



# BACHELOR'S THESIS

SAFE MESSENGER BASED ON EMAIL

Author

Nikola Radovanovic

intended academic degree

Bachelor of Science (BSc)

Vienna, 2025

Study code according to study sheet: UA 033 521

Field of study: Computer science

Supervisor: Univ.-Prof. Dipl.-Ing. Dr. Helmut Hlavacs

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Related Work</b>	<b>6</b>
<b>3</b>	<b>Technical Foundation</b>	<b>10</b>
<b>4</b>	<b>Safe Messenger based on Email</b>	<b>13</b>
4.1	Communication and Key Management . . . . .	16
4.2	Backup and Recovery . . . . .	17
4.3	Secure Email Communication with Hybrid Encryption (Internal)	18
4.4	Case Study: Multi-Client Accessibility and Message Persistence .	19
<b>5</b>	<b>Evaluation and Discussion</b>	<b>21</b>
<b>6</b>	<b>Conclusion and Future Work</b>	<b>24</b>

# 1 Introduction

Email remains one of the oldest yet most indispensable tools for digital communication, central to both professional and personal contexts worldwide. Despite its vast global reach—with billions of active users and hundreds of billions of messages exchanged daily [27]—email paradoxically remains an open and often insecure system. The very features that underpin its universality also introduce significant vulnerabilities in today’s interconnected world.

A critical limitation of email is its inherently open nature: anyone with an email address can contact any other user across organizational and national boundaries without meaningful restrictions. While this openness enables seamless communication, it simultaneously introduces substantial risks. Notably, email lacks built-in mechanisms to distinguish reliably between trusted in-group contacts—such as colleagues within a company—and unknown external correspondents. This absence of contextual separation means emails typically arrive as an unstructured stream, mixing project messages, personal communications, and external queries without clear boundaries. Such poor contextualization impedes efficient collaboration and facilitates social engineering attacks like phishing [26].

Traditional email protocols, notably the Simple Mail Transfer Protocol (SMTP), were designed without strong authentication or confidentiality guarantees [18]. Email sender addresses can be trivially spoofed, and messages are not end-to-end encrypted by default. Adversaries exploit these weaknesses to impersonate trusted contacts, distribute malware, and deceive users into divulging sensitive information. Phishing alone remains one of the most costly cybersecurity threats globally, affecting individuals and organizations alike through fraudulent identities and malicious links [16].

To mitigate email fraud and spoofing, domain-level standards such as the Sender Policy Framework (SPF), DomainKeys Identified Mail (DKIM), and Domain-based Message Authentication, Reporting and Conformance (DMARC) have been introduced [25]. These frameworks allow domain owners to specify authorized sending servers, verify the integrity of messages, and define policies for handling unauthenticated emails. However, they operate exclusively at the infrastructure level and cannot reliably enforce security policies at the point of user interaction. Misconfigured policies, forwarded messages, and cleverly forged domains continue to allow sophisticated attackers to bypass server-side defenses [6]. Moreover, technical controls alone cannot eliminate social engineering risks, as user behavior remains the weakest link in cybersecurity [31].

Beyond these security challenges, modern workplaces face organizational communication hurdles. Collaboration increasingly relies not only on email but also on dedicated tools such as Slack [1] and Microsoft Teams which structure discussions within clearly defined groups [22]. While these platforms provide valuable contextualization and confine conversations within trusted circles, dependence on multiple disconnected applications fragments communication workflows, making project tracking and document retrieval cumbersome. In stark

contrast, traditional email offers little native support for context-aware collaboration and fails to clearly differentiate between “in-group” members and external parties [2]. Consequently, sensitive discussions risk exposure, and phishing attacks thrive by masquerading as routine internal emails.

This disconnect reveals a significant gap in current digital communication systems. Neither existing email infrastructure nor popular clients effectively enforce group boundaries or prevent phishing at the endpoint. Server-side protections cannot substitute for user-facing policy enforcement, which is largely absent, and user workflows are insufficiently integrated with security controls, leaving critical vulnerabilities exposed.

Therefore, the overarching motivation of this thesis is both practical and conceptual. Practically, organizations require solutions that embed stronger, context-aware security directly within their email clients with minimal configuration effort. Conceptually, this work proposes reimagining email not as a broadly open messaging system, but as a foundation for a “safe messenger.” Such a messenger would incorporate successful paradigms from modern groupware, like strict communication boundaries and workspace channels modeled on platforms such as Slack, while preserving the universal reach and open standards intrinsic to email.

The specific focus is to enable **maximum security** and **minimal setup**, ensuring users can enhance communication security without new accounts or complex installations. Achieving this balance is essential to foster adoption and maintain productivity. Guided by this goal, the thesis addresses the following research question:

**RQ1: Is it possible to design a usable email client that effectively mitigates phishing attacks?**

Based on this research question, the central hypothesis is formulated as follows:

**H1: We can design such a system purely based on email and encryption.**

To investigate this, the thesis proposes a prototype “Safe Messenger based on Email.” The client enforces strict communication policies that allow users to interact only within explicitly defined groups unless explicitly authorized otherwise. Incoming messages from outside these groups are blocked or quarantined, irrespective of the sender’s claimed identity. Outgoing emails to external recipients are similarly restricted or marked. Internal communications are protected by end-to-end client-side encryption and authentication mechanisms that reduce spoofing and phishing risks. Additionally, the system supports carefully controlled, curated communication with external parties under enhanced security measures.

By combining traditional email protocols with modern cryptographic techniques and clearly defined group boundaries, this approach aims to create a secure, organized, and usable communication environment. It demonstrates

how endpoint security can complement server-side protections, improving organizational security and teamwork while minimizing workflow fragmentation.

In summary, this thesis contributes: (1) a comprehensive analysis of security and collaboration shortcomings in current email systems related to group boundaries and contextual awareness, (2) the design of a secure, context-aware email client that enforces in-group policies and phishing prevention, and (3) an evaluation of usability and feasibility in practical deployment scenarios. This contribution is relevant for both academic research in secure communication and for practical implementations in enterprise environments.

## 2 Related Work

While electronic mail (email) remains the foundational infrastructure for asynchronous digital communication, its architecture is inherently open and universal, lacking security primitives to enforce organizational separation or group boundaries. In recent years, the increasing need for in-group/out-group distinction, especially in the context of phishing, data leaks, and collaboration, has spurred both industrial innovation and academic scrutiny. In this section, we examine technical literature and standards on (a) authentication and sender validation, (b) group-based security and messaging, and (c) enterprise attempts to implement messenger-style safeguards within the email ecosystem.

The early protocols defining electronic mail—namely SMTP [19] and IMAP [8] were designed in a pre-adversarial era, in which universal good faith was assumed and little emphasis was placed on sender authentication or access control. The ability to communicate with anyone on the Internet, while a core strength of email, also exposes it to vulnerabilities such as spam, phishing, and spoofing.

To address the persistent problem of sender address spoofing and impersonation, the community developed several email authentication standards, including Sender Policy Framework (SPF, RFC 7208) [17], DomainKeys Identified Mail (DKIM, RFC 6376) [9], and Domain-based Message Authentication, Reporting, and Conformance (DMARC, RFC 7489) [20]. These frameworks allow domain owners to specify cryptographic or policy-based constraints that receivers can use to evaluate message authenticity. In practice, however, large-scale adoption and enforcement of DMARC remain limited: most domains never reach the enforcement stage necessary to actively prevent spoofed emails, with fewer than 20% of domains implementing strong DMARC policies that would actually block unauthorized messages [33]. As a result, these protocols primarily help to identify and monitor unauthorized email rather than to automatically exclude it, and technical challenges in deployment can further undermine their effectiveness.

Classic approaches for enhancing email security have focused on end-to-end encryption and sender verification, most notably S/MIME [28] and PGP [35]. These protocols provide strong cryptographic guarantees, but do not enforce communication boundaries by group membership or organizational affiliation. As Ruoti et al. conclude: “The typical user is not able to correctly configure S/MIME or PGP such that only in-group communications are possible, and the key management burden is substantial” [29]. Whitten and Tygar famously demonstrated that even technically competent users face significant challenges configuring secure email correctly, often leading to failed encryption or accidental exposure [34]. More recent usability studies confirm that key management and group-based encryption remain non-trivial for most users, especially in dynamic organizational contexts [30].

Modern secure messaging platforms place strong emphasis on well-defined group membership, user authorization, and clear separation of communication channels. According to Bonneau et al. [5], all successful secure messaging applications treat group boundaries as essential system components. They explain that membership changes, such as user arrival, departure, or expulsion, are managed both at the protocol layer and within the user interface. This integration ensures that the system remains consistent, transparent, and user-friendly when group composition changes.

In enterprise communication environments, Gajek and Schwenk [13] analyze various approaches to establishing and maintaining secure group sessions. Their work highlights the importance of dynamic group management in open and distributed systems, where users and their permissions can change frequently. This concept is crucial for maintaining both security and usability in real-world group communication settings.

The Matrix protocol [15] provides a practical example of these principles. In Matrix, each communication room operates as a cryptographically enforced group, where access control is fine-grained and metadata such as user roles and invitations are both logged and auditable. This design not only improves transparency and accountability but also demonstrates how group-based security can be achieved in decentralized systems.

Despite the success of modern messaging platforms such as Signal, WhatsApp, and Matrix in implementing secure and manageable group communication, traditional email protocols still lack such functionality. Incorporating similar group-aware security concepts into email standards could significantly improve the confidentiality, integrity, and coordination of group communications in the future.

There is a long tradition of seeking to overlay messenger-style boundaries onto the email substrate. Two notable lines of research are:

1. Secure Group Email: Balenson et al. [4] proposed an architecture for group email security, combining message encryption with centralized policy enforcement, arguing that “access control and group membership management are inherently more complex in the store-and-forward email environment than in synchronous messengers.” Their implementation, while flexible, required centralized group key management infrastructure and has not seen commercial adoption.

2. Group Context and Awareness: Dourish and Bellotti’s groupware work [11] showed that “awareness and coordination features are critical in collaborative workspaces,” an insight now standard in commercial collaboration platforms (Slack, Teams). However, native email clients, as surveyed by Garfinkel & Lipford [14, 29], typically lack mechanisms for context-aware filtering, sender-group affiliation visualization, or binding organizational attributes to messages.

3. Enforced Boundaries at Client or Server: Server-side filtering and policy enforcement, as surveyed in [14, 29], is feasible for monolithic organizations, but “is less effective in federated, cross-organizational environments common in academia and partnerships.” Tariq et al. [32] describe a prototype built atop

open-source email clients that “uses organization directory integration to prevent both incoming and outgoing mail between defined in-group and out-group addresses, irrespective of sender claims.” Their system demonstrates that “effectiveness in group-based boundaries is most robust when enforced client-side and supported by explicit directory or cryptographic policy integration.” However, real-world deployment at scale remains elusive.

Beyond explicit policy enforcement, some research addresses user aids for boundary awareness. Canfield et al. [7] analyze the effectiveness of interface indicators distinguishing in-group/out-group senders. They find that “users exposed to organization-based visual cues are significantly less likely to fall for targeted phishing attacks,” but such cues are generally not integrated with core mail protocols, limiting their deployment and standardization.

A recurring challenge is the openness and federated trust model of email. Attempts to overlay access control lists or group-scoped encryption keying have been stymied by inconsistent directory integration, legacy protocol constraints, and the diversity of email hosting architectures.

Existing industry solutions often implement allowlists, deny-lists, and header inspection at the gateway, but “these are brittle in distributed environments with frequent personnel changes and complex collaborations” [14, 30]. Even standards efforts like DMARC have focused more on origin authentication than in-group exclusivity [12]. To summarize, the literature identifies several open challenges:

- **Enforcement of group boundaries:** As shown in [32], while prototypes exist, there is no mature, user-deployable tool that prevents email exchange with out-group addresses at the client level and is compatible with evolving group membership.
- **Integration of cryptography and access control:** While standards like S/MIME and PGP ([28, 35]) supply the primitives for confidentiality and authentication, they do not enforce messaging only within a defined set of recipients. Management of group membership and associated key material remains a major usability and operational challenge, as detailed by [29] and [30].
- **Visualization and user interaction:** Security indicators can help, but are not enough; enforcement must be “default-on” and transparent, not reliant on user vigilance as per [7].
- **Email universality and inter-domain constraints:** As highlighted by Garfinkel [14], any solution must address cross-domain trust and interoperation while maintaining usability and respecting the decentralized nature of email.

In conclusion, while policy-based, cryptographic, and interface-driven mechanisms for secure, in-group email messaging have all been studied, no open,

standardized, or widely-deployed solution yet provides the robust "messenger-like" in-group/out-group enforcement needed in sensitive digital collaboration on top of ubiquitous email protocols. The present thesis addresses precisely this open research niche.

### 3 Technical Foundation

Email communication relies on a robust set of standardized Internet protocols that enable efficient transmission, storage, and retrieval of messages across diverse and distributed systems. Among these, the Simple Mail Transfer Protocol (SMTP) serves as the principal protocol for sending emails, while the Internet Message Access Protocol (IMAP) manages the retrieval and synchronization of mail on client devices [19, 21]. A comprehensive understanding of these protocols, alongside their inherent security considerations, forms the technical basis necessary for developing a secure and group-oriented email client, which is the focus of this thesis.

SMTP functions as a push protocol, facilitating the transfer of outbound emails from a client or intermediate mail server to the recipient’s mail server [19]. SMTP typically operates on TCP port 25 for plain text, while submission and encrypted connections are commonly served via ports 587 (STARTTLS) and 465 (SMTPS). Originally, SMTP was not designed with authentication or encryption in mind, allowing attackers to spoof sender addresses and intercept messages. To address these vulnerabilities, modern SMTP servers support STARTTLS, a command used to upgrade an insecure connection to a secure encrypted connection using Transport Layer Security (TLS) [3]. SMTP sessions protected by TLS ensure that message content and authentication credentials are encrypted during transmission, thus deterring man-in-the-middle attacks and eavesdropping [10].

On the recipient side, IMAP enables clients to access, organize, and synchronize messages stored on the mail server [21]. Unlike protocols that download messages locally and delete them (such as POP3), IMAP preserves the state of messages across multiple devices and sessions by maintaining folders and read/unread status on the server. IMAP usually operates on port 143, with a secure SSL/TLS variant on port 993 [3]. Securing IMAP sessions with TLS similarly provides confidentiality and integrity protection for message retrieval, safeguarding the client from interception or alteration of email during synchronization.

Despite securing transport, email protocols inherently lack comprehensive sender authentication at the user-visible level. SMTP allows the sender address to be set arbitrarily, facilitating spoofing that attackers exploit to launch phishing attacks, impersonating trusted correspondents [26, 16]. To combat this, domain-level policies such as SPF, DKIM, and DMARC have been established. SPF enables domain owners to specify which mail servers are authorized to send mail on their behalf, DKIM attaches cryptographically signed headers to verify message integrity and domain origin, and DMARC enables policy enforcement and reporting regarding unauthenticated messages [25]. Although these frameworks improve domain authentication, they operate mainly on the infrastructure level and cannot entirely prevent spoofed or phishing messages from reaching end-users [6].

End-to-end confidentiality and message integrity require encryption at the

message level, which is not provided by default in standard email protocols [3]. To this end, cryptographic standards such as S/MIME and OpenPGP have been developed, both relying on asymmetric encryption with public and private keys to encrypt message content and to attach digital signatures [21]. S/MIME relies on centralized Public Key Infrastructures (PKI) and digital certificates, while OpenPGP employs a decentralized web-of-trust model for key validation [10]. User adoption has been hampered by complexity around key management and interoperability between clients [26].

The approach in this thesis addresses these challenges using hybrid encryption, a method that combines the advantages of symmetric and asymmetric cryptography [21]. For each message, a random symmetric key (e.g., AES) encrypts the email content efficiently. This key is then encrypted asymmetrically (e.g., RSA-OAEP 2048-bit) for each intended recipient’s public key, ensuring only authorized parties can access the symmetric key to decrypt the message [21]. This hybrid scheme efficiently secures message content and facilitates controlled group communication by enabling the sender and recipients to share keys securely.

Key generation and management occur entirely on the client side, consistent with a zero-knowledge design philosophy. Each user’s private key is generated locally (in the browser for a web-based client), never transmitted to the server unencrypted, and stored securely on the client device [10, 23]. The corresponding public key is sent to the server and made available for encrypting messages to the user or verifying signatures. To guard against loss of keys, a recovery passcode system encrypts and backs up the private key to the server in a form that only the client can decrypt upon entering the passcode. This design ensures that the server operator cannot decrypt user messages, reinforcing user privacy and security [23].

All client-server communications for mail retrieval and submission occur over encrypted TLS channels, preventing interception of credentials, commands, or message content in transit [10]. TLS 1.3, the current standard, provides improved security and performance, including forward secrecy and resistance to several classes of attacks [10]. Secure channels are used not only for SMTP and IMAP connections but also for all auxiliary communication with backend services managing workspaces, channels, and key metadata.

Modern email client architectures commonly support dual operational modes. The first, an external mode, functions as a standard email client enabling communication with the general Internet. The second, an internal mode, introduces workspace-like structures, facilitating group-centric communication with collaboration features reminiscent of platforms such as Slack or Microsoft Teams [24, 2].

The internal mode enforces strict group boundaries by blocking any inbound or outbound email traffic to addresses outside the defined group, regardless of the sender field, preventing phishing and unauthorized contact. Messages within the workspace are encrypted end-to-end with the aforementioned hybrid scheme, ensuring confidentiality and authenticity. The client application transparently manages encryption, key exchange, and message decryption without requiring

user intervention or knowledge of cryptographic details, supporting usability and broad acceptance.

In conclusion, the technical foundation underpinning a secure, phishing-resistant email client combines classical Internet mail protocols—SMTP and IMAP—with modern transport layer encryption (TLS) and message-level hybrid cryptography. Coupled with client-side key management and zero-knowledge security principles, this architecture enables contextual, group-limited messaging built atop the global email infrastructure. The design preserves email’s universality while adding crucial security and usability improvements, directly addressing the research question and hypothesis of this thesis.

## 4 Safe Messenger based on Email

This chapter introduces the architecture and key features of the Safe Email Client, a system designed to enforce strict in-group and out-group communication boundaries while preserving the fundamental versatility of traditional email. Our approach keeps the familiar email transport (SMTP/IMAP) but adds strict, client- and server-side checks that ensure messages stay within a defined workspace. The solution is structured around two major components—a secure backend service and an intuitive frontend client—that together provide both security and usability. The frontend offers a channel-like experience similar to team messengers, while the backend acts as a policy enforcement point that integrates seamlessly with standard mail protocols.

The system architecture, illustrated in Figure 1, consists of a backend implemented in Python using FastAPI, and a frontend web interface built with Vue.js and Bootstrap. The backend mediates all contacts with real email servers via IMAP and SMTP, acting on behalf of users to ensure compatibility with organizational infrastructures and support for external mail services. In practice, it terminates HTTPS, validates short-lived tokens (JWT), applies workspace policies on every send/receive operation, and relays permitted traffic to the user’s actual mailbox. User authentication, workspace management, and session handling are managed centrally within this backend, which also maintains up-to-date records of group membership and enforces access control lists relevant to each workspace or organization. The backend stores only minimal metadata (addresses, timestamps, delivery flags) and avoids persisting plaintext message content; internal messages are processed as opaque ciphertext to preserve end-to-end confidentiality. For cryptographic usability, the backend provides a public-key directory for in-group discovery, while private keys remain on the client (or are backed up only in client-side-encrypted form). This separation of roles allows a clean integration with existing providers, multi-tenant deployments, and clear auditing of policy decisions without sacrificing the openness of email.

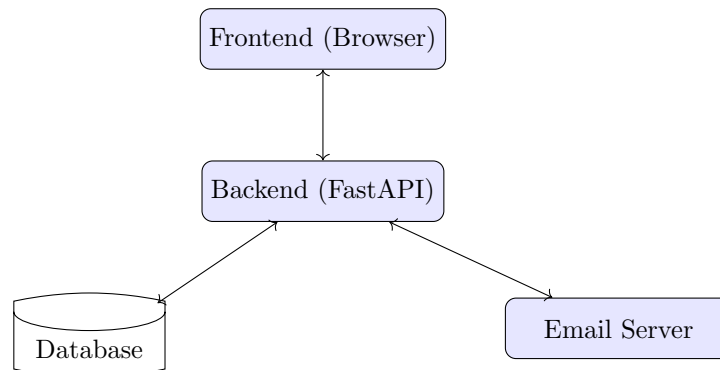


Figure 1: System architecture of the Safe Email Client.

To safeguard group boundaries, the backend implements comprehensive filtering policies: incoming and outgoing messages are examined so that only communication permitted by the workspace settings is allowed. Any attempt to correspond with out-group addresses is automatically blocked or flagged according to defined policy. Security is further strengthened through a strictly protected REST API layer, which is accessible only through authenticated, encrypted HTTPS connections. Importantly, to maximize privacy, the backend is designed to store only basic metadata—such as sender and recipient addresses, timestamps, and delivery status flags—while the content of messages or attachments is never persisted except in encrypted, in-memory processing.

On the client side, the frontend is implemented as a modern single-page application (SPA) that interacts with the backend exclusively via API calls. The design emphasizes clarity and ease of use, presenting email threads and channels in a unified interface inspired by popular collaboration platforms such as Slack [1]. Each workspace is treated as a distinct communication channel, with messages clearly labeled according to their organizational context and sender/recipient affiliation, providing users with an intuitive view of group boundaries.

The frontend is deliberately designed for **minimal configuration**, requiring only a standard web browser and an active internet connection to operate. The first step for a new user is to register a supported email address and provide a display name; there is no need to create an additional email account, as the system leverages the user’s existing email infrastructure.

After successful registration, as shown in Figure 2, the system generates and displays a unique **recovery key** that must be securely stored by the user. This recovery key is essential for account recovery and private key restoration when accessing the system from different browsers, devices, or after clearing browser data. The key is cryptographically generated and displayed only once during the initial registration process, emphasizing the importance of immediate secure storage. Users are strongly advised to save this key in a secure password manager or other protected storage medium.

Once the recovery key has been acknowledged and stored, the user can proceed to log in with their email address and the password associated with their existing email provider. The functioning of the recovery key mechanism and its role within the cryptographic key management system will be discussed later in this section.

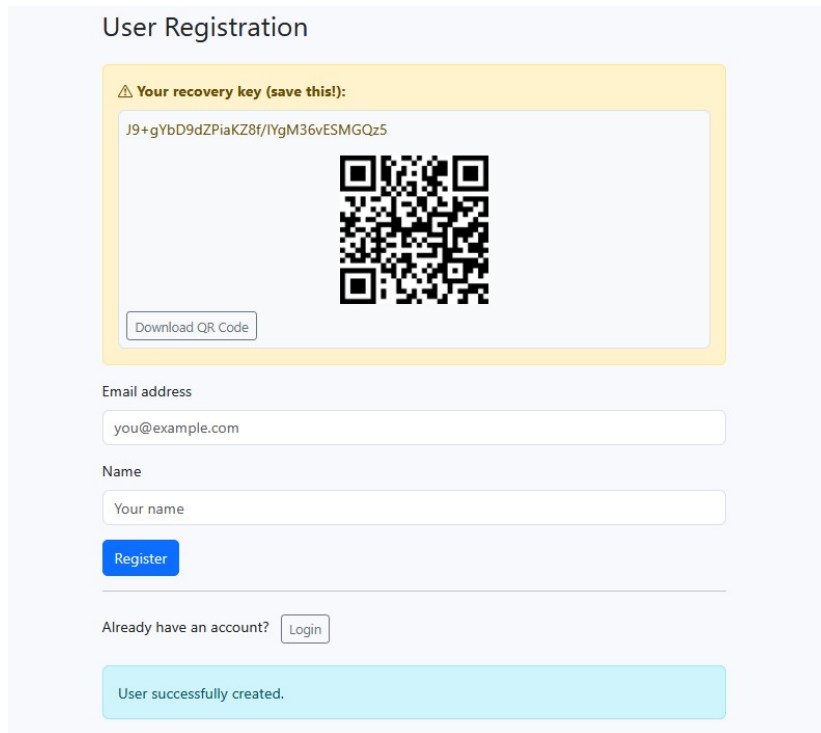


Figure 2: Successful user registration

The registration process maintains a careful balance between security and usability by eliminating the need for complex configuration while ensuring that users maintain control over their cryptographic credentials through the recovery key mechanism.

End-to-end encryption is employed for message contents whenever possible, with all cryptographic operations and key storage taking place entirely within the browser. This ensures that plaintext messages never leave the client device and remain invisible to the backend at all times. The interface incorporates modern usability standards from frameworks such as Bootstrap, guaranteeing responsive layouts, clear visual cues for group and channel membership, and minimal risk of user error even under restrictive security policies. By combining familiar design patterns from Slack with robust security practices, the frontend provides both accessibility and protection without imposing additional configuration burdens on the user.

The Safe Email Client can operate in one of two modes. In *external mode*, the application behaves like a conventional email client, permitting users to send and receive messages to and from any email address, internal or external, without special restrictions. This mode is intended for general correspondence and scenarios in which open communication is appropriate. Conversely, *internal*

*mode* enforces strict constraints: users are prevented from sending or receiving messages outside their designated workspace or organization, ensuring that all communication is contained within predefined group boundaries. Any attempt at external correspondence is blocked at the application level, providing a high assurance that sensitive discussions and data remain inside trusted domains.

Figure 3 provides an example of the user interface navigation, enabling users to easily switch between different modes and workspaces, while always making the current context and communication boundaries explicit.

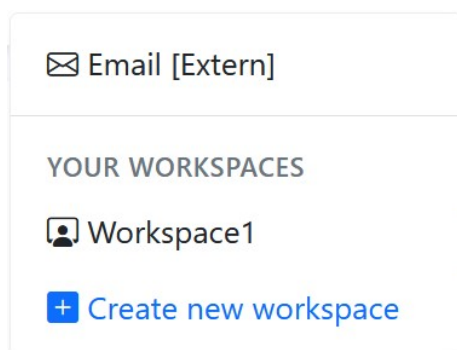


Figure 3: Application navigation on the user interface

## 4.1 Communication and Key Management

Users begin by entering their email address and the corresponding password for the real email server directly into the frontend application. Upon successful authentication, the backend generates a JSON Web Token (JWT), which it securely stores together with the encrypted email server password in its database. This token is then returned to the frontend and must be included in the HTTP header of every subsequent API request, typically formatted as **Authorization: Bearer <token>**. When the backend receives such a request, it verifies the token's validity, decrypts the stored password if necessary, and uses these credentials to connect to the actual email server via IMAP or SMTP on behalf of the user. Session time limits are enforced to improve security; once the token expires, the user must reauthenticate, thereby invalidating the previous token and preventing unauthorized access.

Each user maintains a unique asymmetric key pair consisting of a public and a private key. Key generation occurs exclusively on the client side, ensuring full user control and enhanced privacy. The public key is automatically sent to and stored on the backend server, enabling secure encrypted communication between users. The private key, however, remains securely stored only on the user's device. This design operates transparently, without requiring the user to manually export or import keys, thus simplifying setup and usage. During initial registration, users are also provided with a recovery code, known as the

*Recovery Passcode*, which serves as a backup mechanism for key restoration in case the private key is lost or the device becomes inaccessible.

The creation of the user’s key pair is performed entirely on the client side, within the browser environment, using the RSA-OAEP algorithm. After generation, the private key is exported in PKCS8 format and immediately encrypted using an AES-GCM key. This AES key is derived from the user’s recovery password through PBKDF2, combined with a randomly generated salt. The encrypted private key, along with the salt and initialization vector (IV), is then transmitted to and securely stored on the backend server.

When a user needs to restore their private key, for instance on a new device, they enter their *Recovery Passcode*, which allows the client to re-derive the AES key and decrypt the private key locally. Once decryption succeeds, the private key is imported back into the browser and becomes available for message decryption operations. Table 1 summarizes the cryptographic architecture underlying the key management system.

Area	Description
Key Generation	Client-side (browser); RSA-OAEP 2048 bits; SHA-256 hash
Public Key	Format: SPKI; automatically sent to and stored on the backend
Private Key	Format: PKCS8; remains local; encrypted and securely stored on the backend
Private Key Encryption	AES-GCM 256 bits; PBKDF2 key derivation; Salt 128 bits, IV 96 bits
Recovery	Password-based using Recovery Passcode; decryption only on client side

Table 1: Key management and cryptographic details

## 4.2 Backup and Recovery

During user registration, the private key is encrypted locally on the client before being safely transmitted and stored on the backend server. The backend itself has no capability to decrypt this key, ensuring that only the rightful user can access it. When logging in from a new device, the client checks whether a private key is available locally. If none is found, the user can securely restore it using the previously issued *Recovery Passcode*. This mechanism guarantees that users can regain access to encrypted messages without compromising the confidentiality of their private key.

The scenario shown in the images below demonstrates how the system handles missing key situations. A user logs in on an unknown device and attempts to open an internal direct message within the workspace. Because the private key is not available, the message cannot be decrypted, and the interface displays a warning stating “Cannot decrypt message” along with the error “Your

private key is missing.” When the user navigates to their profile, the system clearly indicates that the private key is missing and prompts the user to recover it by entering the recovery code. This process ensures both usability and data protection during account recovery, as illustrated in Figure 4.

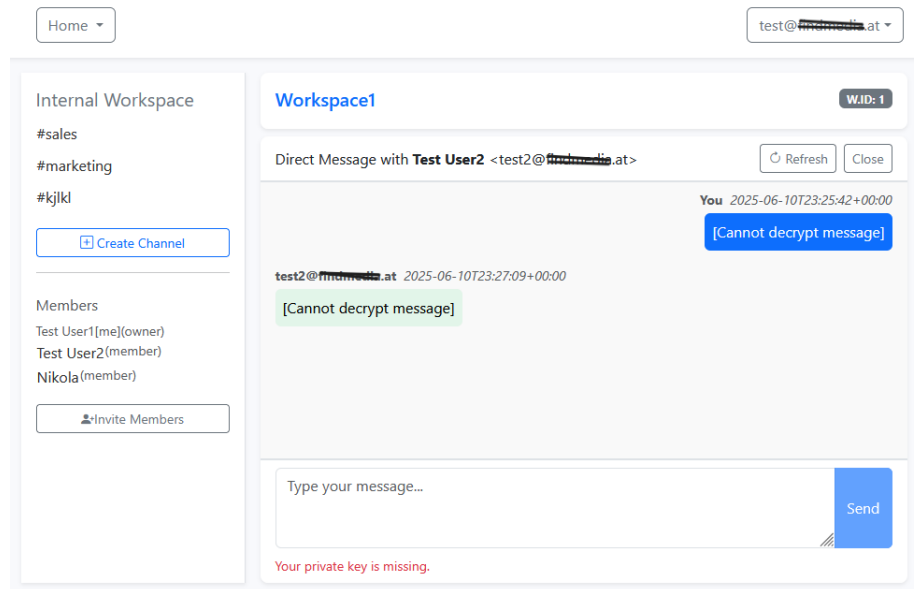


Figure 4: User logged in on an unknown device tries to read an internal direct message. The message content cannot be decrypted because the private key is missing.

### 4.3 Secure Email Communication with Hybrid Encryption (Internal)

The SafeEmailMessenger application is designed to provide secure communication between workspace members while still using standard email infrastructure as the transport medium. In this model, messages are transmitted as regular emails but can only be decrypted and viewed within the SafeEmailMessenger interface. This ensures that even though the underlying transport is conventional email, the actual message content remains protected and inaccessible to traditional email clients.

At the core of this concept lies the principle of *hybrid encryption*. Each internal message is first encrypted using a randomly generated symmetric AES key to ensure efficiency and speed. This AES key is then encrypted twice using asymmetric RSA encryption:

- once with the recipient’s public key, allowing the recipient to decrypt it using their private key and access the message, and

- once with the sender’s public key, enabling the sender to decrypt their own copy and view the message later in the “Sent” folder.

Through this mechanism, both the sender and the recipient are able to decrypt the message content with their respective private keys. As a result, end-to-end encryption (E2EE) is achieved within the workspace, while still maintaining interoperability with standard email delivery protocols. An example of the encrypted message payload, represented in a JSON-like format, is shown below:

```
{
  "subject": "Internal Direct Message (Workspace 1)",
  "from": "john@example.at",
  "to": "doe@example.at",
  "iv": "6jhm01lo2h6Kjho2",
  "encrypted_key": "ChVa7CUNnUJScEyXBr40SNCKoMZF0...",
  "encrypted_body": "LRdrarJP0tSQNobTz370JRD9OVZo+..."
}
```

In this structure, the field “iv” represents the AES initialization vector used for symmetric encryption, “encrypted\_key” contains the AES key encrypted with RSA for both sender and receiver, and “encrypted\_body” stores the ciphertext of the actual message content.

This hybrid encryption approach combines the performance advantages of symmetric cryptography with the security guarantees of asymmetric key exchange. It ensures confidentiality and controlled accessibility while leveraging email as a universal and familiar message transport layer. Consequently, messages remain readable only inside the SafeEmailMessenger application, providing a practical yet secure solution for internal workspace communication.

#### 4.4 Case Study: Multi-Client Accessibility and Message Persistence

A relevant case study considers scenarios where users access their email through multiple clients. While users can log into their email account using any standard email client (such as Outlook or Thunderbird), encrypted messages sent via SafeEmailMessenger will not be readable in these third-party clients—they will only appear as encrypted content. However, this interoperability introduces an important consideration: if a user deletes these encrypted messages through another email client, the messages will be permanently removed from the email server and consequently become inaccessible in SafeEmailMessenger as well. This occurs because the system does not maintain a separate local message store and relies entirely on the email server as the single source of truth for message persistence. This design emphasizes the importance of using SafeEmailMessenger as the primary client for managing encrypted communications while maintaining compatibility with the broader email ecosystem.

In addition to the hybrid encryption mechanism, several other functionalities have already been implemented in the current version of the SafeEmailMessenger

system. One of them is the ability to invite new members to a workspace through an email invitation. A registered user can send an invitation email containing a secure link to another person. By clicking on this link, the invited user can register and automatically join the corresponding workspace. This approach provides a simple and familiar onboarding process that integrates seamlessly with standard email communication while maintaining a secure workflow.

Each user can be a member of multiple workspaces at the same time. This design allows separation between different projects, teams, or organizations, providing flexibility and scalability for collaborative environments. Within each workspace, additional channels can be created to structure communication according to topics or departments, similar to modern team-messaging platforms.

In the internal communication mode, users can exchange direct messages or participate in group discussions within a single channel. Technically, communication inside a channel is realized as the exchange of multiple individually encrypted end-to-end messages between participants. This ensures that even in a group context, every transmitted message maintains the same level of confidentiality and cryptographic protection as a direct message between two users.

## 5 Evaluation and Discussion

This section presents an empirical and inferential evaluation of the Safe Email Client in light of the central research question **RQ1** — whether it is possible to create a usable email client that is safe from phishing — and the hypothesis **H1** that such a system can be designed purely based on email and encryption. The analysis combines descriptive insights from a structured user survey with an exploratory proportion test to provide an initial inferential perspective on user acceptance and perceived security. This mixed-methods approach allows for both quantitative validation of security perceptions and qualitative understanding of adoption barriers in real-world workplace contexts.

The evaluation employed a cross-sectional survey design administered to 13 participants recruited through professional networks and technology-oriented communities. Data collection occurred over a three-week period using Google Forms, with participants representing diverse professional backgrounds to ensure varied perspectives on email usage and security requirements. The survey instrument comprised ten core statements measured on a 4-point Likert scale (0 = strongly disagree to 3 = strongly agree), complemented by demographic questions addressing gender, age, and occupational background.

The participant cohort demonstrated balanced gender distribution with 6 male (46.2%) and 7 female (53.8%) respondents. Age distribution revealed a concentration in the 35-50 year bracket (69.2%), with younger adults aged 18-20 representing 23.1% of the sample, and the 21-34 age group comprising the remaining 7.7%. The absence of participants aged 51+ suggests the sample skewed toward digitally-engaged working professionals, potentially limiting perspectives from older user groups who may face different technological challenges.

Occupational diversity was notable, with IT professionals forming the largest subgroup (33.3%), followed by logistics specialists (16.7%), and individual representatives from administration, hospitality, medicine, sales, software development, and retail sectors. This professional heterogeneity provides valuable insights from both technical domains and non-technical operational roles, though the predominance of IT backgrounds may inflate overall security awareness compared to the general workforce population.

Analysis of email usage behavior revealed that email remains a critical workplace communication tool, with 61.5% of participants reporting daily use of email for professional purposes. This finding aligns with industry reports emphasizing email’s persistent dominance in organizational communication despite the proliferation of alternative messaging platforms.

Phishing exposure data indicated significant security challenges in current email ecosystems. A majority of respondents (53.8%) reported receiving phishing emails within the past week, while 38.5% admitted to having previously clicked on suspicious links or being deceived by phishing attempts. These figures underscore the persistent effectiveness of social engineering attacks even among relatively technology-proficient users. The finding that nearly two-fifths of participants had fallen victim to phishing attempts highlights the critical need for enhanced protective measures in email clients.

Participant confidence in identifying phishing emails showed moderate levels, with 61.6% agreeing or strongly agreeing with their ability to recognize malicious messages (38.5% agree, 23.1% strongly agree). This perceived competence contrasts with the actual phishing victimization rate, suggesting potential overconfidence in phishing detection abilities—a phenomenon documented in cybersecurity literature as the “overconfidence bias” in threat assessment.

A particularly revealing finding emerged from platform preference data: 84.6% of respondents rejected the notion that they preferred alternative messaging tools (e.g., Slack, Microsoft Teams) over email. This strong preference for email as a primary communication medium reinforces the strategic importance of securing email infrastructure rather than attempting to migrate organizational communication to new platforms. The persistence of email as a workplace staple suggests that security innovations within the email ecosystem may achieve greater adoption than competing platforms that require workflow changes.

The evaluation revealed nuanced perspectives on system adoption. While 46.2% of respondents expressed willingness to use Safe Messenger as their default work client, a much stronger majority (92.3%) agreed or strongly agreed that the system could improve communication within their organizations. This disparity between individual adoption intent and perceived organizational benefit suggests that deployment decisions may be influenced by factors beyond individual preference, such as IT policy, interoperability requirements, and migration costs.

Usability perceptions were overwhelmingly positive, with 92.3% of participants disagreeing that the system was too complicated to use. This strong usability rating is particularly significant given the security features implemented, suggesting that the prototype successfully balanced security complexity with user experience simplicity—a critical requirement for hypothesis **H1**.

Security perceptions reached unanimous agreement, with all participants (100%) affirming that Safe Messenger provides better protection against phishing attacks than traditional email clients (69.2% strongly agree, 30.8% agree). However, the advocacy gap persisted, as only 46.2% indicated they would actively recommend the system to colleagues. This discrepancy between personal security satisfaction and willingness to advocate may reflect concerns about implementation complexity, interoperability issues, or organizational suitability rather than doubts about the core security value proposition.

To extend beyond descriptive findings, proportion tests were conducted to determine whether positive response rates significantly differed from a neutral benchmark of 50%. The formal hypothesis framework was defined as:

$$H_0 : P(X = 2 \text{ or } 3) = 0.5$$

$$H_1 : P(X = 2 \text{ or } 3) \neq 0.5$$

For the key security statement “I feel that this email client protects me better against phishing attacks,” all 13 responses were positive (2 or 3), yielding an

observed proportion of 1.0. The two-sided proportion test produced a  $p$ -value  $\leq 0.001$ , allowing rejection of  $H_0$  and supporting the conclusion that positive security perceptions significantly exceed neutral expectations.

Similarly, the statement “I believe this email client would improve communication in my company” received 12 positive responses out of 13 ( $p < 0.05$ ), again significantly surpassing the neutral benchmark. These results provide statistical confidence that the security and communication benefits are not due to random variation.

In contrast, adoption-related items such as “I would consider using this client as my default” showed no statistical significance (6 of 13 positive responses,  $p \approx 0.78$ ), indicating genuinely ambivalent attitudes toward personal adoption despite recognizing the system’s security advantages.

Several limitations warrant consideration when interpreting these findings. The small sample size ( $N = 13$ ) restricts statistical power and generalizability, though the effect sizes observed in security perceptions were substantial enough to achieve significance despite this limitation. The concentration of IT professionals (33.3%) likely inflated technical competence and security awareness relative to broader user populations. Additionally, the age distribution skewed toward middle-aged professionals (35-50 years), potentially underrepresenting both digital natives and older users who may exhibit different security behaviors and adoption patterns.

The evaluation relied exclusively on self-reported perceptions rather than behavioral metrics. Future research should incorporate objective measures such as simulated phishing click-through rates, task completion times, and error rates to complement subjective security assessments. Longitudinal studies tracking actual adoption and usage patterns would provide valuable insights into sustained engagement and real-world security effectiveness.

In conclusion, the analysis demonstrates that Safe Messenger effectively enhances users’ perceived protection against phishing while maintaining usability standards that facilitate positive user experiences. The inferential testing confirms that positive security perceptions are statistically significant, providing substantial support for the central hypothesis **H1** that a usable and phishing-resilient system can be developed using email and encryption technologies.

The findings validate the core research proposition while highlighting the complexity of adoption dynamics in organizational contexts. While security effectiveness is clearly recognized and valued by users, translation into widespread adoption depends on addressing implementation barriers, interoperability concerns, and organizational workflow integration. Future development should focus on enhancing onboarding processes, expanding configuration options for diverse email environments, and demonstrating compatibility with existing enterprise systems.

The findings confirm that robust security and user-friendly design can co-exist in modern email systems. Safe Messenger’s integration of standard infrastructure and end-to-end encryption illustrates a practical path toward safer communication without sacrificing usability.

## 6 Conclusion and Future Work

This thesis addressed a fundamental security challenge in modern digital communication: the inherent structural vulnerability of email systems that allow unlimited message exchange between any parties without built-in trust verification or group boundary enforcement. The open nature of traditional email protocols, while enabling global connectivity, creates significant attack surfaces for spoofing, phishing, and social engineering attacks that compromise organizational security and erode user trust. In response to these critical vulnerabilities, we conceived, designed, and implemented a Safe Messenger based on Email—a novel web-based email client that fundamentally reimagines email communication by enforcing strict communication boundaries within defined trust groups while maintaining controlled interoperability with external contacts.

The core innovation of this work lies in achieving **maximum security** through **minimal setup requirements**, strategically leveraging existing email infrastructure without mandating users to create new email accounts or install complex software packages. This approach significantly lowers adoption barriers while maintaining robust security guarantees. The implemented client architecture operates through two distinct yet complementary modes: the external mode functions as a conventional email client with full compatibility standard email workflows and global internet communication, while the internal mode enforces rigorous group communication policies that effectively create secure workspace channels analogous to modern collaboration platforms like Slack or Microsoft Teams, but operating exclusively through standardized email protocols. Crucially, the system processes all communication via existing email server infrastructure without storing messages locally on client devices, ensuring compatibility with organizational IT policies and backup systems.

From a security architecture perspective, the application integrates multiple advanced protection mechanisms to safeguard user data and ensure communication integrity. All client-server interactions employ JSON Web Token (JWT) authentication with appropriate expiration policies, while connections to email servers are secured through robust SSL/TLS encryption protocols. The system follows a strict zero-knowledge design philosophy where critical security operations—including key generation, encryption, and decryption processes—occur exclusively on the client side, preventing server-side access to plaintext message content under any circumstances. The implementation of end-to-end encryption (E2EE) utilizes hybrid cryptographic approaches that strategically combine the computational efficiency of symmetric encryption with the robust security properties of asymmetric cryptography, ensuring message confidentiality within defined user groups while maintaining practical performance characteristics.

The current system architecture employs a modular design with distinct frontend and backend components, a decision that supports organized development workflows, facilitates independent scaling of system components, and enables specialized optimization of each layer. While consideration has been given to potential future consolidation into a monolithic application structure,

the existing separation provides significant advantages for enterprise deployment scenarios, including the ability to independently update user interface components without disrupting core service functionality and enabling flexible load balancing strategies for high-availability deployments.

Looking forward, **Future Work** will focus on enhancing usability, strengthening security guarantees, and achieving production readiness. The findings indicate that while users recognize the security benefits, adoption depends on seamless integration and ease of use—hence, future development must reduce implementation barriers and improve compatibility with existing enterprise ecosystems.

A primary goal is the establishment of a production-ready hosting environment with enterprise-level reliability, monitoring, and automated update mechanisms. Further development will streamline onboarding processes, enable integration with organizational systems, and support structured team-based deployments with centralized management tools.

Technical improvements will include broader IMAP/SMTP compatibility, comprehensive mobile support, and enhanced key management through multi-factor recovery, biometric authentication, and hardware security keys. Empirical validation through larger, long-term user studies will evaluate usability and adoption under real-world conditions.

Security and performance will be further validated through penetration testing, simulated attack scenarios, and stress testing under high workloads. Advanced research will explore intelligent anomaly detection, integration with Security Information and Event Management (SIEM) systems, and cryptographic enhancements such as forward secrecy and post-compromise protection.

Overall, future work aims to transition the Safe Messenger from a functional prototype to a secure, scalable, and user-friendly communication platform suitable for enterprise deployment.

In conclusion, this thesis demonstrates both the feasibility and practical viability of designing and implementing a secure email client that enforces stringent communication boundaries and employs strong cryptographic protections—without compromising the ubiquitous reach, flexibility, and familiarity that make email an enduring communication medium. By strategically combining traditional email protocols with modern encryption technologies and thoughtfully designed group policies, the Safe Messenger based on Email represents a promising evolutionary step toward more secure, organized, and user-friendly enterprise communication ecosystems. The positive security perceptions and identified adoption considerations provide a solid foundation for continued development, refinement, and eventual real-world deployment of this innovative approach to email security.

## References

- [1] Slack: Ai work management productivity tools, 2025. Accessed: 2025.
- [2] A walled garden among walled gardens: What if email was invented today? — zdnet, 2025. [Online; accessed 1-July-2025].
- [3] What is email security? threats and best practices - perception point, 2025. [Online; accessed 1-July-2025].
- [4] BALENSON, D., PINKAS, D., AND POPE, A. Framework for secure email. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (1999).
- [5] BONNEAU, J., SCHECHTER, S., GREEN, M., AND KIM, S. Sok: Secure messaging. In *IEEE Symposium on Security and Privacy* (2015).
- [6] BRAD SLAVIN, DUOCIRCLE LLC. Can threat actors bypass dmarc? <https://www.duocircle.com/dmarc/can-threat-actors-bypass-dmarc>, 2025. [Online; accessed 2-July-2025].
- [7] CANFIELD, C. I., FISCHHOFF, B., SPAFFORD, E. H., ET AL. The effect of email security indicators on user trust. In *16th Symposium on Usable Privacy and Security (SOUPS 2020)* (2020), pp. 337–354.
- [8] CRISPIN, M. Internet message access protocol - version 4rev1. *RFC*, 3501 (2003).
- [9] CROCKER, D., HANSEN, T., AND KUCHERAWY, M. Domainkeys identified mail (dkim) signatures. *RFC*, 6376 (2011).
- [10] DIGICERT, INC. Buy secure email (s/mime) certificates — digicert. <https://www.digicert.com/tls-ssl/secure-email-smime-certificates>, 2025. DigiCert Secure Email (S/MIME) Certificates – S/MIME certificates for end-to-end email security at any scale.
- [11] DOURISH, P., AND BELLOTTI, V. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work* (1992), pp. 107–114.
- [12] FENTON, J. Email authentication: Lessons learned from dmarc deployment. White paper, Internet Society, 2017. Internet Society Whitepaper.
- [13] GAJEK, S., AND SCHWENK, J. Secure group communication in open distributed systems. *IEEE Security & Privacy* (2007), 56–64.
- [14] GARFINKEL, S. L., AND LIPFORD, H. R. Design principles and patterns for computer systems that are simultaneously secure and usable. In *Proceedings of the 2005 Symposium on Usable Privacy and Security (SOUPS)* (2005), ACM, pp. 75–82.

- [15] HODGES, M., ET AL. Matrix: An open network for secure, decentralized communication. *Open Source Summit North America* (2017).
- [16] HOXHUNT. Phishing trends report (updated for 2025). <https://hoxxhunt.com/guide/phishing-trends-report>, 2025. [Online; accessed 1-July-2025].
- [17] KITTERMAN, S. Sender policy framework (spf) for authorizing use of domains. *RFC*, 7208 (2014).
- [18] KLENSIN, J. Simple mail transfer protocol. *RFC*, 5321 (2008).
- [19] KLENSIN, J. Simple mail transfer protocol. RFC 5321, Internet Engineering Task Force (IETF), October 2008. Accessed: 2025-07-01.
- [20] KUCHERAWY, A., AND ZWICKY, M. Domain-based message authentication, reporting, and conformance (dmarc). *RFC*, 7489 (2015).
- [21] LAUSCH, T., DÜRMUTH, M., AND HOLZ, T. On the security and usability of secure email technologies. In *Proceedings of the 2nd European Symposium on Security and Privacy Workshops (EuroS&PW)* (2017), IEEE, pp. 106–116. Accessed: 2025-07-02.
- [22] MAJCHRZAK, A., MALHOTRA, A., STAMPS, J., AND LIPNACK, J. Can absence make a team grow stronger? *Harvard Business Review* 82, 5 (2004), 131–137.
- [23] MARLINSPIKE, M., AND PERRIN, T. The signal protocol whitepaper. Tech. rep., Open Whisper Systems, 2016. Accessed: 2025-07-01.
- [24] MICROSOFT. The evolution of email: Integrating collaboration tools with outlook and microsoft teams. *Microsoft 365 Blog* (2023). Accessed: 2025-07-01.
- [25] MIMECAST. Spf vs. dkim vs. dmarc: A guide. <https://www.mimecast.com/content/dkim-spf-dmarc-explained/>, 2025. [Online; accessed 3-July-2025].
- [26] PEREMORE, K. The human factors and organizational risks to email security - paubox. <https://www.paubox.com/blog/the-human-factors-and-organizational-risks-to-email-security>, 2025. [Online; accessed 2-July-2025].
- [27] PORCH GROUP MEDIA. 100 compelling email statistics for 2025 — porch group media. <https://porchgroupmedia.com/blog/100-compelling-email-statistics-to-inform-your-strategy-in-2023/>, 2025. [Online; accessed 1-July-2025].
- [28] RAMSDELL, B. Secure/multipurpose internet mail extensions (s/mime) version 3.2 message specification. *RFC*, 5751 (2010).

- [29] RUOTI, S., ANDERSEN, J., HEIDBRINK, T., O'NEILL, K. K., ZAPPALA, D., AND SEAMONS, K. E. Confused johnny: When automatic encryption leads to confusion and mistakes. In *2016 IEEE Symposium on Security and Privacy (SP)* (2016), IEEE, pp. 312–328.
- [30] RUOTI, S., KIM, J. W., BURGON, B., O'NEILL, M. A., AND SEAMONS, K. E. We're on the same page: A usability study of secure email using pairs of novice users. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)* (2015), ACM, p. 1623–1632.
- [31] SASSE, M. A., BROSTOFF, S., AND WEIRICH, D. Transforming the 'weakest link'—a human/computer interaction approach to usable and effective security, 2001.
- [32] TARIQ, U., RASMUSSEN, K. B., AND NIELSEN, T. Towards trusted email communication: Preventing email spoofing and enforcing group boundaries. In *Proceedings of the 12th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (2019), pp. 146–157.
- [33] VALIMAIL, I. Email fraud landscape 2023: Dmarc adoption report, 2023. Accessed: 2025-07-01.
- [34] WHITTEN, A., AND TYGAR, J. D. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th USENIX Security Symposium* (1999).
- [35] ZIMMERMANN, P. R. *The official PGP user's guide*. MIT Press, 1995.