PrivMail: A Privacy-Preserving Framework for Secure Emails

Gowri R. Chandran[1](✉), Raine Nieminen[1], Thomas Schneider[1], and Ajith Suresh[2]

[1] ENCRYPTO, Technical University of Darmstadt, Darmstadt, Germany
chandran@encrypto.cs.tu-darmstadt.de

[2] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE

Abstract. Emails have improved our workplace efficiency and communication. However, they are often processed unencrypted by mail servers, leaving them open to data breaches on a single service provider. Public-key based solutions for end-to-end secured email, such as Pretty Good Privacy (PGP) and Secure/Multipurpose Internet Mail Extensions (S/MIME), are available but are not widely adopted due to usability obstacles and also hinder processing of encrypted emails.

We propose PrivMail, a novel approach to secure emails using secret sharing methods. Our framework utilizes Secure Multi-Party Computation techniques to relay emails through multiple service providers, thereby preventing any of them from accessing the content in plaintext. Additionally, PrivMail supports private server-side email processing similar to IMAP SEARCH, and eliminates the need for cryptographic certificates, resulting in better usability than public-key based solutions. An important aspect of our framework is its capability to enable third-party searches on user emails while maintaining the privacy of both the email and the query used to conduct the search.

To evaluate our solution, we benchmarked transfer and search operations using the Enron Email Dataset and demonstrate that PrivMail is an effective solution for enhancing email security.

Keywords: Private Email · Secret Sharing · Private Keyword Search · Secure Two-party Computation · Private Information Retrieval

1 Introduction

Despite the widespread use of social media, text messages, and online messaging services such as WhatsApp, Signal, and Telegram, electronic mail (email) is still a popular method of communication, and it has a growing user base. In 2020, there were roughly 4 billion users of email; by 2024, it is predicted that there would be nearly 4.5 billion users, with an annual growth rate of 3% [21,23]. The majority of email users are corporate companies and small businesses. For instance, 81% of small businesses rely on email as their primary customer acquisition channel, and 80% for retention [17].

As the use of emails has grown in popularity, it has also caught the attention of attackers who endanger security and privacy. One significant issue is related to data breaches, in which email servers are frequently targeted [45]. Attackers may publicly release the breached data or attempt to profit by selling or negotiating with the email service provider [19,43]. For example, the 'Collection #1' breach revealed 2.7 billion identification records with 773 million emails and made the data available for sale online [55]. Data breaches also threaten the economy significantly, with the average cost of a breach increasing by 15% from 2020 to 4.45 million USD in 2023 [28]. Another concern is the privacy of email content from the Service Provider (SP). Often, emails are processed without encryption by mail servers or are encrypted by the SP, requiring users to trust them completely. However, with the growing concern for individual privacy and the implementation of privacy laws like the EU General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) [53], many users hesitate to use email for communicating sensitive information.

The aforementioned concerns led to the emergence of service providers such as ProtonMail [56] and Tutanota [58], who developed solutions for private emails using End-to-End Encryption (E2EE) techniques. These techniques addressed privacy issues regarding the SP, while also allowing the users to perform search on emails. However, they pose other limitations, for example, only the user can perform the search [38,59]. Furthermore, solutions such as E2EE, which keep the email content hidden from the SP, may not be enough in some situations and require other options. To better illustrate these concerns, we will use a company's email system as an example and provide more information below.

Example Use Case—Company Email System. Consider PrivCorp, a company that wants to establish an email infrastructure for its employees while upholding individual data privacy. PrivCorp is concerned about the potential impact of data breaches, which have recently hit a number of enterprises. Furthermore, unlike some companies that monitor employee emails with or without their consent based on legal jurisdiction, PrivCorp is committed to implementing email monitoring in a privacy-preserving manner. In summary, PrivCorp's email infrastructure development goals include:

1. *Privacy from SP:* The email content[1] should be hidden from the SP. To achieve the desired goal, the company is willing to use multiple SPs, if needed.
2. *Data breach protection:* The email should remain private even if all but one of the SPs are compromised.
3. *Spam filtering:* The company should be able to analyse external emails and perform spam filtering in a privacy-preserving manner to respect the privacy of the email content.
4. *Unintended data leakage prevention:* The company should be able to monitor emails from its employees to the outside world in a privacy-preserving manner for accidental data leaks or other content that violates company policies such as defamation against the company, character assaults, and abusive content.

[1] Mostly the subject and content fields but not other meta data.

To address goals (1) and (2) above, a simple approach is to use E2EE methods, like PGP [5] and S/MIME [50], in which emails are encrypted and signed by the sender using public-key cryptography and decrypted and verified by the receiver. However, the strong privacy guarantees of E2EE make achieving goals (3) and (4) extremely difficult, as both require some sort of processing over the encrypted email content by an external entity.

While techniques like Searchable Symmetric Encryption (SSE) [54] enable encrypted data search, their utility is limited due to: 1) the need for email sender and receiver to manage keys for search, restricting search access to these parties, and 2) practical SSE methods having limitations in terms of information leakage, which can compromise the privacy of the encrypted data [24,31,42]. Additionally, SSE doesn't align with our use-case as it lacks support for external agent search, a requirement for goal (4).

To achieve the aforementioned goals simultaneously and efficiently, we combine Secure Multi-Party Computation (MPC) [7,22] and Private Information Retrieval (PIR) [12,16] techniques.

Overview of Our Solution. At a high level, the idea behind PrivMail is to use multiple email Service Providers (SPs) and secret sharing techniques to ensure that no individual SP sees the email content in the clear. Our idea was inspired by the observation that most people already have multiple email accounts for varied purposes such as personal and professional correspondence, with the average user having 1.75 email accounts [23,60].

In our approach, rather than encrypting emails with cryptographic keys, we secret share them (both subject and content) between multiple SPs using MPC techniques. For example, if PrivCorp uses two SPs, say `gmail.com` and `outlook.com`, then each employee with id `empid` is assigned two email addresses, e.g., `empid@gmail.com` and `empid@outlook.com`. The sender splits the email content into secret shares, sending each via regular email to one of two addresses. The employee retrieves and locally combines these shares to access the content. We provide additional optimizations ensuring that the total communication is the same as a single unencrypted email. With this approach, the users are only required to exchange email addresses except some simple splitting and reconstruction operations (cf. Sect. 3). In contrast to exchanging cryptographic keys or certificates needed with PGP and S/MIME [23,60], such email addresses can be easily shared by the users (e.g., by writing on business cards or websites).

Besides providing privacy from the SP through secret sharing, our approach allows private server-side processing/search using MPC. However, we cannot guarantee privacy in case of a government agency forcing access to all servers. In this scenario, the best solution is to select the servers that are from different jurisdictions such that at least one of the servers is very hard to compromize.

Our Contributions. We propose PrivMail, a privacy-preserving system for emails. PrivMail provides a secure way to transfer and store emails without requiring to use keys or certificates. Our main idea is to secret share emails between multiple (non-colluding) mail servers (e.g., Gmail, Outlook, etc.), thereby keeping the data on each server private. Our approach has the advantage that privately sending and receiving emails can directly run on the existing email infrastructure *without server-side modifications*. PrivMail offers resilience against data breaches since the attacker must breach all of the email service providers involved in order to obtain any useful information. Furthermore, it reduces the usability issues that have been observed for schemes such as PGP and S/MIME [23,60].

A key feature of PrivMail is its support for privacy-preserving server-side processing on secret shared emails. We propose privacy-preserving drop-in replacements for the standard Internet Message Access Protocol (IMAP) SEARCH and FETCH commands, allowing a search agent to securely and efficiently search for keyword(s) over secret shared emails and retrieve the results. Our scheme combines techniques from Secure Multi-Party Computation (MPC) and Private Information Retrieval (PIR) while avoiding leakages and the key management required by schemes like Searchable Symmetric Encryption (SSE) [24,31,42].

We also simulate Simple Mail Transfer Protocol (SMTP) servers and run extensive benchmarks on private keyword search over the Enron Email Dataset [34]. We are able to demonstrate practical performance, which can encourage real-world email service providers to incorporate PrivMail's private search functionalities into their existing feature set. To summarize:

1. We propose PrivMail, a privacy-preserving framework for emails that enhances usability and data breach resilience without needing keys or certificates.
2. PrivMail offers privacy-preserving server-side processing on secret shared emails, facilitating private searches and retrieval while avoiding key management issues and leakages. We also propose multiple efficient keyword search techniques, utilizing specific properties of email text and language.
3. We published an open-source prototype implementation of PrivMail[2] and demonstrate its practicality via benchmarks on private keyword searches on the Enron Email Dataset [34].

2 Preliminaries

The generic design of PrivMail allows to seamlessly use any MPC protocol for secure computation. However, in this work, we focus on the well-explored setting with two servers ($\mathsf{n} = 2$), and the scheme is explained using this setting in the majority of the sections. PrivMail comprises of four entities: the sender ($\mathcal{S}$) and receiver ($\mathcal{R}$) of an email, a collection of MPC servers, and a search agent ($\mathcal{A}$).

Threat Model. All entities in PrivMail are considered to be *semi-honest*. The two MPC servers are assumed to be *non-colluding*. This is justifiable given that the MPC servers in our case are well-established email service providers such as Gmail and Outlook. Because their reputation is at stake, these service providers have strong incentives to follow the protocol and not conspire with other service providers to leak their information. They could even operate in different countries

[2] https://encrypto.de/code/PrivMail.

with different legislations. Such setup of non-colluding servers have already been deployed in the real-world for services such as Firefox telemetry [8], privacy-preserving machine learning [29], cryptocurrency wallets [18,52], and COVID-19 exposure notification analytics [4]. The search agent $\mathcal{A}$ will be either the email sender or receiver, or a pre-consented third party (for instance, in the use-case mentioned in Sect. 1, the company PrivCorp's email filtering service) and is expected to semi-honestly follow the protocol specifications.

Notations. We use the following logical gates: XOR ($\oplus$), AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). Since XOR and NOT gates can be evaluated locally in Secret Sharing (SS)-based MPC, an OR gate can be realised at the cost (communication) of an AND gate as $\mathsf{a} \vee \mathsf{b} = \neg(\neg\mathsf{a} \wedge \neg\mathsf{b})$. Given a set of m bits $\mathsf{b}_1, \ldots, \mathsf{b}_m$, $\bigwedge_{i=1}^{m} \mathsf{b}_i = \mathsf{b}_1 \wedge \mathsf{b}_2 \wedge \cdots \wedge \mathsf{b}_m$ represents the cumulative AND and the other logic operators follow similarly. When performing boolean operations with a single bit b and a binary string $\mathsf{v} \in \{0,1\}^{\ell}$, we assume that the same bit b is used to perform the operation with each bit in the string v.

The values in PrivMail are secret shared between the two MPC servers via *boolean sharing* [22]: For a secret x, the i-th server, for $i \in \{1,2\}$, holds the share $\langle \mathsf{x} \rangle_i$ such that $\mathsf{x} = \langle \mathsf{x} \rangle_1 \oplus \langle \mathsf{x} \rangle_2$. We sometimes abbreviate the notation and use x_i instead of $\langle \mathsf{x} \rangle_i$ for the sake of brevity.

Existing Email Architecture. In a standard email communication, the sender $\mathcal{S}$ sends a message to the receiver $\mathcal{R}$ via a Mail User Agent (MUA) such as Thunderbird as follows. First, the MUA converts the message to email format and sends it to a Mail Transfer Agent (MTA). Upon receiving the email, the MTA transmits it to the receiver's Mail Delivery Agent (MDA) via the Simple Mail Transfer Protocol (SMTP). Finally, the MDA delivers the mail to $\mathcal{R}$'s mailbox. We leave out low-level details such as domain validation and refer to [33] for specifics.

On the receiver's side, $\mathcal{R}$ uses the Internet Message Access Protocol (IMAP) [13] to retrieve the email from the receiving server.[3] In more detail, the two functionalities IMAP *SEARCH* [13, §6.4.4] and *FETCH* [13, §6.4.5] are used to accomplish this. SEARCH provides a comprehensive search interface similar to the Structured Query Language (SQL). FETCH retrieves all the emails from the mail server returned by the SEARCH functionality.

3 PrivMail Architecture

In this section, we specify the architecture of PrivMail. In PrivMail, the sender and the receiver can choose their own independent set of Service Providers (SPs) whom they trust to not collude. W.l.o.g., we first explain our architecture assuming that both the sender and the receiver are registered with two distinct SPs

[3] The older Post Office Protocol (POP) downloads the email from the server and optionally deletes it from the server, but in contrast to IMAP provides no server-side search.

each. In concrete terms, each SP owns an email path that connects the sender's Mail Transfer Agent (MTA) to the receiver's Mail Delivery Agent (MDA).

At a high level, the sender $\mathcal{S}$ splits the original mail (`Subject, Body`) into two shares and sends them to receiver $\mathcal{R}$ via the two SPs, denoted by PrivMail Servers (PMS). The SPs are not required to perform any additional work to support our email transfer because the email shares are simply treated as *regular emails*, thus standard SMTP servers are sufficient for this. The email is retrieved by the receiver $\mathcal{R}$ by locally reconstructing the shares received from the SPs. Unlike an end-to-end encrypted email system (like PGP or S/MIME), this approach does not require either the sender or the receiver to handle any cryptographic keys or certificates. This also eliminates the challenges associated with implementing PGP or S/MIME in practice, where the users have to setup, manage and exchange keys and certificates [48–50].

Security Guarantees: PrivMail guarantees confidentiality, integrity and correctness. Confidentiality and correctness of PrivMail are ensured by the security of the underlying MPC protocols, while its integrity is assured as the users and the MPC servers are assumed to be semi-honest. To provide integrity of email transfer even with malicious servers, an honest sender can simply append a salted SHA256 hash of the email content together with the salt to each shared email. When the email is reconstructed, the receiver can verify that it matches both hashes/salts. This is secure as long as one of the servers is not corrupted and the sender is honest. In the case of search with malicious servers (which we leave as future work), the underlying MPC protocol is responsible for providing integrity, which is typically achieved using authentication tags for protocols in the dishonest-majority setting.

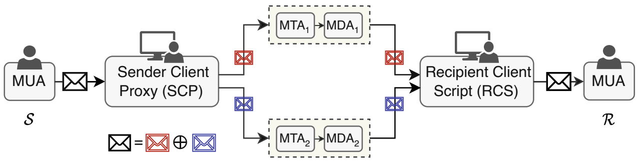


Fig. 1. PrivMail Communication: 2 Server Case.

Two Server Case: Now, we explain the communication phase for the two server case in more detail. As illustrated in Fig. 1, the first step is to split the email $\mathsf{E} = $ (`Subject, Body`) into secret shares as per the underlying MPC protocol. The secret sharing can be done either at $\mathcal{S}$'s Mail User Agent (MUA) using a custom plug-in for mail clients like Thunderbird or Outlook, or using a Sender Client Proxy (SCP) service between $\mathcal{S}$'s MUA and MTAs. In this work, we use the latter method as it is independent of the specific MUA. The email is shared as boolean shares, i.e., $\mathsf{E} = \mathsf{E}_1 \oplus \mathsf{E}_2$ with $\mathsf{E}_1 \in_R \{0,1\}^{|\mathsf{E}|}$.[4] Each of the email

[4] Later in Sect. 3.2 we describe an optimization to send a seed for a Pseudo Random Function (PRF) instead of the whole share E_1.

shares are now treated as *independent emails* and are sent using the sender's respective mail accounts using the regular email procedure. Similar to the SCP, we use a Recipient Client Script (RCS) at the receiver's end to reconstruct the original email from the shares.

3.1 Integration with the Existing Email Infrastructure

So far, we have assumed that each user sends and receives emails through a fixed set of distinct SPs. This assumption, however, may take some time to be adopted in practice, so we will now discuss how our scheme can be integrated with the existing email infrastructure.

The basic goal is to provide a private alternative on top of the existing email system such that the users can choose to communicate their emails either via PrivMail or via the existing email system. Let the sender $\mathcal{S}$ be registered to $\mathsf{N}_{\mathcal{S}}$ SPs and let $\mathsf{PMS}^{\mathcal{S}}$ be the set of these outgoing mail servers. Similarly, $\mathsf{PMS}^{\mathcal{R}}$ denotes the set of incoming email servers of receiver $\mathcal{R}$ of size $\mathsf{N}_{\mathcal{R}}$. Furthermore, we assume that both $\mathcal{S}$ and $\mathcal{R}$ have chosen their respective PMS servers in such a way that not all of the servers in their respective set will collude.

Naïve Approach: The naïve approach for integrating PrivMail to the existing email infrastructure is by splitting the mail at the sender's end into $n = \mathsf{N}_{\mathcal{S}} \cdot \mathsf{N}_{\mathcal{R}}$ shares using an n-out-of-n secret sharing scheme. Then each outgoing server $\mathsf{PMS}_i^{\mathcal{S}}$ will receive $\mathsf{N}_{\mathcal{R}}$ shares, one for each of the receiver's $\mathsf{N}_{\mathcal{R}}$ servers. These shares can then be sent to the corresponding receiver's servers as *regular mails.* This setting corresponds to having a path between each pair of $\mathsf{PMS}_i^{\mathcal{S}}$ and $\mathsf{PMS}_j^{\mathcal{R}}$.

Before looking deeper into the security of this approach, we define the term *secure path* in the context of email servers at the sender's and recipient's ends.

Definition 1. *A path connecting outgoing mail server* $\mathsf{PMS}_i^{\mathcal{S}} \in \mathsf{PMS}^{\mathcal{S}}$ *and incoming mail server* $\mathsf{PMS}_j^{\mathcal{R}} \in \mathsf{PMS}^{\mathcal{R}}$ *is said to be "secure" if neither* $\mathsf{PMS}_i^{\mathcal{S}}$ *nor* $\mathsf{PMS}_j^{\mathcal{R}}$ *colludes with the other servers in the set* $\mathsf{PMS}^{\mathcal{S}} \cup \mathsf{PMS}^{\mathcal{R}}$. *Here,* $\mathsf{N}_{\mathcal{S}}, \mathsf{N}_{\mathcal{R}} \geq 2$, *where* $\mathsf{N}_{\mathcal{S}} = |\mathsf{PMS}^{\mathcal{S}}|$ *and* $\mathsf{N}_{\mathcal{R}} = |\mathsf{PMS}^{\mathcal{R}}|$.

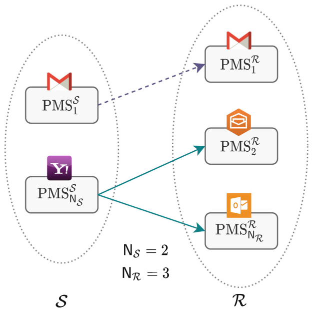
Privacy: Let $\mathsf{N}_{\mathsf{min}} = \min(\mathsf{N}_{\mathcal{S}}, \mathsf{N}_{\mathcal{R}})$. Considering the entire set of servers, i.e., $\mathsf{PMS}^{\mathcal{S}} \cup \mathsf{PMS}^{\mathcal{R}}$, the approach is secure as long as the adversary compromises no more than $\mathsf{N}_{\mathsf{min}} - 1$ servers. This is due to the fact that in this case, there will be at least one secure path (cf. Definition 1) from $\mathsf{PMS}^{\mathcal{S}}$ to $\mathsf{PMS}^{\mathcal{R}}$, and one share out of the $\mathsf{N}_{\mathcal{S}} \cdot \mathsf{N}_{\mathcal{R}}$ shares will be communicated via this path. Because the adversary has no knowledge of this share, the email content's privacy is guaranteed due to the privacy guarantee of the underlying n-out-of-n secret sharing scheme.

We observe that it is preferable to consider security at $\mathcal{S}$'s and $\mathcal{R}$'s ends separately rather than combining them, since in real-world scenarios, the servers in $\mathsf{PMS}^{\mathcal{S}}$ and $\mathsf{PMS}^{\mathcal{R}}$ may be from different legal jurisdictions. In such cases, the number of servers at the clients side does not ensure additional security. As a result of treating the servers separately, the scheme's security is preserved at the $\mathcal{S}$'s (or $\mathcal{R}$'s) end as long as not all servers in $\mathsf{PMS}^{\mathcal{S}}$ (or $\mathsf{PMS}^{\mathcal{R}}$) are compromised.

Note that the naïve approach requires a communication of $\mathsf{N}_{\mathcal{S}} \cdot \mathsf{N}_{\mathcal{R}}$ email shares. We now present an optimized approach that reduces the communication between $\mathsf{PMS}^{\mathcal{S}}$ and $\mathsf{PMS}^{\mathcal{R}}$ to $\max(\mathsf{N}_{\mathcal{S}}, \mathsf{N}_{\mathcal{R}})$ email shares.

Optimized Approach. In this method (Fig. 2), the sender splits the email into $\mathsf{N}_{\mathsf{max}} = \max(\mathsf{N}_{\mathcal{S}}, \mathsf{N}_{\mathcal{R}})$ shares as per the underlying MPC protocol. Without loss of generality, consider the case where $\mathsf{N}_{\mathcal{S}} < \mathsf{N}_{\mathcal{R}}$ and the other case follows similarly. Once the email shares are generated, SCP at $\mathcal{S}$'s end will compute a mapping from the servers in $\mathsf{PMS}^{\mathcal{S}}$ to $\mathsf{PMS}^{\mathcal{R}}$.

If there are common servers, i.e., $\mathsf{PMS}^{\mathcal{S}} \cap \mathsf{PMS}^{\mathcal{R}} \neq \emptyset$, then a mapping is formed between the corresponding servers for each server in the intersection (e.g., in Fig. 2 Gmail server $\mathsf{PMS}_1^{\mathcal{S}}$ is mapped to Gmail server $\mathsf{PMS}_1^{\mathcal{R}}$). The remaining servers are then assigned a random mapping so that each of the servers in $\mathsf{PMS}^{\mathcal{R}}$ receives exactly one email share. When $\mathsf{N}_{\mathcal{S}} \geq \mathsf{N}_{\mathcal{R}}$, the mapping is s.t. each server in $\mathsf{PMS}^{\mathcal{S}}$ is assigned exactly one email share, whereas servers at the receiver's end may receive multiple shares. More details on applying our email sharing in practice are provided in the full version [10, §4.1].



$\mathsf{PMS}_1^{\mathcal{S}}$
$\mathsf{PMS}_{\mathsf{N}_{\mathcal{S}}}^{\mathcal{S}}$
$\mathsf{PMS}_1^{\mathcal{R}}$
$\mathsf{PMS}_2^{\mathcal{R}}$
$\mathsf{PMS}_{\mathsf{N}_{\mathcal{R}}}^{\mathcal{R}}$
$\mathsf{N}_{\mathcal{S}} = 2$
$\mathsf{N}_{\mathcal{R}} = 3$
$\mathcal{S}$
$\mathcal{R}$

Fig. 2. Optimized Approach

Privacy: Similar to the naïve approach, the privacy of the optimized approach is ensured as long as there is at least one secure path between the servers at the $\mathcal{S}$'s and $\mathcal{R}$'s ends, i.e., as long as an adversary corrupts a maximum of $\mathsf{N}_{\mathsf{min}} - 1$ servers, this approach is secure.

3.2 Sharing Optimization Using PRF Keys

In the two approaches mentioned in Sect. 3.1, each share is of the same size as the original mail. Therefore, the total communication required for sending a mail of size $|\mathsf{E}|$ would be $\max(\mathsf{N}_{\mathcal{S}}, \mathsf{N}_{\mathcal{R}}) \cdot |\mathsf{E}|$ for the optimized approach.

To further optimize the communication and storage, we can adopt techniques similar to those used in multi-server PIR schemes [12,15,16], that also use n-out-of-n secret sharing to split the PIR query. In these works, the query is partitioned into several *chunks*. In line with this technique, we divide each mail E into n chunks of size $|\mathsf{E}|/\mathsf{n}$ each. Each chunk is then shared among the servers PMS_i using boolean sharing such that $\mathsf{chunk}_j = \mathsf{flip}_j \oplus_{i=n, i \neq j}^{n} \mathsf{rnd}_i^j$, where flip is a boolean share and $\mathsf{rnd}_i^j = \mathrm{PRG}(\mathsf{key}_i)[j]$, where key_i is a 128-bit symmetric key. Server i then receives the tuple $(\mathsf{flip}_i, \mathsf{key}_i)$ as its share of the mail which has size $|\mathsf{E}|/\mathsf{n} + 128$ *bits*. The mail can thus be reconstructed by concatenating all the chunks together, i.e., $\mathsf{chunk}_1 \| \mathsf{chunk}_2 \| \cdots \| \mathsf{chunk}_\mathsf{n}$. For the correctness of this sharing, we set the number of chunks $\mathsf{n} = \max(\mathsf{N}_{\mathcal{S}}, \mathsf{N}_{\mathcal{R}})$. This reduces the total communication of the optimized approach to $|\mathsf{E}| + \max(\mathsf{N}_{\mathcal{S}}, \mathsf{N}_{\mathcal{R}}) \cdot 128$ *bits*, which is about the size of the original email.

4 Private Queries Using MPC

The main advantage of PrivMail is that private server-side processing of the mails is easily possible because the emails are secret shared among the servers. This allows commonly used functions like keyword search to be implemented using various MPC techniques. Furthermore, keyword searches can be utilised for other functionalities, such as checking for data leakage in outgoing emails or detecting spam in incoming emails. Given the ubiquitous usage of keyword search in the existing email infrastructure, we present multiple private keyword search techniques and discuss optimizations and extensions. Note that the search agent $\mathcal{A}$ can be either of the email users, i.e., $\mathcal{S}$ or $\mathcal{R}$, or a third party with prior consent, e.g., a company mail service.

The discussion that follows assumes that the communication phase (cf. Sect. 3) has been completed and that the email content has been secret shared among the PrivMail servers (PMSs). Consider a mailbox containing p emails that are secret shared among n non-colluding servers. Given a keyword K, the private query phase proceeds using two sub-protocols: i) Private Search ($\mathcal{F}_{\mathsf{Search}}$) emulating IMAP SEARCH and ii) Private Fetch ($\mathcal{F}_{\mathsf{Fetch}}$) emulating IMAP FETCH, which are executed in the order detailed next.

1. The search agent $\mathcal{A}$ secret shares the keyword K among the n servers in accordance with the underlying MPC protocol.
2. Upon receiving the shares of K, the servers initiate the $\mathcal{F}_{\mathsf{Search}}$ functionality that enables $\mathcal{A}$ to obtain a list of indices that corresponds to emails containing the keyword. Concretely, $\mathcal{A}$ obtains a p-bit vector $\mathbf{H} \in \{0,1\}^{\mathsf{p}}$ with $\mathsf{H}[i] = 1$ if the i^{th} email contains K and 0 otherwise. We instantiate $\mathcal{F}_{\mathsf{Search}}$ using different techniques in Sect. 4.2.
3. $\mathcal{A}$ and the n servers jointly execute $\mathcal{F}_{\mathsf{Fetch}}$ (cf. Sect. 4.3) to privately fetch each email from the mailbox. We efficiently instantiate $\mathcal{F}_{\mathsf{Fetch}}$ using an extension of 2-server PIR to a secret shared database (cf. Sect. 4.3).

To summarize, $\mathcal{F}_{\mathsf{Search}}$ and $\mathcal{F}_{\mathsf{Fetch}}$ provide a privacy-preserving drop-in replacement for the standard Internet Message Access Protocol (IMAP) [13] functionalities SEARCH and FETCH respectively. Similar to IMAP SEARCH, PrivMail also supports combining multiple keyword searches for comprehensive filtering of the emails by privately applying boolean operations on the output vectors $\mathbf{H}$. In the following we concentrate on single keyword searches.

4.1 Private Search $\mathcal{F}_{\mathsf{Search}}$

Recall from Sect. 3 that our basic approach secret shares the email's subject and body fields in their original form. One disadvantage of this strategy is that the private search becomes case-sensitive. Working over the actual shares with MPC incurs additional computation and communication to provide a solution similar to the standard IMAP SEARCH, which is case-insensitive. A simple way to avoid this overhead is to let the sender $\mathcal{S}$ secret share a lowercase version of the email as well which doubles the size of the email. The $\mathcal{F}_{\mathsf{Search}}$ function will then be performed over these shares, and the actual shares will be used to reconstruct the original email (in $\mathcal{F}_{\mathsf{Fetch}}$). The search's efficiency can be further improved by employing a special encoding described next.

Special Encoding: We define our own special 6-bit character encoding without losing much information for standard emails in English text that use 7-bit ASCII character encoding, similar to the SixBit ASCII code by AIS [47]. The encoding space is sufficient for all the lowercase alphabets and numbers (0–9) along with 28 special characters.For this, we omit all ASCII control characters as well as a small set of special characters that are uncommon for a keyword. We call this 6-bit encoded email a *compact email* from now on. As shown later in Sect. 4.2, the bit length reduction due to this encoding also helps to improve the performance of the $\mathcal{F}_{\mathsf{Search}}$ protocols.

Computing Servers (CSs): Now that $\mathcal{F}_{\mathsf{Search}}$ operates over shares of the compact email and these operations are computationally intensive (as will be evident from the subsequent sections), the PMS servers can use powerful dedicated servers for this task. Henceforth, we refer to the servers for $\mathcal{F}_{\mathsf{Search}}$ as Computing Servers (CSs). Note that the PMS servers can be used as the CSs as well.

$\mathcal{F}_{\mathsf{Search}}$ *for the 2-Server Case:* As before, we resort to a simple setting of two servers and use a Secure Two-Party Computation (STPC) protocol for our computations, where the CSs enact the role of MPC servers for the private query phase. The functionality $\mathcal{F}_{\mathsf{Search}}$ takes the shares of the search keyword and the emails as input and returns the shares of the p-bit binary vector $\mathbf{H}$. Looking ahead, in the protocol, the shares of the keyword are generated by the search agent $\mathcal{A}$ and sent to the corresponding servers (the PRF optimization from Sect. 3.2 can be used here as well for long keywords). Furthermore, both shares of the result of $\mathcal{F}_{\mathsf{Search}}$ are given to $\mathcal{A}$, who reconstructs the result locally.

Remarks: When the search agent $\mathcal{A}$ is the email user ($\mathcal{S}$ or $\mathcal{R}$), it may be preferable to download the entire mailbox and perform keyword search locally over the cleartext. Though this naïve solution appears to be much cheaper for the CSs,, it may not always be an ideal solution from the perspective of the email user due to factors such as limited (mobile) bandwidth, local storage, battery usage, or cross-device accessibility.

4.2 Instantiating $\mathcal{F}_{\mathsf{Search}}$

In this section, we look at concrete instantiations of the $\mathcal{F}_{\mathsf{Search}}$ functionality. First, we discuss a generic instantiation, called Circuit-Based search, applicable to keywords of varying lengths and frequencies. Then, we present two optimizations: i) Bucketing-Based, and ii) Indexing-Based searches. The former is more efficient for single word keywords, while the latter is more efficient for keywords with higher frequency of appearance in the text. While primarily discussed within a two-server framework, these approaches can be generalized to multiple servers.

Circuit-Based Search. In this approach, the email is treated as a continuous string of characters and the keyword is then matched at each position of this string using an equality test circuit. To be precise, consider a text $W = w_1 \| \cdots \| w_t$ of length t and a keyword $\mathsf{K} = k_1 \| \cdots \| k_s$ of length s, with $s \leq t$. Let b denote the bit-length of characters (e.g., $b = 7$ for ASCII characters and $b = 6$ for compact emails, cf. Sect. 4.1). At a high level, the strategy is as follows: beginning with the i-th character position of W, a block of s characters denoted by $\widetilde{W_i}$ is derived and compared to the keyword K. We use an equality test functionality $\mathcal{F}_{\mathsf{EQ}}$ over ℓ-bit inputs to compare K and $\widetilde{W_i}$, defined as $\mathcal{F}^{\ell}_{\mathsf{EQ}}(x, y) = 1$ if $x = y$, and 0 otherwise. We use the EQ circuit of [35,36,51] to instantiate $\mathcal{F}_{\mathsf{EQ}}$ which is defined as follows:

$$\mathsf{EQ}^{\ell}(x, y) = \bigwedge_{i=1}^{\ell} \neg(x_i \oplus y_i). \tag{1}$$

There are $t - s + 1$ blocks $(\widetilde{W_i})$ with one EQ protocol executed per block. All of these executions are independent and can be carried-out in parallel. Once the results have been evaluated, a cumulative OR of the results can be used to find at least one matching block. To summarize, the search circuit SC is defined as

$$\mathsf{SC}(\underbrace{\mathsf{K}, W}_{\text{(Keyword, Text)}}) = \bigvee_{i=1}^{\overset{\text{\# blocks}}{t-s+1}} \mathsf{EQ}^{\underset{\text{bit length}}{sb}}(\mathsf{K}, \widetilde{W_i}). \tag{2}$$

Complexity: An instance of EQ over ℓ bit inputs require a total of $\ell - 1$ AND gates (cf. Eq. (1)) and has a multiplicative depth of $\lceil \log_2 \ell \rceil$ when evaluated as a tree. The SC consists of $t - s + 1$ such EQ circuits and additionally $t - s$ AND gates (as OR can be implemented via AND). Moreover, another $\lceil \log_2(t - s + 1) \rceil$ rounds are required to compute the final result. Hence, for $\ell = sb$ in our case, the SC circuit has a total of $(t + 1)sb - s^2b - 1$ ($\approx tsb$, when $t \gg s, b$) AND gates and a depth of $\lceil \log_2 sb \rceil + \lceil \log_2(t - s + 1) \rceil$. Note that the depth of the circuit can be further reduced at the expense of increased communication using multi-input AND gates [37,44].

The method described above assumes that the lengths of the text and the keyword are known to all Computing Servers (CSs). In our case, because the subject and body of an email are the text, hiding its length during computation is impractical for efficiency reasons. However, we can hide the length of the s-length keyword by padding it to a fixed length, which results in the following modifications to the above approach.

Length-Hiding Keyword Search. Given two $\ell_{\max}$-bit values x, y, let the functionality $\mathcal{F}_{\mathsf{LEQ}}$ be defined as $\mathcal{F}^{\ell_{\max}, \ell}_{\mathsf{LEQ}}(x, y) = 1$ if $x[i] = y[i]$ for $i \leq \ell$ and 0 otherwise. For a given $\ell \leq \ell_{\max}$, $\mathcal{F}_{\mathsf{LEQ}}$ returns 1 if the first ℓ-bits of x match with y and 0 otherwise. To hide the length $\ell = sb$ bits of keyword K, we use an additional

$\ell_{\max}$-bit *length mask*[5] of the form $\mathrm{M}_x = m_1 \| \cdots \| m_{\ell_{\max}} = \{1\}^{\ell} \| \{0\}^{\ell_{\max}-\ell}$. Given these values, we instantiate $\mathcal{F}_{\mathsf{LEQ}}$ by using a length hiding equality test circuit LEQ, which is defined as

$$\mathsf{LEQ}^{\ell_{\max}}(x, y, \mathrm{M}_x) = \bigwedge_{i=1}^{\ell} \neg((x_i \oplus y_i) \wedge m_i). \tag{3}$$

The logic of the LEQ circuit is similar to that of EQ (Eq. (1)) in that the bits of x, y are compared, but the result for the padded bits (last $\ell_{\max} - \ell$ bits) is discarded by ANDing them with the zero bits of the M_x value. This only adds a layer of parallel AND gates to the EQ circuit.

Optimizations to the Circuit-Based Search: We further optimize the Circuit-Based search using different properties of the mail text and keywords. If the keyword contains no spaces, i.e., is a single word, it is beneficial to "jump" over *spaces* in the target text to avoid unnecessary comparisons. However, checking for spaces in the Circuit-Based Search is difficult as the entire email is treated as a single string. Including the logic to check for spaces within the search circuit SC (cf. Eq. (2)) is costly. Therefore, for our optimizations, each mail is considered as a collection of distinct words rather than a continuous string of characters. This also gives us advantages when the mail contains repeated words, since now we can omit the duplicates from the search.

Bucketing-Based Search. To search on distinct words, we let the sender $\mathcal{S}$ secret share each *distinct* word in the email text individually in a randomized order. The words can also be padded to a fixed length to hide the original lengths.

Padding for length obfuscation should be done with caution, as it is a trade-off between efficiency and privacy. Padding every word to a large fixed length would yield maximum privacy. However, the efficiency may now be even worse than that of the Circuit-Based Search. On the other hand, the padding can be removed completely to maximize efficiency. As a compromise, we propose a *bucketing* technique that allows users to tailor the trade-off between efficiency and privacy to their specific needs.

A naïve way of instantiating a search given a list of secret shared words and a keyword would be to use a circuit-based Private Set Intersection (PSI) protocol [27,39,46]. However, linear complexity PSI protocols do exact but not sub-string matches, and thus cannot be used easily to add padding to the words for increased privacy. Therefore, we use our Circuit-Based Approach to search through the list of words in the email.

The idea behind bucketing is to select buckets for different ranges of character length and do the padding accordingly. For example, if we define a bucket for words with lengths ranging from 1 to 4 characters, each word of that range in the bucket is padded to 4 characters. Furthermore, the search keyword must be padded in accordance with the bucketing scheme. To facilitate substring matches,

[5] The agent $\mathcal{A}$ generates the shares of the mask and sends them with the keyword shares.

the search must be performed over the bucket defined for the keyword length as well as the buckets for longer words. As a result, the bucketing technique is more efficient for longer keywords as the shorter buckets can be ignored. In order to hide the actual length of the keyword, we must use the LEQ circuit, see Eq. (3) for the equality tests.

Indexing-Based Search. All search approaches described before were primarily concerned with searching over each email individually. Multiple emails in a mailbox, on the other hand, are likely to have many words in common. With this observation, we further optimize the Bucketing-Based Search for single-word search and build a *search index* for *all* emails in a mailbox.

Consider a mailbox containing p emails $\{\mathsf{E}_1, \ldots, \mathsf{E}_\mathsf{p}\}$ and let d be the number of distinct words in those emails. In the index table, each distinct word, denoted as $\mathcal{W}_i$, forms one row of the table and is associated with a p-bit string. The vector is referred to as *occurrence bit-string* and has the following format: $\mathbf{B}^{\mathcal{W}_i} = \mathsf{B}_1^{\mathcal{W}_i} \| \cdots \| \mathsf{B}_\mathsf{p}^{\mathcal{W}_i}$. Here, $\mathsf{B}_j^{\mathcal{W}_i} = 1$ denotes the presence of word $\mathcal{W}_i$ in the email E_j and 0 denotes its absence.The index table is then secret shared among the Computing Servers (CSs).

In contrast to the previous approaches where a bit value was returned for each email, in the indexing-based approach, a keyword is first searched against all the distinct words in the table and a d-bit string $\mathbf{u} = \mathsf{u}_1 \| \cdots \| \mathsf{u}_d$ is generated, where d is the number of distinct words. The final search result is nothing but a cumulative OR of all the occurrence bit-strings corresponding to the matched rows. Formally, the p-bit result of the search is

$$\text{Search Result} = \bigvee_{i=1}^{d} \mathsf{u}_i \cdot \mathbf{B}^{\mathcal{W}_i}, \tag{4}$$

where the OR operations over a bit vector are simply the operator applied to each bit position.

Given the secret shares of $\mathbf{u}$ among the CSs, one method for completing the above task is to involve the search agent $\mathcal{A}$. The vector $\mathbf{u}$ is reconstructed by $\mathcal{A}$ and the corresponding occurrence bit strings are obtained securely using $\mathcal{F}_{\mathsf{Fetch}}$ (as will be explained later in Sect. 4.3). The agent can then perform the OR operation locally to obtain the final result. Another method that avoids agent intervention is to let the CSs compute the expression in Eq. (4) using the underlying MPC protocol.

The index-based approach makes the search complexity *independent* of the number of mails, as the search depends only on the number of distinct words, except for final computation in Eq. (4), enhancing efficiency for large mail boxes.

Comparison: Each of the search techniques discussed above, i.e., Circuit-Based, Bucketing-Based and Indexing-Based, have their own pros and cons, depending on the search keyword length, number of mails in the mailbox and distinct keyword occurring. We compare the different methods in Table 2 in Sect. A.

4.3 Private Fetch $\mathcal{F}_{\mathsf{Fetch}}$

The $\mathcal{F}_{\mathsf{Fetch}}$ functionality, enables the search agent $\mathcal{A}$ to privately retrieve emails from the Computing Servers. $\mathcal{F}_{\mathsf{Fetch}}$, takes the p-bit string as input and returns the mails corresponding to the bit positions with value 1. Although the standard IMAP FETCH [13, §6.4.5] command can be used to retrieve the desired email shares from each server, in the long run, frequency analysis can reveal details of the search queries to the servers, particularly the keywords used [31]. To avoid this, we adapt multi-server Private Information Retrieval (PIR) to instantiate $\mathcal{F}_{\mathsf{Fetch}}$.

Consider a database $\mathfrak{D}$ containing p documents with the server CS$_i$ holding the share $\mathfrak{D}_i = \{\langle D_1\rangle_i, \ldots, \langle D_{\mathsf{p}}\rangle_i\}$, where each $\langle D\rangle_i = (\mathsf{flip}_i \| \mathsf{key}_i)$ for $i \in \{1,2\}$ (cf. Sect. 3.2). First, the client $\mathcal{A}$ samples two symmetric keys sk_1, sk_2 for a symmetric-key encryption scheme $\mathsf{Enc}()$ and sends key sk_i to server CSi. The next step, on a high level, is that each CSi will encrypt its share (corresponding to each document) with sk_i and send this to the other server. Let $\widetilde{\mathfrak{D}}_i$ denote the encrypted database share of CSi. The encrypted database is now defined as $\widetilde{\mathfrak{D}} = \widetilde{\mathfrak{D}}_1 \| \widetilde{\mathfrak{D}}_2$. Then the client $\mathcal{A}$ sends the shares of the selection bit vector $\mathbf{b}$ to each server and the servers compute $\mathbf{b}_i \cdot \widetilde{\mathfrak{D}}$ and send the result to $\mathcal{A}$. The client $\mathcal{A}$ then combines these results by XORing them and decrypts the result using the encryption keys to obtain the queried email.

5 Implementation and Benchmarks

This section describes our implementation of PrivMail and evaluate the performance of reconstruction and keyword search over secret shared emails. A detailed discussion and more benchmarks are provided in the full version [10].

5.1 Email Transfer

The implementation of PrivMail consists of several parts, which were discussed in Sect. 3. Our implementation of the Sender Client Proxy (SCP) runs in a Docker container [1] and works with any Mail User Agent (MUA) that allows the user to manually specify the outgoing SMTP server (i.e., basically any email client). We also implemented a simple Recipient Client Script (RCS) for email reconstruction at the receiver's end and use it for performance evaluation. We use the YAML data-serialization language [3] to store the emails in the filesystem.

Reconstruction Performance. The reconstruction of the shares consists of three steps: fetching of the email shares from the (IMAP) servers, pairing, and finally combining the shares. For our Recipient Client Script (RCS), the retrieval is handled using the `imaplib` module [2] and after each fetch, the email is stored in a dictionary, where the email's Unique Identifier (UID) is the key and the share is the value. In the second step, the dictionaries are combined together. Since the dictionaries are hash maps in Python and the keys are random looking identifiers, the lookup for each email is a constant time operation. The runtime to pair and combine 500 emails from two servers takes only around 0.235 seconds, giving us a throughput of 2,127 emails per second on a regular laptop using Intel Core i7-8565U. The runtime of the fetching step depends on the IMAP server capacity, geographic location, and network setup, but is likely to be dominant (throughput was only 20 emails per second in our experiments). Thus the overhead introduced by PrivMail compared to fetching and viewing regular emails is negligible.

5.2 Email Search

We implement the private queries described in Sect. 4 using the mixed-protocol Secure Multi-Party Computation (MPC) framework MOTION [9]. We use the boolean GMW protocol [14,22] between two parties for our performance evaluation, but our implementation can also be used in the N-party setting with full threshold. Our code is publicly available under the MIT License[6].

Benchmark Settings for Search. We tested all three approaches from Sect. 4.2 for private search (Circuit-Based, Bucketing-Based, and Index-Based) on a real-world dataset with varying parameters, using our special encoding from Sect. 4.1. For the Bucketing-Based approach, we chose to create four buckets, each of size 5 characters, i.e., buckets for words with (1–5, 6–10, 11–15, 16–20) characters. All words that are more than 20 characters long are ignored. We instantiate the MPC protocols with computational security parameter 128 and statistical security parameter 40.

We run experiments against subsets of the publicly available Enron Email Dataset [34], which contains over 500,000 emails, with each email containing on average 1,607 characters and 237 words. On examining the distribution of distinct words in this database, we conclude that the bucket size distribution for our chosen buckets are $(18.8\%, 50.6\%, 21.7\%, 8.9\%)$. This implies that, on average, more than half of the words are between 6 and 10 characters long. Our benchmark subsets are drawn from Kenneth Lay's inbox.

The benchmarks are run on two dedicated simulation machines, each with an Intel Core i9–7960X (16 physical cores @ 2.8 GHz) processor and 8×16 GB DDR4 RAM. We simulate a Wide Area Network (WAN) setting with bandwidth limited to 100 Mbit/s, and Round-Trip Time (RTT) of 100 ms. To ensure consistency, we iterate each simulation 5 times and compute the mean for the final result.

Search Performance. We implement and compare our three search methods from Sect. 4.2. Since the Bucketing and Indexing-Based approaches already provide keyword length hiding, we use the Circuit-Based search's length hiding variant for a fair comparison. The Circuit-Based approach thus requires an additional layer of AND gates over the keyword length non-hiding variant.

Table 1 compares the performance of our three search methods in the online phase across various keyword lengths. Except for keyword length $s = 3$, we find that the Indexing-Based Approach outperforms the other two approaches in all cases. This is justified because the search domain for both the bucketing

[6] https://encrypto.de/code/PrivMail.

Table 1. Evaluation of online phase of keyword length-hiding search methods: circuit, bucketing and indexing. Best results are in bold.

Keyword Length s	Method	# Emails = 100		# Emails = 200	
		Time (s)	Comm. (MiB)	Time (s)	Comm. (MiB)
3	Circuit	9.63	**2.61**	18.19	**5.80**
	Bucketing	9.15	16.28	14.07	35.76
	Indexing	**3.57**	6.41	**6.58**	11.22
8	Circuit	13.68	4.62	25.96	10.41
	Bucketing	7.51	7.09	12.98	15.91
	Indexing	**3.19**	**3.73**	**5.22**	**6.97**
13	Circuit	13.78	6.59	23.73	14.93
	Bucketing	4.59	1.27	10.09	2.91
	Indexing	**2.48**	**0.63**	**4.26**	**1.48**
18	Circuit	15.38	8.58	27.51	19.47
	Bucketing	4.19	0.16	8.97	0.45
	Indexing	**2.68**	**0.06**	**3.96**	**0.25**

and Indexing-Based Approaches shrinks for long keywords due to the omission of buckets for short keywords. We also see that the Indexing-Based Approach improves by $\approx 2\times$ over the Bucketing-Based Approach in terms of both runtime and communication. Furthermore, the Indexing-Based method is expected to be more and more efficient compared to the other methods for a larger number of emails, since it eliminates duplicate words across the entire email set. We remark that in settings where the number of target documents is large and the document format is more regular, our indexing-based search can be orders of magnitude more efficient than bucketing-based search.

A detailed benchmark of the search approaches from Sect. 4.2, along with the details of the Thunderbird plugin, is provided in the full version [10, §6.2].

6 Related Work

A line of work that is very closely related to the secure keyword search in Sect. 4 is Secure Pattern Matching (SPM). SPM [25,57,61,62] entails a server with text $x \in \Sigma^n$ (over some alphabet Σ) and a receiver with pattern $p \in \Sigma^m$ with $m \leq n$. Without revealing additional information, the receiver learns where the pattern occurs as a substring of the server's text. SPM is a highly researched field with applications in various areas such as database search [11,20], network security [40], and DNA analysis [41]. Circuit-based approaches for pattern matching techniques have been proposed in [30,32]. These circuits are designed primarily for genomic computing and DNA matching, thus they aren't directly applicable in our context of keyword search (cf. Sect. 4.2). Later works such as [6,26,62] presented techniques based on homomorphic encryption. However, in these works, one of the parties involved owns the keyword while another (or a group of parties) owns the database, so they are not directly applicable to our case. In Sect. 4.2, we create custom circuits for our use cases in PrivMail.

We give a detailed discussion of related works on topics such as E2EE, SSE, MPC, PSI and PIR in the full version [10, §2].

Acknowledgements. This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230.

A Comparison of the Different Search Techniques

Each of the search techniques discussed in Sect. 4, i.e., circuit-based, bucketing-based, and indexing-based, have their own pros and cons depending on the keywords being searched. The user or the email client can therefore choose the most beneficial technique according to their requirements. In Table 2, we give a comparison between the different techniques for various use-cases, highlighting the most efficient techniques for each use-case.

Table 2. Efficiency comparison of the different search techniques for different types of keywords. The most efficient technique is marked in bold.

	Circuit	Bucket	Index
Longer Keywords	More efficient for longer keywords than for smaller	**Significantly more efficient as smaller buckets will be completely skipped**	Efficient only if the keyword is frequent in the text
Higher Frequency Keywords	Efficiency remains the same as for any keyword	Efficient if the keyword is of longer length	**Very efficient for frequent words in the text**
Partial Matches	**Very efficient as the text is considered as a continuous string of characters**	Would incur higher cost to implement	Would incur higher cost to implement
Special Words	**Efficient, as all words are treated with same priority**	**Efficient if the word is of medium to long length**	Not very efficient as the word won't be frequent in the text

Performance of Circuit-Based Search. Table 3 summarizes our benchmarks for search across four different keyword lengths, $s \in \{3, 8, 13, 18\}$ (corresponding to the average of our bucket sizes), on email sets of sizes 100 and 200. The total computation and communication overheads grow proportionally to the keyword length and number of emails in the sets as the search circuit size grows.

Table 3. Evaluation of circuit-based search (Sect. 4.2). Runtime (Time) is in seconds and communication (Comm.) between the servers in mebibytes (MiB).

Keyword Length s	Phase	# Emails = 100		# Emails = 200	
		Time (s)	Comm. (MiB)	Time (s)	Comm. (MiB)
3	Online	4.80	1.22	11.67	2.72
	Total	12.19	63.42	33.45	145.28
8	Online	5.20	3.12	10.15	7.03
	Total	20.96	174.25	48.65	399.61
13	Online	5.12	5.03	11.35	11.33
	Total	25.33	284.15	60.43	652.05
18	Online	4.31	6.89	9.40	15.52
	Total	32.47	393.07	73.74	902.54

We parallelize each equality test circuit (see Eq. (1)) with Single Instruction, Multiple Data (SIMD) operations, which results in an almost linear total runtime with respect to the keyword length. The minor difference in online runtime is caused by runtime fluctuations in our WAN simulation and can be evened out with additional iterations. The remaining cumulative OR in Eq. (2) dominates the online runtime, giving a nearly constant runtime.

References

1. Docker Container. https://www.docker.com
2. imaplib. https://docs.python.org/3/library/imaplib.html#imaplib.IMAP4.fetch
3. YAML Data Serialization Language. https://yaml.org
4. Apple and Google: Exposure Notification Privacy-Preserving Analytics (ENPA) white paper (2021)
5. Atkins, D., Stallings, W., Zimmermann, P.: PGP Message Exchange Formats. RFC 1991 (1996). https://www.rfc-editor.org/rfc/rfc1991.txt
6. Baron, J., Defrawy, K.E., Minkovich, K., Ostrovsky, R., Tressler, E.: 5PM: secure pattern matching. In: SCN (2012)
7. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC (1988)
8. Blog, M.S.: Next steps in privacy-preserving Telemetry with Prio (2019). https://blog.mozilla.org/security/2019/06/06/next-steps-in-privacy-preserving-telemetry-with-prio/

9. Braun, L., Demmler, D., Schneider, T., Tkachenko, O.: MOTION - a framework for mixed-protocol multi-party computation. ACM TOPS **25**(2), 1–35 (2021)
10. Chandran, G.R., Nieminen, R., Schneider, T., Suresh, A.: PrivMail: a privacy-preserving framework for secure emails (full version). ePrint Archive, Paper 2023/1294 (2023). https://encrypto.de/code/PrivMail
11. Chase, M., Shen, E.: Substring-searchable symmetric encryption. PoPETs (2015)
12. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS (1995)
13. Crispin, M.: Internet Message Access Protocol - Version 4rev1. RFC 3501 (2003). https://rfc-editor.org/rfc/rfc3501.txt
14. Demmler, D., Schneider, T., Zohner, M.: ABY – a framework for efficient mixed-protocol secure two-party computation. In: NDSS (2015)
15. Demmler, D., Herzberg, A., Schneider, T.: RAID-PIR: practical multi-server PIR. In: CCSW (2014)
16. Demmler, D., Holz, M., Schneider, T.: OnionPIR: effective protection of sensitive metadata in online communication networks. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017. LNCS, vol. 10355, pp. 599–619. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_30
17. ETail Emarsys WBR SMB Report: Adapting to the pace of omnichannel commerce (2016). https://emarsys.com/learn/white-papers/adapting-to-the-pace-of-omnichannel-commerce/
18. Fireblocks: MPC Wallet as a Service Technology (2022). https://www.fireblocks.com/platforms/mpc-wallet/
19. Franceschi-Bicchierai, L.: T-Mobile says hacker accessed personal data of 37 million customers (2023). https://techcrunch.com/2023/01/19/t-mobile-data-breach/
20. Gennaro, R., Hazay, C., Sorensen, J.S.: Text search protocols with simulation based security. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 332–350. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_20
21. Gilbert, N.: Number of Email Users Worldwide 2022/2023: Demographics & Predictions (2022). https://financesonline.com/number-of-email-users/
22. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game: a completeness theorem for protocols with honest majority. In: STOC (1987)
23. The Radicati Group, Inc.: Email Statistics Report, 2019–2023 (2018). https://www.radicati.com/wp/wp-content/uploads/2018/12/Email-Statistics-Report-2019-2023-Executive-Summary.pdf
24. Gui, Z., Paterson, K.G., Patranabis, S.: Rethinking searchable symmetric encryption. In: S&P (2023)
25. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. J. Cryptol. **23**, 422–456 (2010). https://doi.org/10.1007/s00145-008-9034-x
26. Hazay, C., Toft, T.: Computationally secure pattern matching in the presence of malicious adversaries. J. Cryptol. **27**, 358–395 (2014). https://doi.org/10.1007/s00145-013-9147-8
27. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS (2012)
28. IBM Security: Cost of a Data Breach Report 2023 (2023). https://www.ibm.com/reports/data-breach
29. Inpher: XOR Secret Computing Engine (2022). https://inpher.io/xor-secret-computing/

30. Jha, S., Kruger, L., Shmatikov, V.: Towards practical privacy for genomic computation. In: S&P(2008)
31. Kamara, S., Kati, A., Moataz, T., Schneider, T., Treiber, A., Yonli, M.: SoK: cryptanalysis of encrypted search with LEAKER - a framework for LEakage AttacK Evaluation on Real-world data. In: EuroS&P (2022)
32. Katz, J., Malka, L.: Secure text processing with applications to private DNA matching. In: CCS(2010)
33. Klensin, D.J.C.: Simple Mail Transfer Protocol. RFC 5321 (2008). https://rfc-editor.org/rfc/rfc5321.txt
34. Klimt, B., Yang, Y.: The enron corpus: a new dataset for email classification research. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 217–226. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30115-8_22. https://www.cs.cmu.edu/~./enron/
35. Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 1–20. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_1
36. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_40
37. Koti, N., Patra, A., Rachuri, R., Suresh, A.: Tetrad: actively secure 4PC for secure training and inference. In: NDSS (2022)
38. Martinoli, M.: Behind the scenes of ProtonMail's message content search (2022). https://proton.me/blog/engineering-message-content-search
39. Mohassel, P., Rindal, P., Rosulek, M.: Fast database joins and PSI for secret shared data. In: CCS (2020)
40. Namjoshi, K.S., Narlikar, G.J.: Robust and fast pattern matching for intrusion detection. In: INFOCOM (2010)
41. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: SCiFI - a system for secure face identification. In: S&P(2010)
42. Oya, S., Kerschbaum, F.: Hiding the access pattern is not enough: exploiting search pattern leakage in searchable encryption. In: USENIX Security (2021)
43. Page, C., Whittaker, Z.: It's All in the (Lack of) Details: 2022's badly handled data breaches (2022). https://techcrunch.com/2022/12/27/badly-handled-data-breaches-2022/
44. Patra, A., Schneider, T., Suresh, A., Yalame, H.: ABY2.0: improved mixed-protocol secure two-party computation. In: USENIX Security (2021)
45. Perlroth, N.: Yahoo Says Hackers Stole Data on 500 Million Users in 2014 (2016). https://www.nytimes.com/2016/09/23/technology/yahoo-hackers.html
46. Pinkas, B., Schneider, T., Tkachenko, O., Yanai, A.: Efficient circuit-based PSI with linear communication. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 122–153. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_5
47. Raymond, E.S.: AIS Payload Data Types (2017). https://gpsd.gitlab.io/gpsd/AIVDM.html
48. Ruoti, S., et al.: A usability study of four secure email tools using paired participants. ACM TOPS **22**(2), 1–33 (2019)
49. Ruoti, S., Andersen, J., Zappala, D., Seamons, K.E.: Why Johnny still, still can't encrypt: evaluating the usability of a modern PGP client. CoRR 1510.08555 (2015)

50. Schaad, J., Ramsdell, B.C., Turner, S.: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. RFC 8551 (2019). https://rfc-editor.org/rfc/rfc8551.txt
51. Schneider, T., Zohner, M.: GMW vs. Yao? Efficient secure two-party computation with low depth circuits. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 275–292. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_23
52. Sepior: Advanced MPC Wallet$^{\mathrm{TM}}$ (2022). https://sepior.com/products/advanced-mpc-wallet/
53. Simmons, D.: 17 Countries with GDPR-Like Data Privacy Laws (2022). https://insights.comforte.com/countries-with-gdpr-like-data-privacy-laws
54. Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: S&P (2000)
55. Song, V.: Mother of All Breaches Exposes 773 Million Emails, 21 Million Passwords (2019). https://gizmodo.com/mother-of-all-breaches-exposes-773-million-emails-21-m-1831833456
56. Proton Technologies: ProtonMail Security Features and Infrastructure (2016). https://protonmail.com/docs/business-whitepaper.pdf
57. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.U.: Privacy preserving error resilient DNA searching through oblivious automata. In: CCS (2007)
58. Tutanota: Secure email made for you. https://tutanota.com/security
59. Tutanota: Searching encrypted data is now possible with Tutanota's innovative feature (2017). https://tutanota.com/blog/posts/first-search-encrypted-data
60. Watson, T.: The number of email addresses people use [survey data] (2019). https://www.zettasphere.com/how-many-email-addresses-people-typically-use
61. Wei, X., Zhao, M., Xu, Q.: Efficient and secure outsourced approximate pattern matching protocol. Soft. Comput. **22**, 1175–1187 (2018). https://doi.org/10.1007/s00500-017-2560-4
62. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Privacy-preserving wildcards pattern matching using symmetric somewhat homomorphic encryption. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 338–353. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08344-5_22