Chapter 8
Privacy-Enhancing Technologies

Kent Seamons

Abstract An increasing amount of sensitive information is being communicated and stored online. Frequent reports of data breaches and sensitive data disclosures underscore the need for effective technologies that users and administrators can deploy to protect sensitive data. Privacy-enhancing technologies can control access to sensitive information to prevent or limit privacy violations. This chapter focuses on some of the technologies that prevent unauthorized access to sensitive information. These technologies include secure messaging, secure email, HTTPS, two-factor authentication, and anonymous communication. Usability is an essential component of a security evaluation because human error or unwarranted inconvenience can render the strongest security guarantees meaningless. Quantitative and qualitative studies from the usable security research community evaluate privacy-enhancing technologies from a socio-technical viewpoint and provide insights for future efforts to design and develop practical techniques to safeguard privacy. This chapter discusses the primary privacy-enhancing technologies that the usable security research community has analyzed and identifies issues, recommendations, and future research directions.

8.1 Introduction

An increasing amount of sensitive information is being communicated, stored, and shared online. Unauthorized access to this information can lead to serious privacy violations. This chapter focuses on some privacy-enhancing technologies (PETS) that prevent unauthorized access to sensitive information.

The need for effective privacy-enhancing technologies has never been greater. The Snowden revelations in 2013 exposed a broad array of government surveillance programs and ignited renewed interest in privacy-preserving technologies that

K. Seamons (✉)
Brigham Young University, Provo, UT, USA
e-mail: seamons@cs.byu.edu

B. P. Knijnenburg et al. (eds.), *Modern Socio-Technical Perspectives on Privacy*,
https://doi.org/10.1007/978-3-030-82786-1_8

prevent eavesdropping. The steady stream of data breaches underscores the risks of trusting third parties with sensitive information and the importance of robust defenses against unauthorized account access. Along with the risk of sensitive information disclosure, the rise of online social networks and other data-sharing cloud services presents increasing opportunities for privacy violations.

This chapter discusses several technologies that the usable security research community has analyzed and identifies issues, recommendations, and future research directions. For example, privacy-enhancing technologies protect sensitive information during transmission (HTTPS) and storage at untrusted third parties (end-to-end encryption in secure messaging and secure email). Technologies can also keep sensitive conversations private so that the fact that two parties are even communicating is not made public (Tor).

Usable security researchers explore the human-computer interaction aspects of PETS, an important socio-technical element beyond the formal security guarantees established by a security analysis. A security technology that is not usable can lead to reduced security or no security at all.

8.2 Secure Messaging

Instant messaging (IM) applications provide an online chat capability that allows users to communicate in real time. Without end-to-end encryption, the conversation is not private. Early IM applications did not provide any encryption, allowing eavesdropping by the service provider or anyone with access to the network during transmission. For instance, other users on a wireless network with the proper software could easily view the chat messages.

IM applications typically rely on a centralized server to relay messages between the users. If each user connects to the server using HTTPS, no network eavesdroppers can read the messages. However, the service provider has access to the entire chat conversation when the server relays or stores the plaintext messages.

A significant post-Snowden development was the creation of the Signal Protocol [1] by Moxie Marlinspike and Trevor Perrin. The protocol provides end-to-end encryption and is available in WhatsApp, the Signal app by Open Whisper Systems, and Facebook Messenger Secret Conversations. Together, these applications serve over a billion users. Moxie and Trevor received the 2017 Levchin Prize for Real-World Cryptography for the development and widespread deployment of the Signal Protocol.

The widespread adoption and use of the Signal Protocol may represent the largest, most rapid adoption of end-to-end encryption in history. Interestingly, privacy was not a driving motivation for users to adopt these systems [2]. Instead, adoption was based on the natural spread of messaging applications so that friends could communicate with their friends. The privacy benefits of encrypted conversations are a side benefit that did not drive adoption. Nonetheless, the result is that billions of users have private conversations protected against passive eavesdropping

by the service provider, hackers, and governments without taking any action or even being aware of that protection.

With Signal, each user has a public and private key that the messaging app generates upon installation. Signal implementations rely on a server to relay messages and distribute public keys. When Alice and Bob want to communicate securely, they obtain each other's public key (sometimes referred to as their identity key) and jointly compute a shared key using their identity keys. As Alice and Bob exchange messages, they continue to calculate new shared keys to encrypt each message with a different encryption key. This approach provides *forward secrecy*, a significant privacy protection property in many state-of-the-art encryption systems. Even if the encryption key for a message is compromised, the attacker gains access to only a single message and cannot read any previous or future messages using the compromised key.

All deployments of the Signal Protocol rely on a centralized key server trust model. The messaging provider maintains a key server that stores and hands out public keys for all users. Reliance on a trusted key server means that even though Signal protects against passive eavesdropping, users are vulnerable to an active man-in-the-middle (MITM) attack.

An active MITM attack works as follows. Suppose Alice and Bob want to have a private conversation. A compromised key server can hand out fake keys for Alice and Bob. As Alice and Bob send messages to each other, the compromised server can decrypt and possibly modify each message as it flows through the provider. Alice and Bob are not even aware that their messages are not private.

To defend against an active MITM attack, users can complete an authentication ceremony with each contact to verify out-of-band that their device has the correct public key of their contact and not a fake key from an active MITM attacker. Key verification requires that a pair of users each navigate to an interface in their messaging client and confirm that both users have the same fingerprint (sometimes called a safety number) on their phone as their partner (see Fig. 8.1). There are two ways to accomplish key verification. First, if the users are physically co-located, the app provides each with a QR code that they each scan from their partner's phone to confirm that they have each other's correct public key. The QR code confirmation requires that the partners meet and conduct the ceremony in person. If they are remote, they can confirm that they each have the same. One partner can read the safety number to the other partner to confirm. If the numbers match, the conversation is private. If the numbers don't match, an active MITM attack is likely in progress.

The usable security research community has studied the effectiveness of the current authentication ceremony and proposed improvements based on usability study results. There have been studies that compare various methods for comparing fingerprints [3–5], along with studies that analyze the full authentication ceremony.

Early studies of Signal revealed that the ceremony was unusable, error-prone, and took too long [6, 7]. More recently, a redesign of the authentication ceremony interface using opinionated design resulted in fewer mistakes and reduced the average time to find and complete the ceremony from 11 to 2 min [8].

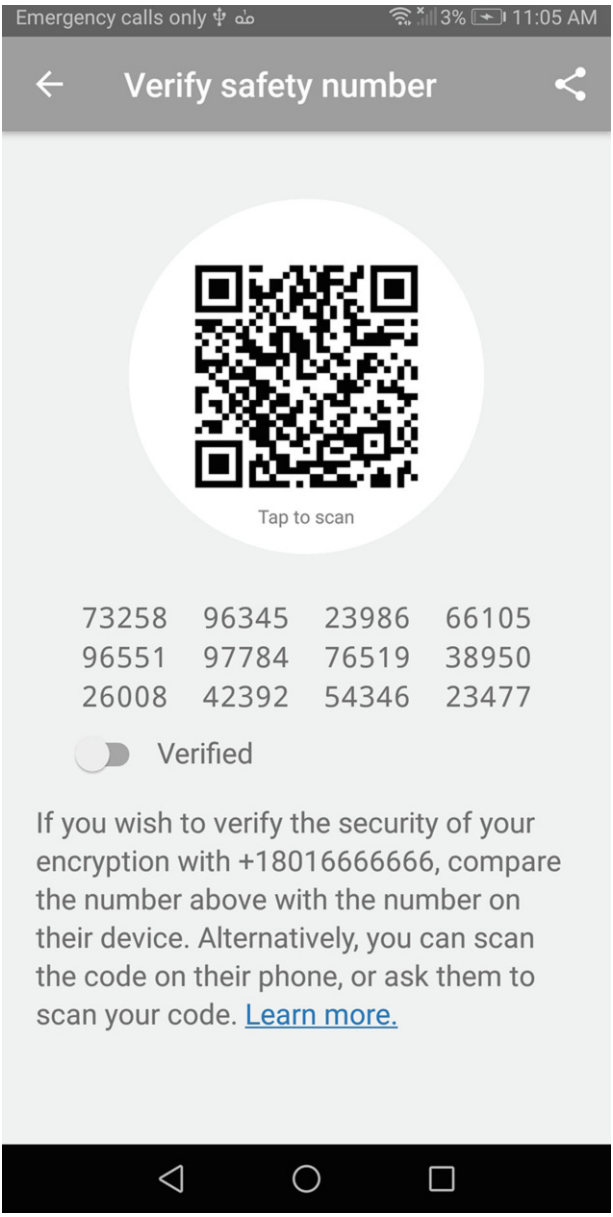


Fig. 8.1 Signal app authentication ceremony

Next, an approach to automate the ceremony by confirming key ownership using social media accounts removed the requirement that both participants be physically present or communicate online while confirming each other's keys. However, the

results show that users do not trust social network providers as the third party to distribute their keys automatically [9].

Finally, Wu et al. [10] applied risk communication theory to test modifications to the Signal app authentication ceremony that showed increased user understanding and decreased usage of the authentication ceremony as users made choices based on risk assessments.

In summary, the authentication ceremony in secure messaging applications is broken [11]. Research reveals the problems and limitations of the current approach. It is unlikely that very many of the billions of secure messaging users will start using the ceremony. Hence, a better approach to obtaining assurance for the masses is to automatically detect attacks to distribute fake keys for users.

Issues

- **Lack of user understanding**—Users do not understand the need for the authentication ceremony to detect man-in-the-middle attacks.
- **Authentication ceremony is not usable**—Lab studies show that the authentication ceremony is time-consuming, error-prone, and hard to use.
- **No interoperability**—The current IM systems are walled gardens or silos. There is no interoperability between different providers. A WhatsApp user cannot communicate securely with a Facebook Messenger user.
- **Server must be trusted**—The secure messaging provider is trusted to deliver public keys and provide the client software. Even if a user has the assurance that a contact's correct key is in use, there is still the risk that compromised software can leak keys or sensitive information. The government could still coerce a company to update the software on a target's phone with a backdoor.

Recommendations

- **Use secure messengers for private communication**—For secure communication, the best option available today is for individuals to use an instant messenger client that supports the Signal Protocol for private conversations.
- **Vulnerable users should complete the authentication ceremony**—If you are a high-risk target or are communicating highly sensitive data, complete the authentication ceremony with each contact to ensure there is no MITM attack.

Research Directions

- **Nudge users to verify keys only when the risk is high**—Explore the use of machine learning to detect when a user is communicating sensitive information and then prompt the user to complete the authentication ceremony when the risk of data compromise is greatest. This approach enables users to make risk decisions at the moment the risk is present.
- **Automate key verification**—Design methods to automatically detect or prevent active MITM attacks that relieve users from the burden of the authentication ceremony.

- **Provide application-independent key management**—Provide centralized support for key management in the browser or operating system to enable interoperability between applications that support the Signal Protocol.
- **Better support for message deletion**—Study user behavior and attitudes in response to new features that allow them to forget or delete messages.
- **Safeguard encryption keys**—Design approaches utilizing secure enclaves to protect encryption keys and encryption software from compromise.

8.3 Secure Email

Email was not originally designed to be secure. Figure 8.2 illustrates the vulnerabilities in plaintext email, including (1) unsecured links, (2) message forgeries, (3) malicious content, and (4) untrusted servers. Without the use of encryption, sensitive email messages are vulnerable to eavesdroppers while in transit. In addition, sensitive messages stored in the server are also vulnerable to unauthorized access. Email messages are often stored indefinitely, and the long-term storage adds to the privacy risks even if the data is compromised well into the future.

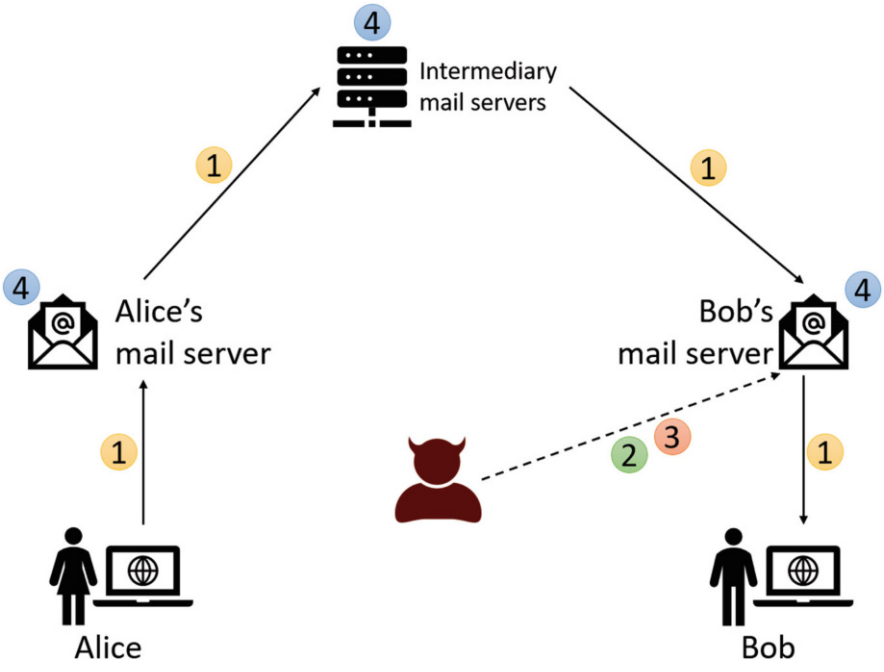


Fig. 8.2 Email vulnerabilities

Technologies exist to address some of these vulnerabilities [12]. Transport Layer Security (TLS) encrypts email messages during transmission between communication links. Sender Policy Framework (SPF) lets the domain owner specify the legitimate servers that send email messages for that domain. DomainKeys Identified Mail (DKIM) includes a signature on each email message from a domain to guard against message forgery. Recent studies show that these techniques are not universally deployed, leaving a significant gap in the secure email infrastructure [13–16]. Even if we close this gap, servers still have access to plaintext email messages, and the threat of disclosure to hackers or government surveillance remains.

End-to-end encryption addresses the issue of server access to plaintext email. The most well-known secure email systems are S/MIME and PGP. S/MIME supports a hierarchical, top-down trust model and is used mainly in corporations. PGP supports a grass-roots, bottom-up trust model that is suitable for individuals. Deployment of these technologies continues to languish after decades. The reasons are complex and nuanced, and center around a diverse set of stakeholders with competing interests that no one-size-fits-all solution can satisfy [12].

Secure email is one of the challenges that launched the field of usable security with the seminal paper *Why Johnny Can't Encrypt* [17]. The paper included a lab usability study of PGP that failed miserably and provided a wake-up call to the security community of the importance of user-centered design. Nearly 20 years later, a lab usability study of a modern PGP web client (Mailvelope) had similarly disappointing results when 9 of 10 participant pairs were unable to exchange a secure email message after 1 h of trying to use the system [18].

Despite the lack of a usable production PGP tool, recent research has produced several highly usable secure email interfaces (e.g., [19–21]). Figure 8.2 illustrates the interface for Private Webmail 2.0, a system that grew out of a series of studies (e.g., [22, 23]) over many years. The research shows that the following are essential properties for a usable, secure email interface:

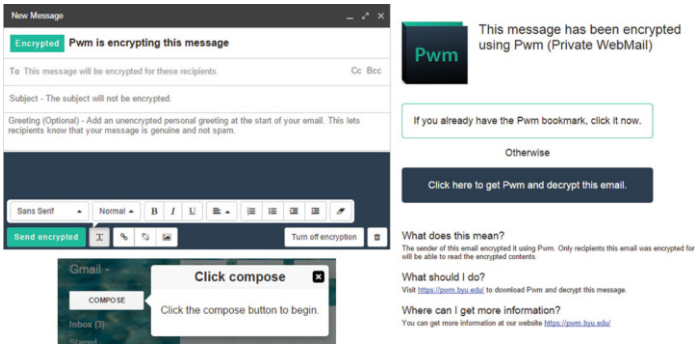
1. *Tight Integration.* Users want secure email systems that enhance their existing email clients and fit within their existing workflows [20]. This integration is both visual and functional—that is, it looks like a part of the client application and has similar functionality, respectively (see Fig. 8.3). While visual integration is important, users should be able to clearly distinguish between emails protected with end-to-end encryption and emails that are not [22].

   While most users prefer integrated solutions, a small but consistent portion prefer standalone clients [20, 21], believing that handling secure email in a separate client makes it more obvious to the user when encryption is in use.
2. *Inline, context-sensitive tutorials.* Tutorials are essential in helping users understand how to use secure email properly [22]. For users to pay attention and use these tutorials, the tutorial must be shown inline with the secure email system [22]. Additionally, the system should provide context-sensitive tutorials, walking first-time users through the process of sending and receiving secure email (see Fig. 8.3, left side).

3. *Streamlined onboarding.* Encrypted email should be designed to help recipients understand what they have received and what actions they need to take next [22] (see Fig. 8.3, right side). If the secure email system requires recipients first to generate a key pair, the system should automatically send an email explaining what the recipient needs to do [20]. Additionally, the system should save a draft of the sender's message to send automatically after the recipient generates and makes their public key available.
4. *Understandable and trustworthy design.* Interfaces need to help users understand how secure email is protecting them—for example, telling them whether the subject line is encrypted (it usually is not). Increased understanding allows users to make informed decisions and avoid mistakes [22, 24]. Additionally, system operation needs to conform to user expectations; otherwise, users reject the system. For example, studies show that if encryption happens too quickly, users assume that their messages were not encrypted and did not trust the tool [22].
5. *Easy-to-use key management.* Users struggle with managing their keys. Automation of key generation, uploading, and discovery significantly improve the user experience [19, 20, 24].

Studies show that systems applying (most of) these principles are perceived as highly usable, result in a low mistake rate, and help novice users begin sending encrypted email without expert assistance [20–24].

Secure email is a two-body problem, and Ruoti et al. [23] pioneered a novel two-person methodology where pairs of participants are brought into a user study to test a system. They found that participants were more relaxed during the study and



Top left—placeholder text that acts as an inline tutorial instructing users about how secure email works.
Bottom left—an inline, context-sensitive tutorial helping users send an encrypted email for the first time.
Right—the body of the encrypted email providing plaintext instructions to streamline onboarding.

Fig. 8.3 Interface for private webmail (Pwm) 2.0 [22], a modern usable secure email system

did not automatically assume that they were responsible for system mistakes. This methodology has been utilized in recent secure messaging studies [8, 10].

The storage of plaintext email at the server presents an attack surface in case of future account compromise. Research is beginning to explore email deletion capabilities that reduce the risk of future disclosure of sensitive messages that do not require long-term storage [25]. Another approach is to encrypt the plaintext email on the server with a locally stored encryption key and delete the plaintext copy [26]. The advantage of this approach is that a user can do this unilaterally without the cooperation of their communication partner. However, it does not safeguard the copy of the message in the original sender's outbox.

Multiple stakeholders with competing priorities for secure email make it difficult for a one-size-fits-all solution [27]. There needs to be more willingness by the various stakeholders to allow for alternatives that support the needs of only some of the stakeholders. There has been significant effort by the usable security research community to study the issues surrounding secure email, and usable system designs have been demonstrated in a laboratory setting [28].

Issues

- **PGP is dead**—PGP is a failed experiment. Even the proponents of secure email have been abandoning PGP recently. Modern PGP clients exhibit poor usability in laboratory user studies.
- **Secure email solutions are not interoperable**—The various approaches to secure email are not interoperable (PGP, S/MIME, proprietary web-based systems).
- **It is difficult to introduce a secure email solution that maintains ubiquity**—Email has a history emphasizing ubiquity and interoperability. Anyone can send an email to any other user if they have an email address for them. It is challenging to introduce secure email into this environment and maintain the same service guarantees.

Recommendations

- **Individuals should use secure messaging instead**—Given the current state of secure email, use a secure messaging client that supports the Signal Protocol for private conversations with friends and family.
- **Businesses should use S/MIME or secure webmail**—For sensitive business communications that must occur over email, enterprises can use S/MIME, and small businesses and individuals can use web-based secure email services such as ProtonMail and Tutanota.
- **All email providers should support TLS**—It is exasperating that all email is not protected from passive eavesdropping today. All email systems should support TLS for exchanging email between email systems.
- **Delete old sensitive email when possible**—Unless legally required to retain a copy, delete outdated sensitive email messages that could be problematic if made public following an account compromise.

Research Directions

- **Longitudinal studies are needed**—Previous secure email usability studies are all short-term lab studies. Longitudinal studies could confirm whether the above design principles are sufficient for email or whether more improvements are needed to support long-term usage.
- **Providers should only have access to encrypted messages**—Develop techniques to process encrypted data so the email provider can scan for malware and spam without having access to the plaintext.
- **Support easy deletion of old messages**—Analyze the usability of secure email deletion approaches.

8.4 HTTPS

The privacy of sensitive information is protected when it is encrypted during transmission over an insecure network. HTTPS is the protocol for encrypting data transmitted between browsers and web servers. It relies on a lower-level protocol known as Transport Layer Security (TLS), formerly known as the Secure Socket Layer (SSL). At a high level, HTTPS/TLS/SSL are synonymous—it is the most common protocol for encrypted communication on the Internet. As part of the HTTPS setup, a website authenticates to the browser using a certificate digitally signed by a trusted third party.

In 2010, the Firesheep browser extension demonstrated the risks of session hijacking for sites that used HTTPS only to protect the login page and sent browser cookies in the clear. The tool allowed an attacker sniffing traffic on a wireless network to easily hijack another user's session to gain unauthorized access to their social network or webmail. The publicity and ease of performing this attack contributed to major websites, like Google and Facebook, requiring HTTPS for all session traffic to their website.

More recently, there has been a significant uptick in the use of HTTPS for all web traffic, which provides increased protection against unauthorized access to browser activity. The increase in HTTPS has been mainly fueled by Let's Encrypt, a service that offers free digital certificates that are easy for admins to request and manage.

The most recent version of the protocol (TLS 1.3) supports only ciphers that provide the forward secrecy property described earlier in the secure messaging discussion. It also provides increased privacy protection by encrypting the server certificates transmitted to the client.

There are three aspects of HTTPS that have socio-technical implications: (1) HTTPS warning messages, (2) developer and administrator development and deployment hurdles, and (3) HTTPS inspection.

8.4.1 HTTPS Warning Messages

Users are sometimes faced with HTTPS warning messages when the browser encounters a website certificate that does not properly validate. The web server certificate authenticates a website to the browser. HTTPS warnings have been an area of significant usability studies and modification over the past decade.

Researchers at Google have been at the forefront of this effort[29–35]. Google is well-positioned to gather telemetry data from its users and conduct large-scale A/B tests comparing alternate designs. The results of these studies have shown a decrease in the click-through rate of HTTPS warning messages. It isn't clear that the reduction is due to increased user understanding as much as it is that the browsers make it more difficult to click through the warning message by adding additional steps that may discourage users from continuing.

Browsers display a lock icon whenever HTTPS is in use. Users interpret the lock icon as an indicator that the website is secure. Several studies show that users interpret the lock icon to mean that the website is secure [36]. The lock icon is unrelated to the website's security, and this misunderstanding is precisely the wrong interpretation when a phishing website impersonates a well-known company and employs HTTPS.

8.4.2 HTTPS Development and Deployment

Software developers that build HTTPS applications can make mistakes, such as introducing certificate validation errors that compromise user privacy [37–39].

Two recent research projects presented user-level and system-level approaches for automatically intercepting and verifying a website certificate, which overrides an application with broken HTTPS authentication [40, 41]. An approach that goes beyond intercepting broken HTTPS applications is to design a more straightforward, abstract interface to HTTPS applications that make it easier for developers to build reliable applications that encrypt network traffic. O'Neill et al. [42] proposed a simple extension to the existing POSIX secure socket API. They demonstrate how their approach can replace a 300+ line program using the complex OpenSSL library with a 16-line program using their simplified approach for creating a secure connection to a website.

Administrators of websites can misconfigure HTTPS and create privacy risks for users. A qualitative study examined mental models of users and administrators, and administrators did not understand protocol components and technical terms associated with HTTPS [36].

8.4.3 HTTPS Interception

HTTPS interception (also known as a TLS Proxy) often occurs within a corporation, library, or school so that the organization can monitor the security of their network and make sure clients are not unknowingly downloading malware or releasing sensitive business secrets from a compromised machine. Several studies measure how often HTTPS inspection occurs by using detection techniques from the server side [43–45]. A study of client-side HTTPS interception software shows that these systems can introduce privacy risks to users due to programmer error [46].

The same mechanisms utilized for inspection can also be the source of an active man-in-the-middle attack. Since the two options appear similar, it makes the task of informing the user all the more difficult.

A survey of user attitudes and opinions of HTTPS inspection showed that most users are unfamiliar with this practice and consider it an invasion of privacy [47, 48]. Some results of the survey are illustrated in Figs. 8.4 and 8.5. Although security proponents often object vehemently to the practice of HTTPS inspection, users were more moderate and accepting that a business or organization has the right to inspect the traffic on their network. Surveys show that users overwhelmingly want to be notified when HTTPS inspection is happening.

Issues

- **Companies perform HTTPS Interception to protect their network and systems.** Intermediate servers run by a company or organization can access a user's HTTPS web traffic, known as HTTPS inspection. Some users consider this a privacy breach, but they understand why a company must protect its network.
- **Users overwhelmingly desire notification** when an HTTPS proxy is operating.
- **Many HTTPS applications are broken.** Developers make mistakes when building HTTPS applications due to the complexity of HTTPS libraries.

Recommendations

- **Use HTTPS on all websites.** All owners of websites should be using HTTPS for all connections to their site. Certificates are inexpensive and easy to acquire using Let's Encrypt.
- **Website designers should not mix HTTP and HTTPS traffic** on a single web page since this has been a source of compromise in the past.

Research Directions

- **HTTPS warning redesign**—As we transition to a web where most sites support HTTPS, new approaches are needed for warning users when there is a risk of using an HTTP website and when they should not trust an HTTPS site.
- **HTTPS inspection detection**—We need improved techniques to detect when HTTPS inspection occurs and the associated risks.
- **Developer and administrator usability studies** can determine the effectiveness of recent advances in easy-to-use libraries for building and deploying HTTPS applications.

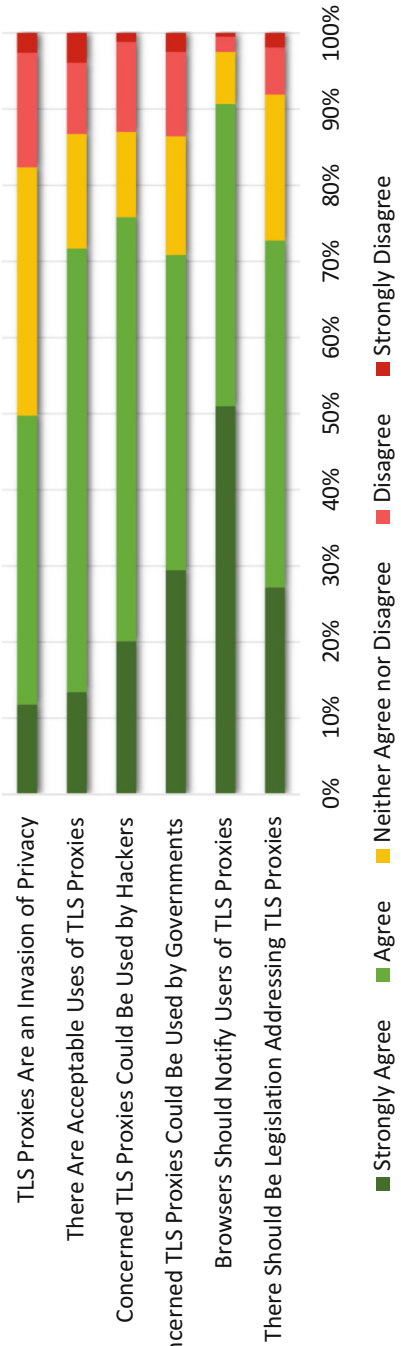


Fig. 8.4 Participant attitudes toward HTTPS inspection (N=1049)

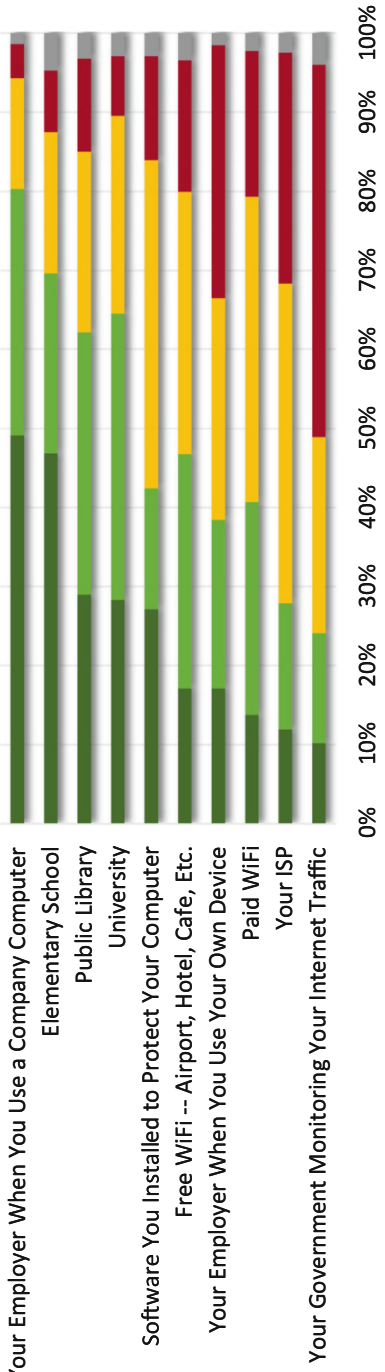


Fig. 8.5 Participant responses on scenarios—should the organization be allowed to inspect HTTPS traffic? (N=927)

8.5 Two-Factor Authentication

Controlling access to sensitive user data is paramount to privacy. Passwords remain the most common form of online user authentication today, despite the tremendous amount of research that demonstrates their security and usability weaknesses [49]. The number of data breaches is increasing rapidly [50], and most data breaches involve weak, compromised, or default passwords [51]. In response, many organizations and individuals turn to two-factor authentication (2FA) to protect against privacy violations that occur when user accounts are compromised.

2FA requires users to present two factors from *something they know* (such as a password or the answers to a set of security questions), *something they have* (such as a phone or hardware token), and *something they are* (a biometric such as a fingerprint or facial recognition). 2FA protects against remote attackers because attackers are not able to compromise user accounts using passwords alone.

Companies deploy 2FA internally to strengthen security. Lang et al. [52] report on Google's internal deployment of security keys to their employees. It was generally successful and reduced the number of authentication-related support tickets.

Later, two academic studies of security keys showed that the setup was challenging for some end-users who completed tasks without the aid of an IT support staff. Das et al. [53] performed two studies measuring both the usability and the acceptability of using the YubiKey (a type of FIDO U2F compliant hardware token) as a second factor in securing a Google account. Employing a think-aloud protocol, they made some recommendations to Yubico (the manufacturer of the YubiKey) based on common points of confusion. After 1 year, they repeated the study with a new group of users, finding that although many of the previous usability concerns had been addressed, many users still did not see much benefit in using the YubiKey. Das et al. postulated that this lack of acceptability was due partly to the lack of awareness of the risks mitigated through using the YubiKey. Reynolds et al. [54] describe two usability studies of YubiKeys. The study found many usability concerns with the setup process of the YubiKey but found that day-to-day usability was significantly higher.

Reese et al. [55] compared five types of 2FA over 2 weeks and found that users generally had a positive experience. The setup experience for users in this study was favorable compared to earlier setup studies, indicating that improved setup instructions make a difference.

Many universities in the United States are adopting Duo 2FA to reduce the risk of compromised student data. Two qualitative studies completed at CMU [56] and BYU [57] explored user attitudes toward Duo after being required to use it for some time. Both institutions report that some users are annoyed at being required to use 2FA on all university websites, even some that may not have personal information at risk. There is also evidence that users are frustrated with certain limitations that have potential solutions that they are unaware exist. For instance, some users are frustrated when they need access to the web when they have no Wi-Fi available, and they do not know that there are alternative 2FA options that work without any Wi-Fi, such as a phone app that generates one-time passwords.

Issues

- **Lack of user understanding**—Users do not understand how 2FA makes them more secure. Users also do not understand the pros and cons of various 2FA options.
- **No support for account sharing**—Studies show that account sharing occurs, and two-factor solutions are not easy to adapt to legitimate account-sharing scenarios such as for family members or a caregiver.
- **Weak backup authentication**—Many websites require a fallback authentication method in case a 2FA device is lost, effectively reducing security to the strength of the authentication method used for backup.

Recommendations

- A **standardized setup process** for 2FA devices across the major websites would make it easier for users to setup 2FA on multiple accounts easily.

Research Directions

- **Longitudinal studies with diverse populations**—We need longitudinal studies of 2FA technology with a broader population of users, not just university student populations. Some potential populations include the elderly, low-income, and disabled.
- **Integrate 2FA with password managers**—Password managers may be well-suited to help with easy 2FA registration and recovery.
- **Automate the 2FA setup process** to make mass 2FA enrollment simpler and to make it easy to transition to a new authenticator device when the old one is lost or stolen.

8.6 Anonymity

Anonymous communication aims to hide the identity of participants as they communicate to protect their privacy. The Onion Router (Tor) is a free software program supporting anonymous web browsing. It relies on multiple layers of encryption to hide the origin of a request from the website. As messages flow through multiple hops in the Tor network, each node in the random set of relay servers only knows the relay server immediately before and after it along the route through the network.

Qualitative studies consisting of semi-structured interviews with Tor users reveal that users have inaccurate mental models of Tor and Tor Onion services, which raises the risk that user mistakes could deanonymize them [58, 59]. Earlier studies examined the usability of the Tor Browser [60], the Tor Browser Bundle[61, 62], and the Tor Launcher [63]—about 78% of users failed to set up Tor correctly in a usability study of the Tor Launcher. Even though prior research has identified weaknesses and limitations in Tor implementations, proposed design recommendations have yet to be implemented and tested.

Bitcoin supports pseudonymity, meaning a user has a persistent alias unrelated to their real-world identity. To maintain pseudonymity, users can install and run Bitcoin client software on their own, but they run the risk of losing access to their Bitcoin if they forget their password or lose access to their wallet. Usability studies of Bitcoin also show that poor understanding can lead to mistakes that reveal a user's identity or a loss of their cryptocurrency [64–66].

Issues

- **Incorrect mental models**—Non-expert users have incorrect mental models of anonymity technology that increase the risk of mistakes that could deanonymize them.
- **Privacy technology can cause harm**—There are tensions between privacy capabilities that protect individuals from harm and the use of privacy tools to commit crimes and harm individuals.

Recommendations

- **Use tools to limit information collection**—Tools like Tor, private browsing modes, and VPNs can limit the information that is collected about your on-line activities.

Research Directions

- **Easier setup**—How can users easily install or configure Tor to protect their anonymity.
- **User education**—How can we educate or inform users about online privacy risks and ways to effectively mitigate those risks?
- **Tighter integration of privacy technology**—Tighter integration between anonymity technology and browsers or operating systems may increase usability and reduce the risk of errors.
- **Increase the benefits while reducing the potential for harm**—How can we provide users with tools that adequately protect their privacy while reducing the risks that criminals will use those tools to harm vulnerable users [67].

8.7 Summary

This chapter provides an overview of some of the privacy-enhancing technologies that protect access to sensitive data and the socio-technical challenges surrounding them. It discusses results from the usable security community that evaluates this technology and provides insight into future directions to make improvements. The following are some overarching themes and takeaways.

- **Key management presents a usability challenge** for PETs technology and is an essential focus of usable security research moving forward. Key management usability challenges exist for key recovery and key portability in applications that

require encryption. Secure enclaves present new opportunities to protect keys and enable privacy-preserving applications.

- Usability research for PETs extends beyond non-technical end-users to include **developers and administrators**.
- Due to the variety of users and differing goals, **privacy solutions need to be adaptable** to the context, preferences, and goals of individual users.
- Lab user studies help identify when deficiencies are present that need to be corrected. However, once lab results are positive, **longitudinal studies with a general population** are necessary to determine whether an approach is suitable for the day-to-day use of typical users.
- The insights from usability studies in academia and research demonstrate the potential early gains that technology companies could realize by doing **more usability testing for new products**.
- Usability studies show that **users have misunderstandings** about most of the privacy-related software they encounter (e.g., end-to-end encryption, HTTPS inspection, two-factor authentication, anonymous services). Can automated solutions hide technical details and provide security benefits without making the user aware?
- The Signal Protocol is an example of a system that was able to introduce security that prevents eavesdropping without the user having to do anything to configure that protection. However, in most cases, there are options or potential false positives that require the user to decide how best to proceed. A hard question is how to **empower users to easily and conveniently make informed choices** when they face security and usability trade-offs.

Acknowledgments I want to thank Daniel Zappala and our students at BYU that participated in some of the research described in this chapter. This work was supported in part by the National Science Foundation under grants CNS-1528022 and CNS-1816929.

References

1. Systems, O. W. Signal. https://signal.org/. Accessed Feb 10 2018.
2. Abu-Salma, R., M. A. Sasse, J. Bonneau, A. Danilova, A. Naiakshina, and M. Smith. 2017. Obstacles to the adoption of secure communication tools. In *IEEE Symposium on Security and Privacy*. Piscataway: IEEE.
3. Dechand, S., D. Schürmann, K. Busse, Y. Acar, S. Fahl, and M. Smith. 2016. An empirical study of textual key-fingerprint representations. In *USENIX Security Symposium*. Berkeley: USENIX Association.
4. Tan, J., L. Bauer, J. Bonneau, L. F. Cranor, J. Thomas, and B. Ur. 2017. Can unicorns help users compare crypto key fingerprints? In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. New York: ACM.
5. Shirvanian, M., N. Saxena, and J. J. George. 2017. On the pitfalls of end-to-end encrypted communications: A study of remote key-fingerprint verification. In *Annual Computer Security Applications Conference (ACSAC)*. New York: ACM.

6. Herzberg, A., and H. Leibowitz. 2016. Can Johnny finally encrypt? Evaluating E2E-encryption in popular IM applications. In *Workshop on Socio-Technical Aspects in Security and Trust (STAST), Los Angeles*.
7. Vaziripour, E., J. Wu, M. O'Neill, R. Clinton, J. Whitehead, S. Heidbrink, K. Seamons, and D. Zappala. 2017. Is that you, Alice? A usability study of the authentication ceremony of secure messaging applications. In *Symposium on Usable Privacy and Security (SOUPS)*.
8. Vaziripour, E., J. Wu, M. O'Neill, D. Metro, J. Cockrell, T. Moffett, J. Whitehead, N. Bonner, K. Seamons, and D. Zappala. 2018. Action needed! Helping users find and complete the authentication ceremony in Signal. In *Proceedings of the Fourteenth Symposium on Usable Privacy and Security.*
9. Vaziripour, E., D. Howard, J. Tyler, M. O'Neill, J. Wu, K. Seamons, and D. Zappala. 2019. I don't even have to bother them! Using social media to automate the authentication ceremony in secure messaging. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. New York: ACM.
10. Wu, J., C. Gattrell, D. Howard, J. Tyler, E. Vaziripour, K. Seamons, and D. Zappala. 2019. "Something isn't secure, but i'm not sure how that translates into a problem": Promoting autonomy by designing for understanding in signal. In *Symposium on Usable Privacy and Security (SOUPS)*.
11. Herzberg, A., H. Leibowitz, K. Seamons, E. Vaziripour, J. Wu, and D. Zappala. 2020. Secure messaging authentication ceremonies are broken. *IEEE Security & Privacy* 19 (2): 29–37.
12. Clark, J., P. C. van Oorschot, S. Ruoti, K. E. Seamons, and D. Zappala. 2018. Securing email. CoRR abs/1804.07706.
13. Durumeric, Z., D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman. 2015. Neither snow nor rain nor MITM...: An empirical analysis of email delivery security. In *Proceedings of the Internet Measurement Conference (IMC)*. New York: ACM.
14. Foster, I. D., J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko. 2015. Security by any other name: On the effectiveness of provider based email security. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
15. Holz, R., J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar. 2016. TLS in the wild: An Internet-wide analysis of TLS-based protocols for electronic communication. In *Network and Distributed System Security Symposium (NDSS)*.
16. Mayer, W., A. Zauner, M. Schmiedecker, and M. Huber. 2016. No need for black chambers: Testing TLS in the e-mail ecosystem at large. In *IEEE 11th International Conference on Availability, Reliability and Security (ARES)*.
17. Whitten, A., and J. D. Tygar. 1999. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *USENIX Security Symposium*.
18. Ruoti, S., J. Andersen, L. Dickinson, S. Heidbrink, T. Monson, M. O'Neill, K. Reese, B. Spendlove, E. Vaziripour, J. Wu, D. Zappala, and K. Seamons. 2019. A usability study of four secure email tools using paired participants. *ACM Transactions on Privacy and Security (TOPS)* 22 (2): 1–22.
19. Garfinkel, S. L., and R. C. Miller. 2005. Johnny 2: A user test of key continuity management with S/MIME and outlook express. In *Symposium on Usable Privacy and Security (SOUPS)*. New York: ACM.
20. Atwater, E., C. Bocovich, U. Hengartner, E. Lank, and I. Goldberg. 2015. Leading Johnny to water: Designing for usability and trust. In *Symposium on Usable Privacy and Security (SOUPS)*. New York: ACM.
21. Lerner, A., E. Zeng, and F. Roesner. 2017. Confidante: Usable encrypted email: A case study with lawyers and journalists. In *Proceedings of the IEEE European Symposium on Security and Privacy*. Piscatawy: IEEE.
22. Ruoti, S., J. Andersen, T. Hendershot, D. Zappala, and K. Seamons. 2016. Private webmail 2.0: Simple and easy-to-use secure email. In *Proceedings of the 2016 Symposium on User Interface Software and Technology*. New York: ACM.

23. Ruoti, S., J. Andersen, S. Heidbrink, M. O'Neill, E. Vaziripour, J. Wu, D. Zappala, and K. Seamons. 2016. We're on the same page: A usability study of secure email using pairs of novice users. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. New York: ACM.
24. Bai, W., M. Namara, Y. Qian, P. G. Kelley, M. L. Mazurek, and D. Kim. 2016. An inconvenient trust: User attitudes toward security and usability tradeoffs for key-directory encryption systems. In *Symposium on Usable Privacy and Security (SOUPS)*. Berkeley: USENIX.
25. Monson, T., S. Ruoti, J. Reynolds, D. Zappala, T. Smith, and K. Seamons. 2018. A usability study of secure email deletion. In *European Workshop on Usable Security (EuroUSEC)*.
26. Koh, J. S., S. M. Bellovin, and J. Nieh. 2019. Why Joanie can encrypt: Easy email encryption with easy key management. In *Proceedings of the Fourteenth EuroSys Conference 2019*. New York: ACM.
27. Clark, J., P. van Oorschot, S. Ruoti, K. Seamons, and D. Zappala. 2021. SoK: Securing email—a stakeholder-based analysis. In *Proceedings of the 25th International Conference on Financial Cryptography and Data Security*.
28. Ruoti, S., and K. Seamons. 2019. Johnny's journey toward usable secure email. *IEEE Security Privacy* 17 (6): 72–76.
29. Reeder, R. W., A. P. Felt, S. Consolvo, N. Malkin, C. Thompson, and S. Egelman. 2018. An experience sampling study of user reactions to browser warnings in the field. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. New York: ACM.
30. Acer, M. E., E. Stark, A. P. Felt, S. Fahl, R. Bhargava, B. Dev, M. Braithwaite, R. Sleevi, and P. Tabriz. 2017. Where the wild warnings are: Root causes of Chrome HTTPS certificate errors. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. New York: ACM.
31. Felt, A. P., R. W. Reeder, A. Ainslie, H. Harris, M. Walker, C. Thompson, M. E. Acer, E. Morant, and S. Consolvo. 2016. Rethinking connection security indicators. In *Symposium on Usable Privacy and Security (SOUPS)*.
32. Felt, A. P., A. Ainslie, R. W. Reeder, S. Consolvo, S. Thyagaraja, A. Bettes, H. Harris, and J. Grimes. 2015. Improving SSL warnings: Comprehension and adherence. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. New York: ACM.
33. Akhawe, D., and A. P. Felt. 2013. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *USENIX Security Symposium*.
34. Akhawe, D., B. Amann, M. Vallentin, and R. Sommer. 2013. Here's my cert, so trust me, maybe? Understanding TLS errors on the web. In *Proceedings of the 22nd International Conference on World Wide Web*. New York: ACM.
35. Sunshine, J., S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. 2009. Crying wolf: An empirical study of SSL warning effectiveness. In *USENIX Security Symposium*.
36. Krombholz, K., K. Busse, K. Pfeffer, M. Smith, and E. von Zezschwitz. 2019. "if HTTPS were secure, i wouldn't need 2FA"-end user and administrator mental models of HTTPS. In *IEEE Symposium on Security and Privacy*.
37. Brubaker, C., S. Jana, B. Ray, S. Khurshid, and V. Shmatikov. 2014. Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations. In *IEEE Symposium on Security and Privacy*. Piscataway: IEEE.
38. Fahl, S., M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. 2012. Why Eve and Mallory love android: An analysis of android SSL (in) security. In *ACM Conference on Computer and Communications Security (CCS)*. New York: ACM.
39. Georgiev, M., S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. 2012. The most dangerous code in the world: Validating SSL certificates in non-browser software. In *ACM Conference on Computer and Communications Security (CCS)*, 38–49. New York: ACM.
40. Bates, A., J. Pletcher, T. Nichols, B. Hollembaek, D. Tian, K. R. Butler, and A. Alkhelaifi. 2014. Securing SSL certificate verification through dynamic linking. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. New York: ACM.

41. O'Neill, M., S. Heidbrink, S. Ruoti, J. Whitehead, D. Bunker, L. Dickinson, T. Hendershot, J. Reynolds, K. Seamons, and D. Zappala. 2017. Trustbase: An architecture to repair and strengthen certificate-based authentication. In *USENIX Security Symposium*.
42. O'Neill, M., S. Heidbrink, J. Whitehead, T. Perdue, L. Dickinson, T. Collett, N. Bonner, K. Seamons, and D. Zappala. 2018. The secure socket API:TLS as an operating system service. In *USENIX Security Symposium*.
43. Huang, L. S., A. Rice, E. Ellingsen, and C. Jackson. 2014. Analyzing forged SSL certificates in the wild. In *IEEE Symposium on Security and Privacy*. Piscatawy: IEEE.
44. O'Neill, M., S. Ruoti, K. Seamons, and D. Zappala. 2016. TLS proxies: friend or foe? In *Proceedings of the Internet Measurement Conference (IMC)*. New York: ACM.
45. Durumeric, Z., Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, and V. Paxson. 2017. The security impact of HTTPS interception. In *Network and Distributed Systems Symposium*.
46. de Carné de Carnavalet, X., and M. Mannan. 2016. Killed by proxy: Analyzing client-end TLS interception software. In *Network and Distributed System Security Symposium*.
47. O'Neill, M., S. Ruoti, K. Seamons, and D. Zappala. 2017. TLS inspection: How often and who cares? *IEEE Internet Computing* 21 (3): 22–29.
48. Ruoti, S., M. O'Neill, D. Zappala, and K. Seamons. 2016. User attitudes toward the inspection of encrypted traffic. In *Symposium on Usable Privacy and Security (SOUPS)*.
49. Bonneau, J., C. Herley, P. C. V. Oorschot, and F. Stajano. 2012. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy*.
50. ITRC Data Breach Overview 2007 to 2017, Identity Theft Resource Center, 2018. https://itrcold.cmctempsites.com/images/breach/Overview20052017.pdf.
51. 2017 Data Breach Investigations Report. Verizon. 2017. https://www.knowbe4.com/hubfs/rp_DBIR_2017_Report_execsummary_en_xg.pdf.
52. Lang, J., A. Czeskis, D. Balfanz, M. Schilder, and S. Srinivas. 2016. Security keys: Practical cryptographic second factors for the modern web. In *International Conference on Financial Cryptography and Data Security (FC)*.
53. Das, S., A. Dingman, and L. J. Camp. 2018. Why Johnny doesn't use two factor: A two-phase usability study of the FIDO U2F security key. In *2018 International Conference on Financial Cryptography and Data Security (FC)*.
54. Reynolds, J., T. Smith, K. Reese, L. Dickinson, S. Ruoti, and K. Seamons. 2018. A tale of two studies: The best and worst of yubikey usability. In *IEEE Symposium on Security and Privacy*.
55. Reese, K., T. Smith, J. Dutson, J. Armknecht, J. Cameron, and K. Seamons. 2019. A usability study of five two-factor authentication methods. In *Symposium on Usable Privacy and Security (SOUPS)*.
56. Colnago, J., S. Devlin, M. Oates, C. Swoopes, L. Bauer, L. Cranor, and N. Christin. 2018. "it's not actually that horrible": Exploring adoption of two-factor authentication at a university. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*.
57. Dutson, J., D. Allen, D. Eggett, and K. Seamonsy. 2019. "Don't punish all of us": Measuring user attitudes about two-factor authentication. In *European Workshop on Usable Security (EuroUSEC)*.
58. Winter, P., A. Edmundson, L. M. Roberts, A. Dutkowska-Żuk, M. Chetty, and N. Feamster. 2018. How do Tor users interact with onion services? In *USENIX Security Symposium*.
59. Gallagher, K., S. Patil, and N. Memon. 2017. New me: Understanding expert and non-expert perceptions and usage of the Tor anonymity network. In *Symposium on Usable Privacy and Security (SOUPS)*.
60. Clark, J., P. C. Van Oorschot, and C. Adams. 2007. Usability of anonymous web browsing: An examination of Tor interfaces and deployability. In *Symposium on Usable Privacy and Security (SOUPS)*. New York: ACM.
61. Norcie, G., K. Caine, and L. J. Camp. 2012. Eliminating stop-points in the installation and use of anonymity systems: A usability evaluation of the Tor browser bundle. In *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETS)*. Princeton: Citeseer.

62. Norcie, G., J. Blythe, K. Caine, and L. J. Camp. 2014. Why Johnny can't blow the whistle: Identifying and reducing usability issues in anonymity systems. In *Proceedings 2014 Workshop on Usable Security*. Princeton: Citeseer. https://doi.org/10.14722/usec
63. Lee, L., D. Fifield, N. Malkin, G. Iyer, S. Egelman, and D. Wagner. 2017. A usability evaluation of Tor launcher. *Proceedings on Privacy Enhancing Technologies* 2017 (3): 90–109.
64. Eskandari, S., J. Clark, D. Barrera, and E. Stobert. 2018. A first look at the usability of Bitcoin key management. arXiv:1802.04351.
65. Krombholz, K., A. Judmayer, M. Gusenbauer, and E. Weippl. 2016. The other side of the coin: User experiences with Bitcoin security and privacy. In *International Conference on Financial Cryptography and Data Security*. Berlin: Springer.
66. Gao, X., G. D. Clark, and J. Lindqvist. 2016. Of two minds, multiple addresses, and one ledger: Characterizing opinions, knowledge, and perceptions of Bitcoin across users and non-users. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. New York: ACM.
67. Levine, B. N., and B. Lynn. 2020. Tor hidden services are a failed technology, harming children, dissidents and journalists. In *Lawfare*.