Yivi app and scan the QR code (2), agree to disclose all of the requested attributes (3), and upon doing so, the mail is decrypted by the add-on (4).

Note that to decrypt a message, mere *knowledge* of the values of the required attributes is not enough. Just like a person needs to *show* their ID card to pick up a package at the local post office, users need to show/disclose their Yivi attributes to prove they *are*

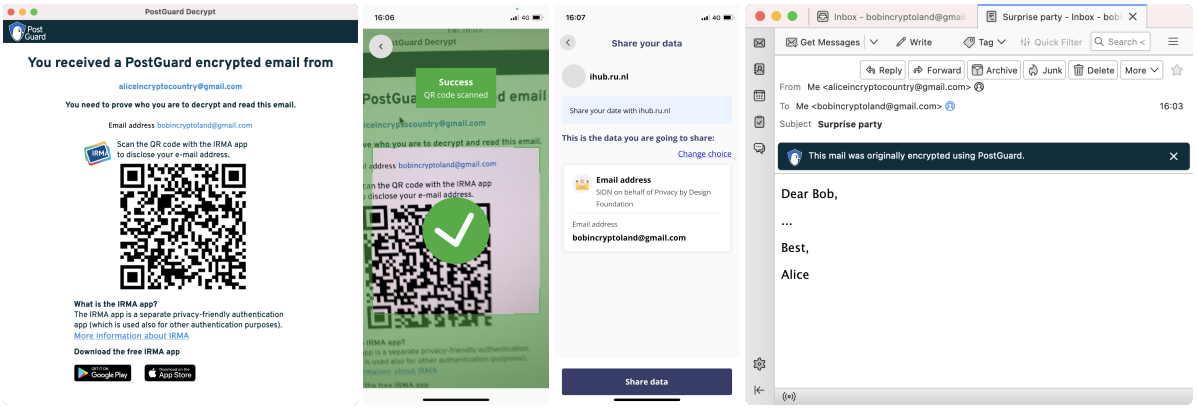

Figure 4: Decrypting emails with PostGuard and the Yivi app (formerly known as IRMA)

the intended recipient (e.g., show that they *have* a certain name or mobile number rather than that they *know* it).

As scanning the QR code to disclose one's attributes for every single encrypted mail can quickly become annoying, we have taken measures, so users only need to do this once every 24 hours. This trade-off between usability and security will be fine-tuned based on user feedback, and users will be able to change this in their PostGuard settings.

4 DISCUSSION, STATUS, AND FUTURE WORK

To the best of our knowledge, PostGuard is the first IBE solution that integrates an identity wallet for authentication, not only technically but also conceptually. Thanks to this, neither the sender nor the recipient is bothered with managing encryption keys. Instead, users manage personal attributes (e.g., their name, date of birth, or email address) in a separate digital wallet app (similar to how identity cards and driver's licenses are kept in physical wallets). In this sense, PostGuard has re-defined the complex *key management* problem as an *identity management* problem, which can easily be solved with Yivi (or other future identity wallets). Furthermore, users need no complex mental model of encryption to understand and use PostGuard. Rather, they are confronted with the intuitive tasks of specifying who should be able to read the emails they send and proving that they are eligible to read the emails they receive. Thus, PostGuard has shifted the focus from the foreign and difficult-to-grasp concept of *encryption* to the more intuitive concept of *access control* and, similarly, reduced *decryption* to the more familiar concept of *authentication*.

The major goal of this project is to make email encryption user-friendly. We are currently conducting usability tests. As part of this, we are investigating how the integration of an identity wallet app in an identity-based encryption process affects usability, user understanding, trust, adoption, and perceived security. In particular, we are interested in whether our system indeed aligns well with users' mental models. A question to explore in this context is how to deal with the possibility of decrypted emails being forwarded, as this might challenge the idea that only intended recipients can read a mail. Furthermore, we are looking into how using Yivi and a TTP affects trust (e.g., are users cautious of disclosing their information to the TTP?), and how the additional steps required to set up PostGuard impact usability (e.g., are users willing to accept the effort and time needed for encryption and decryption?). In the future, we plan to also compare PostGuard to existing alternatives (e.g., PGP and S/MIME, PWN 2.0) in terms of usability and security. Also, more technical work on the threat model, the trust/security model and PostGuards security properties is planned.

A great advantage of the proposed setup (as well as IBE in general) is that people can send encrypted emails to an acquaintance, even if this acquaintance has never performed any measures to receive encrypted emails – thus solving the "chicken and egg problem" of other PKI-based systems, where "most users will not perform key management until they have received an encrypted email, and users cannot receive an encrypted email until they perform key management" [28, p .2]. However, finding an encrypted mail in one's inbox that requires one to install an app and an add-on to read it can still be overwhelming. In the future, we plan to explore how this onboarding procedure can be supported more, e.g., by involving the communication partner in this process.

By default, PostGuard only uses the recipient's email address as an identity for the encryption. A disadvantage of this setting is that anyone with access to the recipient's email inbox (including partners or email service providers) can decrypt the recipient's emails.[8] Hence, the default setting does not prevent emails that are sent to the wrong person from being read. To address this, we have implemented the optional usage of *additional attributes*. We plan to explore how and when to encourage their use. In addition, we wish to explore user experience design strategies that can prevent mistakes in the recipient selection process, both when using PostGuard and in general.

For security purists, the fact that a TTP is involved could be considered a no-go. Indeed, the TTP is a single point of failure and needs to be trusted. However, using a TTP comes with great usability benefits due to its automatic key management. In future research, we wish to address the risks associated with using a TTP.

[8]The reason for this limitation is that if someone has access to the recipient's emails, they can also use this access to load the email-attribute needed for the decryption process into their Yivi app. Note that this limitation is shared by IBE-based systems that rely on email-based authentication, such as Pwm 2.0 [25].

For instance, we plan to extend our system with the possibility of distributing trust by having PKGs employed under the control of different organisations. (Adida et al. [3] show how PKGs could be distributed per mail provider domain.) Also, we plan to increase availability, i.e., if one PKG instance becomes unavailable, a secondary PKG will automatically become available.

Our add-ons do not (yet) support encrypting emails for recipients in the BCC nor the decryption of forwarded emails. These features are on the road map. In addition, we wish to give users the ability to decide whether plain text copies of emails should be stored on the server of their email provider and to customize other user settings, such as whether to encrypt emails by default. Furthermore, more add-ons, e.g., for web clients like Gmail, are planned for the medium term.

In the longer term, a planned feature is to support digitally signing emails, as it would strengthen message integrity and guarantee source authenticity. Because encryption and digital signatures share the same technical foundation, we can reuse the infrastructure we have in place. Furthermore, we plan to explore use cases where emails are encrypted for groups with common attributes (e.g., 'doctor at a hospital') rather than individuals.

Our proof of concept demonstrates that by combining IBE with an identity wallet, we can build on users' existing and intuitive mental models of encryption. Our project leverages Yivi but is flexible enough to support other authentication mechanisms and different (future) identity wallets. We believe researching potential use cases of identity wallets is a fruitful direction for future HCI research, especially given the European Commission's recent call for a European digital identity.

ACKNOWLEDGMENTS

We thank Jorrit Geels, Wouter Sluis-Thiescheffer and Thea van der Geest for their feedback on the prototype and contributions to the design process. We also thank our project partners Nedap Healthcare, VGZ, SURFnet, the Municipality of Nijmegen and Compumatica for their contributions. Furthermore we thank the CHI 2023 reviewers for their valuable feedback on the paper. Finally, we thank NWO, SIDN fonds and NLnet for supporting the project financially.

REFERENCES

[1] Ruba Abu-Salma, Elissa M. Redmiles, Blase Ur, and Miranda Wei. 2018. Exploring User Mental Models of End-to-End Encrypted Communication Tools. In *FOCI @ USENIX Security Symposium*. USENIX Association.

[2] Ruba Abu-Salma, M. Angela Sasse, Joseph Bonneau, Anastasia Danilova, Alena Naiakshina, and Matthew Smith. 2017. Obstacles to the Adoption of Secure Communication Tools. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 137–153.

[3] Ben Adida, Susan Hohenberger, and Ronald L. Rivest. 2005. Lightweight Encryption for Email. In *SRUTI*. USENIX Association.

[4] Erinn Atwater, Cecylia Bocovich, Urs Hengartner, Ed Lank, and Ian Goldberg. 2015. Leading Johnny to Water: Designing for Usability and Trust. In *SOUPS*. USENIX Association, 69–88.

[5] Wei Bai, Moses Namara, Yichen Qian, Patrick Gage Kelley, Michelle L. Mazurek, and Doowon Kim. 2016. An Inconvenient Trust: User Attitudes toward Security and Usability Tradeoffs for Key-Directory Encryption Systems. In *SOUPS*. USENIX Association, 113–130.

[6] Dan Boneh and Xavier Boyen. 2004. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *EUROCRYPT (Lecture Notes in Computer Science, Vol. 3027)*. Springer, 223–238.

[7] Dan Boneh and Matthew K. Franklin. 2001. Identity-Based Encryption from the Weil Pairing. In *CRYPTO (Lecture Notes in Computer Science, Vol. 2139)*. Springer, 213–229.

[8] Merel Brandon, Hanna Kathrin Schraffenberger, Wouter Sluis-Thiescheffer, Thea van der Geest, Daniel Ostkamp, and Bart Jacobs. 2022. Design Principles for Actual Security. In *NordiCHI (Adjunct)*. ACM, 41:1–41:6.

[9] Sergej Dechand, Alena Naiakshina, Anastasia Danilova, and Matthew Smith. 2019. In Encryption We Don't Trust: The Effect of End-to-End Encryption to the Masses on User Perception. In *EuroS&P*. IEEE, 401–415.

[10] Whitfield Diffie and Martin E. Hellman. 1976. New directions in cryptography. *IEEE Trans. Inf. Theory* 22, 6 (1976), 644–654.

[11] European Commission. . European Digital Identity. https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_en. Accessed: 6 March 2023.

[12] Ann Fry, Sonia Chiasson, and Anil Somayaji. 2012. Not sealed but delivered: The (un) usability of S/MIME today. In *Annual Symposium on Information Assurance and Secure Knowledge Management (ASIA'12), Albany, NY*.

[13] Simson L. Garfinkel. 2003. Email-Based Identification and Authentication: An Alternative to PKI? *IEEE Secur. Priv.* 1, 6 (2003), 20–26.

[14] Simson L. Garfinkel and Robert C. Miller. 2005. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *SOUPS (ACM International Conference Proceeding Series, Vol. 93)*. ACM, 13–24.

[15] Peter Gutmann. 2004. Why isn't the Internet secure yet, dammit. In *AusCERT Asia Pacific Information Technology Security Conference*.

[16] L. Ingham. 2018. 88cyberattacks. https://www.verdict.co.uk/uk-data-breaches-human-error/

[17] Ada Lerner, Eric Zeng, and Franziska Roesner. 2017. Confidante: Usable Encrypted Email: A Case Study with Lawyers and Journalists. In *EuroS&P*. IEEE, 385–400.

[18] Albert Levi and Can Berk Güder. 2009. Understanding the limitations of S/MIME digital signatures for e-mails: A GUI based approach. *Comput. Secur.* 28, 3-4 (2009), 105–120.

[19] Divakaran Liginlal, Inkook Sim, and Lara Khansa. 2009. How significant is human error as a cause of privacy breaches? An empirical study and a framework for error management. *Comput. Secur.* 28, 3-4 (2009), 215–228.

[20] Cristian Thiago Moecke and Melanie Volkamer. 2013. Usable secure email communications: criteria and evaluation of existing approaches. *Inf. Manag. Comput. Secur.* 21, 1 (2013), 41–52.

[21] Privacy by Design Foundation and SIDN. 2023. *IRMA [Mobile application]*. https://irma.app

[22] Privacy by Design Foundation and SIDN. 2023. *Yivi [Mobile application]*. https://yivi.app

[23] Adrian Reuter, Ahmed Abdelmaksoud, Karima Boudaoud, and Marco Winckler. 2021. Usability of End-to-End Encryption in E-Mail Communication. *Frontiers Big Data* 4 (2021), 568284.

[24] Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent E. Seamons. 2016. "We're on the Same Page": A Usability Study of Secure Email Using Pairs of Novice Users. In *CHI*. ACM, 4298–4308.

[25] Scott Ruoti, Jeff Andersen, Travis Hendershot, Daniel Zappala, and Kent E. Seamons. 2016. Private Webmail 2.0: Simple and Easy-to-Use Secure Email. In *UIST*. ACM, 461–472.

[26] Scott Ruoti, Jeff Andersen, Tyler Monson, Daniel Zappala, and Kent E. Seamons. 2018. A Comparative Usability Study of Key Management in Secure Email. In *SOUPS @ USENIX Security Symposium*. USENIX Association, 375–394.

[27] Scott Ruoti, Jeff Andersen, Daniel Zappala, and Kent Seamons. 2015. Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client. *arXiv preprint arXiv:1510.08555* (2015).

[28] Scott Ruoti, Nathan Kim, Ben Burgon, Timothy W. van der Horst, and Kent E. Seamons. 2013. Confused Johnny: when automatic encryption leads to confusion and mistakes. In *SOUPS*. ACM, 5:1–5:12.

[29] Scott Ruoti and Kent E. Seamons. 2019. Johnny's Journey Toward Usable Secure Email. *IEEE Secur. Priv.* 17, 6 (2019), 72–76.

[30] Adi Shamir. 1984. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO (Lecture Notes in Computer Science, Vol. 196)*. Springer, 47–53.

[31] Steve Sheng, Levi Broderick, Colleen Alison Koranda, and Jeremy J Hyland. 2006. Why Johnny still can't encrypt: evaluating the usability of email encryption software. In *SOUPS*. ACM, 3–4.

[32] Christian Stransky, Oliver Wiese, Volker Roth, Yasemin Acar, and Sascha Fahl. 2022. 27 Years and 81 Million Opportunities Later: Investigating the Use of Email Encryption for an Entire University. In *IEEE Symposium on Security and Privacy*. IEEE, 860–875.

[33] Alexander Ulrich, Ralph Holz, Peter Hauck, and Georg Carle. 2011. Investigating the OpenPGP Web of Trust. In *ESORICS (Lecture Notes in Computer Science, Vol. 6879)*. Springer, 489–507.

[34] Alma Whitten and J. Doug Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *USENIX Security Symposium*. USENIX Association.

[35] Justin Wu and Daniel Zappala. 2018. When is a Tree Really a Truck? Exploring Mental Models of Encryption. In *SOUPS @ USENIX Security Symposium*. USENIX Association, 395–409.