Harry Halpin (Greenhost/LEAP))
Kali Kaneko (Greenhost/LEAP)
Ruben Pollan (Greenhost/LEAP)
Elijah Sparrow (Greenhost/LEAP))
Mooness Davarian (Greenhost/LEAP)
Raf Degens (Medialaan/Mobile Vikings)
Tariq Elahi (KUL)
George Danezis (UCL)

Applying Mix Nets to Email Document

Deliverable D 7.1

June 8, 2016
Panoramix Project, # 653497, Horizon 2020
http://www.panoramix-project.eu

Contents

Chapter 1

Introduction

This report presents the use-cases and requirements, based on empirical data, of the mix networking infrastructure, with a focus on usability, a mathematical analysis of anonymity sets, and threat models. In detail, chapters are organized as the following:

1. **Introduction**: Outline of the deliverable
2. **Use-cases**: Initial user groups for privacy-preserving email
3. **Email Systems**: An overview of modern email systems, providing the necessary background to understand the rest of the deliverable.
4. **Threat Models and Requirements**: The threats to user privacy that mix-nets can address, with a focus first on local and then on global passive adversaries, as well as the security and privacy requirements that we want PANORAMIX to achieve.
5. **LEAP Software**: A comprehensive discussion of the LEAP email system to provide end-to-end encryption that will serve as the base open-source codebase for deploying PANORAMIX mix networking solutions.
6. **End-user Usability**: How the mix-net can be integrated into the existing user interface of the email client, including the provisioning of different levels of privacy in an easy-to-understand manner across the different operating systems supported by the email client, including the simple "red/green" light when mix networking is enabled.
7. **System Administrator Usability**: An outline of what advanced features would allow the system administrators and users to customize their levels of privacy of the mix network.
8. **Empirical Mix-net Parameters**: What data was gathered and how to analyze data from existing LEAP-enabled servers in order to determine the timing parameters and other parameters (such as size parameters) of real email traffic needed to adopt mix networking for everyday email usage so that the email arrives within a reasonable time frame while preserving user privacy.
9. **Anonymity Analysis**: To determine how the global privacy of the system, quantified in terms of anonymity sets, is affected by the number of users, the number of servers, and other local parameters.

Chapter 2

Use-cases

The overall use-case for privacy-preserving email would be to provide an eemail solution that can be customized towards deployment with different types of organizations or stakeholders. The "Unique Selling Proposition" of the provided solution would be an "anonymous and secure email app." The following potential use cases and market segments can be identified are given below:

2.1 Companies and Government Use-case

Email for "Executive-Level" sensitive information sharing for Companies and Governments *(Business to Business Application)*: A privacy-preserving email system can easily be adapted towards a particular companyâĂŹs branding in order to be considered as the email system that can be used to transfer sensitive company internal information that is considered to be 'for your eyes only' without leaking the information to espionage from malicious third-parties (such as other companies) or insider threats (such as unauthorized employees). Both sender and receiver will need to have PANORAMIX-enabled email, and a company can control the distribution of accounts on this system although they may need to share sensitive email with third-parties that also have privacy-preserving email (third-party lawyers, accountants, etc.). This email system can be considered as being an extra-secure email for CEO-level individuals with access to the highest level of security and company information. As executives are very used to email, sharing a particular type of sensible information can be done easily with an email based application. Governments in Europe who are worried about national-level actors, particularly those with the ability to do large-scale surveillance such as the NSA or the Chinese or Russian governments, would also want a secure and privacy-preserving email for sensitive government information. There are real-world examples of companies possibly having business deals ruined by using ordinary off-the-shelf webmail systems internally. For example, it is rumored that a possible business take-over of Twitter by Gmail was ruined by the fact that Twitter used Gmail for its internal negotiations, including when attempting to hide information from Google itself.

The market advantage of PANORAMIX would be that the email would be delivered securely and anonymously from third-parties, which can be considered as a great asset for both the organization as a whole as well as the stakeholders involved. Both companies and governments would be interested in the self-hosting aspects of the email. Currently, there are a number of corporate applications such as Silent Circle aimed at this market.[1] However, while Silent Circle originally promised compatibility with email, it failed ultimately to integrate encrypted email and so ended up with their own custom messaging applications. A number of companies provide S/MIME based email for encryption such a Symantec Encryption Management Server (SEMS), but few are compatible with communicating outside strictly defined S/MIME systems that are too hierarchical and difficult to set-up for today's business environments. Furthermore, very few of these options are both self-hosted and open-source, and so would likely be untrusted by government actors, as companies can easily be subverted by the government in the area they are located in, as demonstrated by the use of "National Security Letters" in the United States of America.

[1] *https://silentcircle.com*

2.2 Journalist Use-case

Email for whistle-blowers and other sources for the media and press *(Business to Consumer Application)*: Whistle-blowers frequently face reprisal, sometimes at the hands of the organization or group which they have accused, sometimes from related organizations, and sometimes even from the law if there are not adequate legal protections for whistle-blowers. While questions about the legitimacy of whistleblowing, the moral responsibility of whistleblowing, and the appraisal of the institutions of whistleblowing are part of the field of political ethics, beyond Edward Snowden we are seeing the rise, with phenomena such as the "Panama Papers", of an increased use of anonymous sources by the media. These different anonymous sources that journalists have within governmental, juridical or criminal environments, to obtain inside information or tips about a particular event, would motivate the usage and purchase of a privacy-preserving email application as the media is dependent on the trust and security of their sources for breaking stories, which fuels their own business model. Yet we find that sources such as Snowden found it nearly impossible to communicate to journalists such as Greenwald, resulting in Snowden even making a "video" instructing Greenwald how to use GPG and eventually sending him the software via mail. We cannot expect most whistle-blowers and other sources to be as patient or as technically skilled, and so an easy to use email solution is needed.

The market advantage of a "whistle-blower email system" would be to provide the highest level of crucial anonymity for the source while allowing them to communicate with a journalist using normal email. Potential customers thus include media institutions and freelance journalists. We have already seen adoption of similar software such as SecureDrop[2] by major news outlets such as Forbes, The Guardian, The New Yorker, The Intercept, and The Washington Post. However, SecureDrop only allows communication via uploading of documents. There seems to be a market gap as there is no way for a source to simply email a journalist, or vice-versa.

2.3 Activists

Email for human rights defenders and other high-risk activists *(Consumer to Consumer Application)* Although many users are comfortable using Gmail, Yahoo and other off-the-shelf webmail solutions, there are certain high-risk activities that even ordinary users believe should not be done using these email providers. For example, many activists are concerned that their email is being monitored. While Gmail would be sufficient for some uses, Google is ultimately incorporated in the United States and so subject to US jurisdiction, which includes secretive "National Security Letters" to demand user information as well as normal jurisdictional and intelligence orders in the USA. This is a problem insofar as some activists have been targeted by the United States government, particularly if they are American activists such as those involved in protest groups such as "Black Lives Matter" that have been targeted by the FBI or environmental activists such as those involved in popular civil disobedience to mitigate global climate change. While many American high-risk activists use systems such as Tor to attempt to get their Internet traffic outside of the United States, they often then re-connect to services such as Gmail or Yahoo that are hosted within the United States, making the "round trip" to Europe somewhat pointless as their email can simply be directly and involuntarily seized due to an out-of-control government investigation or given voluntarily upon a request to their email provider. The same general problem holds true not just of activists in the United States, but of any activists whose email may be under threat due to government repression, such as in China, Brazil, or Tunisia.

The market advantage of privacy-preserving email is that currently no reputable privacy-preserving email system exists that is trusted by activists. Although many human rights defenders use services such as *Greenhost* and *Riseup.net*, these have problems that need to be addressed, as otherwise human rights defenders may move to providers such as Gmail even at the cost of their security and privacy. With upgrades to Gmail such as certificate-pinning and forcing TLS, it is unclear if these "human rights" email providers really provide more or better technical security protection than Gmail. However, while activists would like to use end-to-end encryption such as PGP with email, they find it too difficult to use. Activists tend to be aware of metadata analysis, and are concerned that advertised "secure" encrypted email servers such as *Protonmail* still reveal their metadata when communicating. Thus, some activists prefer to use a single server to communicate, but email services are known by the activists to be centralized and so possibly under seizure threat. Thus, the ability of PANORAMIX to provide a mix networking backbone would offer considerable

[2] *http://securedrop.org*

advantage, allowing activists to use different servers and preserve both their security and privacy, offering valuable "metadata protection" not offered by Gmail. This sort of consumer market is also likely not limited to human rights defenders, but also to a general purpose "private email" for individuals engaged in legal but risky activity who prefer to stay anonymous towards the outside world due to the risk involved. The consumer market is very creative, and thus we imagine that email that preserves privacy will be used in new and creative ways by end-users.

Chapter 3

Email Systems

Email relies on an open and federated protocol, where service providers relay messages on behalf of their users using SMTP (Extended Simple Mail Transfer Protocol).[1] These service providers form a *federation* such that each delivers mail on behalf of their users to each other. Furthermore, the federation is *open* such that mail can be delivered anytime. This is in contrast to a peer-to-peer model, where messages are delivered more directly from user to user or a centralized model where all delivery is handled by a single organization. To some degree, currently distinctions are institutional more than technical: In practice, large centralized infrastructures, such as Facebook, are internally organized in a federated manner. Likewise, peer-to-peer networks are rarely designed so that one user directly contacts the IP address of another user without going through another "hop" in an IP address.

In email, there are many **Mail Exchangers (MX)**. In detail, there is a **Mail User Agent (MUA)**, which hosts the mail on the user's client system. Then there is a **Mail Transfer Agent (MTA)** that delivers the mail in transit. Lastly, there is a **Mail Delivery Agent** that stores the delivered email until the user can retrieve it using a protocol such as IMAP[2] or POP.[3] MX records on the DNS identify the preferred servers for MX per domain name, so that a given domain name may have multiple MTAs.

The typical email exchange between two human end-users Alice and Bob can be thought of in the following sequence: $Alice \rightarrow MUA^1 \rightarrow MTA^1 \rightarrow ... \rightarrow MTA^2 \rightarrow MDA^2 \rightarrow MUA^2 \rightarrow Bob$

We assume for the purpose of our example that Alice's email is *alice@alice.org* and Bob is *bob@bob.org*. The steps are defined below:

1. ***Alice***: human end-user wants to communicate to Bob
2. ***Alice's MUA***: Alice's personal device, using a client such as Thunderbird or Outlook for the MUA function.
3. ***alice.org's MTA***: *alice.org* MX server running software such as Postfix, Mailman, Sendmail or Microsoft SMTP server. The server queries the MX record of *bob.org* in order to determine what MTA to use.
4. ***Multiple MTAs***: Between *alice.org* and *bob.org* there are potentially many intermediary MTAs, typically belonging to either *alice.org* or *bob.org* but also potentially belonging to third parties.
5. ***bob.org's MTA***: *bob.org* MX server running software such as Postfix, Mailman, Sendmail or Microsoft SMTP server.
6. ***bob.org's MDA***: *bob.org* with the MDA function being software such as Fetchmail or Procmail.
7. ***Bob's MUA***: Bob's personal device also running an MUA such as Thunderbird or Outlook.
8. ***Bob***: human recipient of the mail.

[1] *https://tools.ietf.org/html/rfc5321*
[2] *https://tools.ietf.org/html/rfc5321*
[3] *https://tools.ietf.org/html/rfc1939*

It is not enforced by protocol that a sender's MUA must relay outgoing mail through their own provider's MX. However, in practice, it is typically impractical to bypass the sender's provider as the ports that MX servers use to relay mail to one another are typically blocked for any end-user device and the recipient MX will be very skeptical of an incoming message that comes from a totally unknown source. This latter point could be a significant problem with mix networking and email, and so special work will have to be done to authorize email from a mix network to a recipient's MX server.

Chapter 4

Threat Models and Requirements

In this chapter, we will outline local, global, active and passive threat models, in particular the passive global adversary that PANORAMIX needs to counter. Then we will go through the security and privacy requirements that are necessary in order to defeat these threats. Lastly, we will walk through some of the threats in more detail, and inspect current mitigations as used by email systems in practice. Finally, we will outline how PANORAMIX can integrate and improve these enisting mitigations to counter some of the new threats caused by abusive users in mix networking systems for messaging.

4.1 Threat Models

Despite the degree to which SMTP and the email infrastructure has been able to slowly evolve over time, there are still many persistent threats that an email service provider faces. In this section, we list a subset of them. While these threats are not unique to mix networking, they are exacerbated by mix networking due to the anonymity provided by mix nets. We will list each of the threats, how they are currently dealt with by production email systems, and include analysis of how mix networking affects the threat.

In general, we consider the goal of the attacker to be to read the email messages of one or more clients. The attacker, if unable to read the email message, has another goal of determining which users are sending email messages to each other. The attacker may also have a goal to stop the sending of email messages. There are *active* attackers that can stop network traffic, inject network traffic, and attempt to gain access to servers and clients. There are *passive adversaries* that simply copy messages in transit on the network without altering the content or the delivery of the message. Attackers can be *local* attackers, and so capable of attacking only a subset of the entire email network such as particular clients, servers, and the network routes between them. Generally, active attacks are localized to some extent. There are also *global* attackers that can monitor the entire email network itself. There is of course vast scope between many local attackers in practice. For example, the Turkish government may be a powerful passive adversary inside Turkey, but likely has limited ability to affect the rest of the network, while an adversary such as the NSA or Chinese government may have truly global scope and be able to monitor most network traffic.

In general for email, the threat model has been historically confined to *network adversaries* who are attacking the network transport between two or more email servers in order to read email messages. A *passive network adversary* is easily defeated by enforcing the use of encryption on the network transport, such as by enforcing the usage of TLS between the client, the server and all MTAs. As older versions of TLS have been open to attacks[2], it is possible that traffic can be copied to be decrypted later, but we will assume that proper deployment of a modern version of TLS that has bug patches or TLS 1.3 can prevent this attack. An *active network adversary* may commit attacks such as replay attacks on email traffic. These attacks are generally prevented by using authentication with certificates, although studies have shown that most providers fail to use certificates correctly[19].

A more powerful *active server attacker* may attempt to read the messages of many users at once by attacking the server.

The active server attacker uses either technical attacks or legal means to force a server to hand over the private messages of its users and any keys so the attacker can decrypt the encrypted messages. An active server attacker may also ask for TLS keys to allow it to decrypt previously recorded TLS traffic. To prevent this, email needs to be encrypted on the server such that private key material does not reach or remain in cleartext form on any server, so that an attacker cannot decrypt the encrypted message by compromising the server or placing the server under legal compulsion. An example would be Lavabit, which had a single point of failure in the form of the system administrator himself: Ladar Levinson had access to the key material for all his users, defeating the purpose of having end-to-end encrypted email.[1] Strangely enough, other services such as Protonmail[2] seem to be repeating this flawed model for encrypted messaging. Lastly, this is trivially true (as shown by the NSA Prism programme) for centralized messaging services such as Gmail that do not store the content of messages encrypted. Server seizures are a threat to providers in the USA that legally resist backdoors, such as recent seizures of a Mixminion anonymous remailer on *riseup.net*[9, 18]. However, even if the messages themselves are encrypted, an active server attacker may be able to compromise server-internal metadata. An active server attacker is usually temporally bounded, and a stronger attacker would be a *malicious server* that attacks its own users to read their messages or metadata.

If there is concern over TLS being broken or the server being compromised, then end-to-end encryption (i.e. from client to client) may be used with PGP or SMIME. However, this may cause the attacker to focus on the client itself, leading to an *active client attacker*. An active client attacker may attempt to read a single or group of encrypted messages by impersonating another user, for example, by providing a false public key. Another active client attacker may attempt to gain access to the client in order to obtain the private key necessary to decrypt all the client's email. Such attacks are made possible, for example, by attacking the client software upgrade process or simply gaining root access to the client itself. Dealing with this kind of attacker is beyond our scope and needs to be dealt with by upgrades to the MUA.

The primary kind of attacker that is to be dealt with by PANORAMIX is a *passive network adversary* that simply copies all messages in transit between any two email servers (encrypted or not). The goal of the *global passive adversary* is to simply watch all the content on the network between the various MTAs as well as from clients to their email providers. The primary goal of the global passive adversary should also be considered that of collecting metadata, including the social network of a given user that another user is communicating with via email. This may be accomplished by looking at the timing and size of the messages via traffic analysis. It may also passively copy data on the network, and so may be read the content of the messages by breaking both TLS or end-to-end encryption eventually. Today, email systems offer no protection against global passive adversaries. Furthermore, even if the social connections of a user are unknown, the global passive adversary may be interested in other aspects of the metadata, such as the timing of the messages. For example, a message delivered through an IP-level anonymizer such as Tor can be detected simply by using timing information combined with geographic information [10].

Lastly, the most common adversary for email is an *abusive user*. Although not typically thought of as an attacker, these kinds of attacks are very common on email systems, even when the user is authenticated. These users often lead to spam problems and denial of service attackers.

4.2 Requirements

Requirements are divided between security properties and privacy properties. Although there are a number of security requirements that are not directly relevant to mix networking as provided by PANORAMIX, the are necessary for the underlying email system to be trusted by users.

The email system as currently structured works as an open system with a number of properties that are desirable to maintain. These differentiate email from many other non-email systems and so it is important to maintain these over and above any other messaging systems:

1. *Protection from abuse*: Ideally an email over mix-nets should not be more vulnerable to spam or other system abuse than existing email systems. A provider that supports mix-nets will still need to disable the accounts of

[1] *https://www.thoughtcrime.org/blog/lavabit-critique/*
[2] *https://protonmail.ch*

users abusing system resources and defend against spam and denial of service attacks.

2. *Delivery robustness*: Even if the recipient provider is currently unreachable, delivery should eventually succeed unless some high threshold has been reached and the sending provider decides to finally give up on delivering the message.

3. *Bounces*: The ability to support bounced messages like normal SMTP. In other words, if a message does not get delivered, for any reason, a bounce message should be generated and sent back to the original sender. The time period for a true failure of delivery should be quite long, although warning should be given of the delay to the user as soon as possible.

4.2.1 Security Requirements

In terms of attackers, we need to secure the privacy-enhanced email system developed by PANORAMIX against both active network attackers and active server attackers as a precondition for privacy-enhanced email. If there was a successful network or server attacker, then the security properties of email (confidentiality, authentication and integrity of the messages) would be violated. The first requirement is that *confidentiality and integrity of messages be preserved on the network level*. The second security requirement is that *messages between providers must be authenticated*. The final is that *messages between a client and their provider must be authenticated*. These are the properties provided by the correct use of TLS in email[19].

Furthermore, we will note that these properties as applied to the content of an email are guaranteed on the TLS level *only* to the email providers, not the clients themselves. The further security properties that we require are that *confidentiality and integrity of messages be preserved between clients*. The second security requirement is that for messages that are not coming from an anonymized source, *messages between clients should be authenticated*. Whether or not anonymous messages, as typical of anonymous remailers, will be supported by the LEAP codebase is left as an open design decision for PANORAMIX at this stage. Only by using end-to-end user-centric encryption from client to client via S/MIME or PGP can these security properties be extended to the clients (and so users) themselves. As email is an open and federated protocol and S/MIME requires a centralized key infrastructure, PGP is to be preferred.

4.2.2 Privacy Requirements

In contrast to security, privacy requirements are the main added value of PANORAMIX to email providers such as Greenhost. While privacy builds on security, currently existing systems such as a properly configured Postfix codebase or even the end-to-end encrypted PGP-based systems only match the security requirements, not the privacy requirements.

The fundamental attacker that PANORAMIX needs to counter is the *global passive adversary*. The goal of the global passive adversary is to watch network traffic to determine for a given user, all other users the user is communicating with, and the adversary can observe and record all traffic in the network. One privacy requirement is *unlinkability*. Unlinkability can be defined as the inability for an attacker to link together two actions and so use those actions to distinguish two items of interest. In particular, we are interested in *user unlinkability* where the passive adversary cannot distinguish a user from any other user based on sending of messaging, including any metatdata in the message. A more powerful privacy requirement is *provider unlinkabilty*, where the adversary cannot determine what provider a user communicates with, so that the user could be with many possible providers. In general, we want *third-party anonymity*, where a "third party" cannot de-anonymize a given user by observing their sending of messages. Most email systems do not offer anonymous communications at this time as users and service providers are given stable identifiers. A weaker enemy may be defeated by *pseudonymous communication*, where users have stable identifiers bound to one or more service providers that may or may not be tied to their actual person. These pseudonymous identifiers may be discarded. However, over time it is likely that any even local adversary with enough computational power can use machine-learning to de-anonymize a pseudonymous user.

There are also privacy requirements that could be met by PANORAMIX if the email provider of a user is not trusted by the user themselves. This could be due to either a *malicious server* or an *active server attacker* threat model. Having the privacy requirements apply to the server would counter these powerful attackers. Another privacy requirement that is more of a long-term goal for PANORAMIX is that the *users should be unmappable to providers*. We hope privacy-preserving email based on mix networking should be able to keep private the information regarding who is communicating with whom from not only outside observers but also from the service providers themselves. Specifically, two possible privacy requirements are *recipient anonymity from provider* where the email provider should not know the address of the recipient and *sender anonymity from provider*, where the provider should not know the address of the recipient. In a weaker version, these properties may then be given to hold between MTAs, but not at either the "first" hop of the sender into the mix network via an MTA or at the MDA of the recipient. In the strongest version, these properties will hold except at the MUA of the users themselves (i.e. in their client devices), so that the servers have limited to no message passing powers. These should hold true even when the sender and recipient are using the same provider. Stronger requirements could require either sender or recipient anonymity from the client (user) themselves as in a classical anonymous remailer, but this would likely create a large number of spam problems and so we will consider these cases out of scope, at least in the initial design.

4.3 Problems and Meditations

In this section, we go through how the security and privacy requirements may be fulfilled in current systems in the face of particular threats, as well as presenting initial ideas around how mix nets may help or hurt current best practice in email. After each existing problem, we also outline the mediation and how mix networking may impact current practices of mediation.

4.3.1 Security Requirement Problem: StartTLS downgrade

StartTLS is opportunistic transport encryption for either client-to-server or server-to-server SMTP relay.[3] Because it is opportunistic and unauthenticated, a man-in-the-middle attack can easily disrupt the negotiation and downgrade the TCP connection to cleartext. StartTLS sessions are not authenticated because it is common practice for MX servers to use a self-signed x.509 certificate for the purposes of StartTLS. Therefore, we must strictly enforce TLS with authentication via valid certificates.

Mediation

Existing email systems and mix nets should start supporting only StartTLS connections. Where the other party presents a server certificate from one of the established certificate authorities, it would greatly reduce the number of StartTLS encrypted sessions. In the past a new certificate was considered dangerous. However, advances such as the *Let's Encrypt* project that provides certificates for free, are likely to change the practice of self-signed certificates for StartTLS as they will allow servers to get validated certificates.[4] This approach has been put forward by the *StartTLS Everywhere* effort from the EFF.[5] For email connections over the mix network between the mail server and MUA and any other MTA connection, valid certificates should be used to authenticate the session.

4.3.2 Security Requirement Problem: DNS hijacking

There are many methods an attacker can use to trick one service provider into fetching bogus DNS MX records for another provider. Because DNS queries are not encrypted and typically not signed, a man-in-the-middle that is interposed in the network between a service provider and a DNS server can rewrite the DNS queries to return whatever MX server the attacker chooses. Such bogus results look fairly normal, since it is common to have the MX server for a domain be handled by a third party.

[3] *http://www.ietf.org/rfc/rfc3207.txt*
[4] *https://letsencrypt.org/*
[5] *https://github.com/EFForg/starttls-everywhere*

Mediation

The most common recommended and standardized mediation for DNS hijacking is to use *DNSSEC/DANE* deployment, which would prevent DNS hijacking if supported by the other party's MX server. DNSSEC allows the authentication of information provided by a domain name,[6] and DANE allows key material to be bound to that domain.[7] However, the main problem is that the actual set-up of DNSSEC and DANE is complex and non-automated, and thus not heavily deployed. This process should be automated as much as possible and domains that support the use of mix nets for email should follow DNSSEC and DANE.

4.3.3 Security Requirement Problem: MX impersonation

Note that entirely separate from DNS hijacking there is simply the hijacking of an MX server. This is possible as there is typically no authentication, so any man-in-the-middle attacker can just pretend to be any MX server for any domain. Although MX impersonation can happen by a compromised domain, many email systems will tend by default to accept incoming MX email.

Mediation

The Sender Policy Framework (SPF)[8] allows a domain owner to specify which hosts are authorized to send email on its behalf. However, it is IP address based, so it may not be very useful for mix nets although we could limit IP addresses to those authorized explicitly as part of a static mix net, although this would not allow dynamic mix network topologies. The other technique is DKIM, which allows the sending provider to vouch for the authenticity of certain headers and the body by adding a header with a digital signature. To verify these signatures, the recipient MTA can fetch the RSA public key from the DNS records of the sending provider. Minor modifications to DKIM that allow the "From" header and other metadata to be wrapped, given the changes in headers likely needed by PANORAMIX, will be necessary for DKIM to work with mix-nets. SPF may work for mix networks between MTAs where we "greylist" known IP addresses for MTAs, as well as DKIM.

4.3.4 Privacy Requirement Problem: Abusive Users

One of the primary threats that a service provider faces is that of their own users. Even for very small service providers, if they offer accounts to the public, they will have many instances of users who send spam, phishing attacks, or some other form of abusive behavior. A major function of an email provider is to govern their own users. However, the concern is that the mix network infrastructure will make this impossible.

Mediation

The most important step that a provider can do to keep a good reputation is to aggressively govern the behavior of their users. While this goes against many of the goals of those that wish to encourage privacy and anonymity, there does not seem at the moment to be an easy technical solution to the problem of abusive users. This requires that a provider is able to identify with certainty which user sent a particular email when that email is reported as spam or abusive and to remove or suspend that user account. However, as a mix network provides anonymity for users from the service provider itself, then this problem becomes nearly impossible to solve unless some sort of function is provided to allow users to be tracked and accounts disabled without revealing the identity of the user. This could take the form of a "coin" or "token" used by a user that can be disabled, if the "coin" is not attached to the identity of the user in a provable way (for example, the coins could be mixed).

[6] *https://datatracker.ietf.org/wg/dnssec/documents/*
[7] *https://datatracker.ietf.org/wg/dane/charter/*
[8] *https://www.ietf.org/rfc/rfc6652.txt*

The second measure a provider can take to keep a good reputation is to offer limited - even if the limitation is "artificial" - capacity. If the email provider simply does not let a user send very many emails, then the potential impact of an abusive user is minimized. the provider should also make it hard to create new user accounts. This makes spammers and phishers move on to other providers where their life is easier. In this case, for mix networks there still needs to be an on-boarding process that is more complex than traditional ones, and in particular it may require more checks in order to ensure the user will not abuse the mix net for messaging. Furthermore, strict if artificial limits of the sending of mail or use of storage may help prevent abusive behavior.

The last measure is *abuse reports*, that offer some mechanism for other email providers or email recipients to notify an email provider if there appears to be abusive behavior. Currently, abuse reporting is not standardized, although there are blacklists and other reputation mechanisms run by M3AAWG.[9] Since PANORAMIX will be adding mix networking, we would need to specify how mix net based email needs to be reported between providers and make sure we can also receive complaints about users on a particular provider. It could be something as simple as a dedicated address such as *abuse@domain.org*, and messages must keep to a specific format such as the "Abuse Reporting Format."[10] However, abuse reports break metadata protection as it will require identifying the user. Yet if the recipient of abusive email needs to report abuse and can provide the email address sending the abuse, it seems that there should be some manner for a provider to de-anonymize to itself (not necessarily the other provider or an outside observer) so that investigation of abuse reports lose the metadata protection mix nets provide.

In terms of privacy requirements around sender or recipient anonymity, it is necessary that the "from" header can be used for identification of a sender to the end-recipient, so long as the provider does not allow the sender's MUA to send arbitrary "from" headers. If sender or recipient anonymity is not required, an "authenticated sender" header can be used that uses DomainKeys Identified Mail (DKIM) that provides a signature of the body and header from the provider as part of an email.[11] This assumes that the agent reporting a message as spam, probably the user's MUA, has full access to all the headers (from user, to user, from provider, to provider). The problem in both these cases is that the sending user is often able to change their account name or alias at will. This would allow a spammer to send a bunch of spam, then change their username to avoid getting their account shut down (of course, they could not receive replies in this scenario). However, again rate de-limiting and a complex on-boarding process, perhaps with time delimited 'tokens' that could take the form of even a "crypto-currency coin" for access, may prevent this. Another non-standardized possibility is to add a "token" to the header in the email that maps exactly to the user's account, but is unique for every message so that it cannot be used to correlate all the other messages sent by the same user by the recipient provider. The header could be the user's unchanging account ID encrypted using a server secret and the email message ID so it was reversible, and it could be used by the provider to prevent spam without revealing the sender to the provider of the recipient.

4.3.5 Privacy Requirement Problem: Spam

One of the most common abuses of users of email systems is the use of spam. The legal definition of spam is fairly narrow including unsolicited commercial email in which the recipient is not able to opt-out. In practice, this legal definition is often ignored. From the perspective of a service provider any email that is not wanted, no matter how legitimate, is spam, both in terms of incoming messages delivered to the provider's users and outgoing messages sent by the provider's users. The reason for this is that sending unwanted email, regardless of its type, can be a serious strike against the reputation of a provider and so allow them to be "black-listed" by other providers, and delivering too much unwanted email will frustrate users and drive them away.

Spam is often not sent by users with the ability to get through. Instead, a "spam bomb" is a type of denial of service attack in which an attacker floods the target MX server with a massive amount of incoming email. Unlike typical spam, the purpose of a spam bomb is not to have anyone ever read the messages, which can be entirely bogus, but to overwhelm the resources of the MX server that is trying to process the flood of incoming mail and so (temporarily) shut the email server provider down. Even if every message in a spam bomb can be disregarded as easily identified as spam, a large flood of email can overwhelm the capacity of the spam filters and so a spam filter may be ineffective.

[9] *https://www.m3aawg.org/*
[10] *https://www.irt.org/rfc/rfc5965.htm*
[11] *https://www.ietf.org/rfc/rfc5672.txt*

Mediation

In email systems today, email providers rate limit incoming mail based on their heuristic of the reputation of the relaying MX server. This is an incredibly useful anti-abuse prevention measure for them yet unfortunately small providers do not have the ability to do this.

Allowing everyone to be able to limit sending providers is not necessarily a disadvantage but could also cause problems that are specific to mix networking. For example, mix net email could suffer from a kind of sybil attack where it is easy to start a new relay and flood email providers with mail, then create another provider and repeat. It is important to note that the goal of the LEAP codebase is to making it so that you can start an email provider in a short amount of time.

Nonetheless, the current email system is vulnerable to this right now, and most providers don't have much defense against unexpected "spam bombs" from new providers and yet the email system has not yet fallen apart because of this. But maybe there is a specific reason: for normal email, email-based DDoS is expensive to pull off, because most ISPs block port 25, and it is easy to just block the troublesome IPs (and not so easy for an attacker to get new IPs with port 25 open). Given the proliferation of cheaper DDoS attacks, email-based ones are currently in decline, but a mix net email-based DDoS could be easier to pull off, and harder to prevent. Due to this threat, having a dynamically expanding mix network is difficult compared to a static mix network with an expicit network of trust. This could be ameliorated by a reputation system for email servers using the mix net.

Another common technique in email is *Realtime Blackhole Lists* (RBLs). RBLs are the mail provider's primary line of defense against spam, but most RBL lists are lists of IP addresses that should be blocked. Given it is based on IP addresses of the message sender being known, the technique is not very useful for mix nets. Some RBLs are lists of domains, like the Spamhaus Domain Blocklist, so that could be used.[12] RBLs are the main line of defense for most providers but almost all RBLs are IP address based (a database of IP addresses to block). However, a domain-based RBL could be created that would work in a mix net if all the domains co-operating in the mix net could be discovered by an email provider, and such DNS-based blacklists have already been standardized.[13] Furthermore, such a list could be useful if mail is to be sent out of the mix net and into the wider network. For example, Tor provides a list of IP addresses of known Tor exit nodes to prevent confusion over the source of traffic from Tor,[14] so providing a list of domains that use PANORAMIX would also be needed if email for PANORAMIX is to enter the larger federation of email. One solution is to create a new kind of RBL: the MORBL (Mix net and Onion Routing Blackhole List). We could then define a PANORAMIX-specific mechanism to detect and report mix net abuse, and submit these reports to MORBL through a reporting protocol customized for PANORAMIX. The MORBL could feature the IP addresses of known exit nodes from the PANORAMIX mix network as well as domains of PANORAMIX-enabled email providers.

4.3.6 Privacy and Abuse Prevention Mediations

As email is an open and federated architecture and anonymity from the server and other MTAs creates a situation that has a potential for abuse, PANORAMIX will need to address a broad approach to abuse prevention if privacy requirements necessary for anonymity from the server are to be implemented. There are two broad approaches to abuse prevention:

- **Delegated reputation:** If the email provider generally has a good reputation, then this reputation is also passed on to their users. Currently best practice in email.
- **Pre-authorization:** Both users must authorize one another in advance in order for messages to be delivered. Possible using end-to-end encryption, although not done widely in practice and has usability issues.

[12] *https://www.spamhaus.org/dbl/*
[13] *https://www.ietf.org/rfc/rfc5782.txt*
[14] *https://check.torproject.org/cgi-bin/TorBulkExitList.py*

Delegated reputation

Delegated reputation is what works on email systems at the moment. However, in PANORAMIX with any anonymity-related privacy requirement where the server is untrusted, the email providers will be delivering messages where they may not be able to know the reputation of the original provider nor know the identity of the recipient. This is in conflict with two underlying issues that are tackled by delegated reputation systems today:

- Sending providers want to maintain a good reputation.
- Receiving providers want to block bad reputation relays.

To maintain a good reputation a server can rate limit how many messages a user can send in a give time window, such as 100 per hour. So long as the sender must relay mail through their provider and is not able to directly inject messages for delivery into the mix net via multiple servers, this allows server reputation to hold. A server can also be able to receive abuse reports from the recipient of the message. These reports are often sent to *abuse@domain* or through RBLs or through an automated feedback loop. Note presently that some providers like AOL use their own non-standardized system, so one cost of running an email system is that you must use the AOL system if you want to send email to AOL to be able to positively identify the user account for a particular spam message and to punish, delete or disable the account. This requires that the spam report contains some kind of unique identifier that can be traced back to the user account. So in order to do delegated reputation, sender anonymity from the provider at the MDA as a privacy requirement should be dropped, and likely recipient anonymity from the provider also be dropped so the email sender can validate the recipient. Note that the user's account does not have to be revealed, as a "coin" or unique "token" system that simply allows the provider to be identified, but then the provider to identify the abusive user, may be enough. Using this "coin" or "token" in combination with the mail sent, a provider may identify a user. However, the privacy properties of this sort of solution need to be analyzed.

It must be necessary to block incoming mail based on reputation, and this requires to authenticate a sender's provider. The recipient provider needs to be able to have confidence that a particular email really does come from the provider that is claiming to have sent it. Then the next step is to check the domain against an RBL (list of malicious providers). Once the sender's domain is authenticated, then the recipient provider must check against a database of "spam" domains and block if there is a hit. In a mix net situation, PANORAMIX-enabled email providers can enforce sending provider authenticity over domains if we cannot use IP addresses. It is also necessary to have the ability to report other providers as bad actors. We could just use the DKIM support that is already built into the MTAs. Traditional use of DKIM is not possible with a mix net, because the primary value of DKIM is signing the "from" header, which we want to hide from the recipient server. However, DKIM would still be useful for verifying authenticity of the sender's provider. In detail, the sender's provider would add a header containing just their domain, then sign it using the private key that corresponds to a public key fetchable via a DNS TXT lookup on the sender's provider DNS. Current MTAs can do all this automatically if configured to do so.

Lastly, PANORAMIX could use a greylist of all email received over mix nets. This would mean that the provider puts "on hold" any email from a domain it does not recognize (via techniques such as SPF, RBLs). If the mail is not delivered and the provider is not compromised, then a legitimate MTA would simply try to deliver the message again. The main drawback is that this would really slow things down for new providers.[15]. Greylisting works because the recipient MTA can immediately terminate the incoming connection from the sender MTA before it needs to do any expensive processes, and so puts all the work back on the sender MTA. With mix nets greylisting puts all the work on the mix net itself, which would increase the expense of using the mix network and possibly interfere with any timing paramters in the mix. In addition, "whitelisting" the domains of good actors could be useful and formalize the conditions of a provider for entry into the PANORAMIX network. Since we expect PANORAMIX to be started with a network of trusted providers, this would be feasible with a static topology or a dynamic topology where the network could update the whitelist dynamically.

[15] *https://en.wikipedia.org/wiki/Greylisting*

Chapter 5

LEAP Software

In order to meet the security requirements, the first precondition to having PANORAMIX-based mix networking work over email systems it that systems properly deploy both TLS for the network level between the client and provider, as well as any intermediary MTAs. A second precondition is to use end-to-end encryption. While it is possible to configure an email provider to use TLS properly, it is nearly impossible for an email provider to use end-to-end encryption for its users. The LEAP codebase is an open source codebase that will be deployed by Greenhost and Mobile Vikings to provide end-to-end encryption for their users. LEAP is a recursive acronym for the "LEAP Encryption Access Project." LEAP is still in development, although the core functionality of basic opportunistic encrypted email is now available for beta testing.[1] The project source-code on Github is available to all.[2] LEAP infrastructure will be supported by providers such as Greenhost. The description of LEAP below comes from the wiki pages available publicly[3] and an earlier version of the design documentation presented at the Surveillance and Technology workshop[29]. The wiki-pages are kept up to date and provide more detail. The goal of our work is to add support for privacy requirements to LEAP using the PANORAMIX API and any other integration work needed.

Well-understood technologies such as OpenPGP-based email encryption are not used by the vast majority of people for reasons that have been understood for nearly a decade and a half [31]. While there has been considerable progress in the deployment of IP-address level anonymity via the Tor project[11], most people rely on insecure and centralized silos for email. There are few working solutions for encrypted and anonymous email. While Tor provides the best solution for IP-level anonymity, this purpose is defeated when users rely on centralized email systems, where the dangers of their communication being intercepted via disclosures by the service provider are considerable [12]. For example, many users simply use Tor to "anonymize" their access to email services such as Gmail that can simply hand over their data, or even systems such as *riseup.net* that likely have all outgoing and ingoing traffic monitored even if the server itself refuses requests for user data. Beyond email, Off-the-Record messaging for chat works well, but requires synchronous chat between two users.[4] Current high-profile efforts such as Mailpile are aimed at essentially replacing the user-experience of Thunderbird and Enigmail, not at actually solving the underlying problems of key management and provisioning encrypted email.[5] Although message security rests entirely on a foundation of authenticity, since without proper validation of encryption keys a user cannot be assured of confidentiality or integrity, current systems of establishing message authenticity are so difficult to use that many users simply ignore this step [21]. Our goal should be mass adoption of encrypted email. To achieve mass adoption of encrypted email, key provisioning, key validation to determine message authenticity, and managing the server-side must be done as well as an excellent client-side user experience.

The design of the LEAP codebase tackles each of the requirements for high data availability, automatic key authenticity, and unmappability. The primary new contribution of LEAP is tackling the problem of high data availability while defending against active server attackers: How can we keep the key material from being inaccessible to the server

[1]To try, follow instructions on *http://demo.bitmask.net*.
[2]*https://github.com/leapcode/*
[3]*https://leap.se/en/docs/design*
[4]*http://www.cypherpunks.ca/otr/*
[5]*http://mailpile.is*

and at the same time having the keys available for syncing across devices? The problem can be broken down into a number of distinct components: Server-side infrastructure, usable client software, and the fundamental protocols needed to communicate between the server and the client. What is necessary is to have the client and server actively work together in order to encrypt the message, as to prevent the situation where private key materials stored only on the server are defended only by weak defenses such as passwords. Simply storing the private key on a single device of the user, as done by most encrypted mail programs, is not enough as users need to access their email through multiple devices and keep the state of their inbox synchronized. Thus the main problem facing such a system is safely getting the correct keys onto users' devices, a problem known as *key synchronization*. This becomes an even more important problem if best practices such as frequent key rotation are to be employed.

LEAP solves the problem of key synchronization through the installation of a multi-purpose LEAP client application called Bitmask, that appears to the user mainly as an OpenVPN[6] client. However, there is more to Bitmask than just a VPN. Inside of the Bitmask client are the routines for generating, validating, and discovering keys as well as synchronizing keys and related material (such as the status of messages being "read" across multiple devices). The LEAP client appears to be a VPN as many users likely would install a VPN (but not special 'key manager' software) and the VPN provides additional security benefits by creating an authenticated and encrypted channel for all traffic between the LEAP client and server. Currently the VPN is required for use with LEAP's encrypted email provider. In the future, we hope to de-couple the VPN from the email service while still providing an authenticated channel for only email and to also support Tor as a transport protocol for the VPN.

The LEAP client keeps the messages on the server where these messages are stored in an encrypted form so that the sever cannot read the messages as it does not have access to the private key material to decrypt the messages. The server has only the necessary metadata needed to synchronize and download messages between the user's clients on various devices. When the messages are downloaded, the LEAP client can then decrypt them with the user's private key material that is unavailable to the server. While some email clients may natively support encryption, LEAP allows users to continue using their existing non-Web email clients by providing local proxies for IMAP and SMTP: This way not only can locally decrypted messages can be displayed to the user, but the LEAP client can capture unencrypted outgoing email, automatically encrypt the email with the private keys in the local keyring, and then send it out using the LEAP data synchronization protocols. This prevents the server from accessing the data while still maintaining the same state across multiple devices. Note that the high-security design of LEAP prevents the use of webmail via generic browsers, as the server controls the execution environment and thus can access or easily spoof key material. Until browsers allow private key material to be handled safely, our security requirements require the installation of a LEAP client separate from the browser. Browser plug-in approaches as being pursued by Google and Yahoo! end-to-end encryption projects are preferable, but still suffer in terms of data availability and verification of lack of malicious code in the plug-in.[7]

5.1 The LEAP Architecture

In detail, the LEAP federated architecture consists of three-components: (1) a server-side platform automation system; (2) an easy-to-use client application; and (3) new protocols such as Soledad and Nicknym that allow the user to place minimal trust in the provider, as well as well-known and standardized protocols such as IMAP. These components are illustrated in Diagram 5.1.[8] The cryptographic details are also subject to change (in particular, migrating from large RSA keys to Curve 25519 when possible) and are maintained online.[9]

The LEAP platform offers a set of automation tools to allow an organization to deploy and manage a complete infrastructure for providing user communication services in the servers controlled by them. The LEAP client is an application that runs on the user's local device and is tightly bound to the server components of the LEAP platform. The client is cross-platform, auto-configuring, and auto-updating, with the initial configuration and updates verified via The Update Framework[10] in order to prevent a compromised server from forcing new key material or accessing

[6] *http://openvpn.net/*
[7] *https://github.com/google/end-to-end*
[8] Note that parts of Section 5.1 are modified versions of material available on the LEAP wiki at *http://leap.se/en/docs* (Accessed June 5th 2015).
[9] *https://bitmask.net/en/features/cryptography*
[10] *http://theupdateframework.com/*

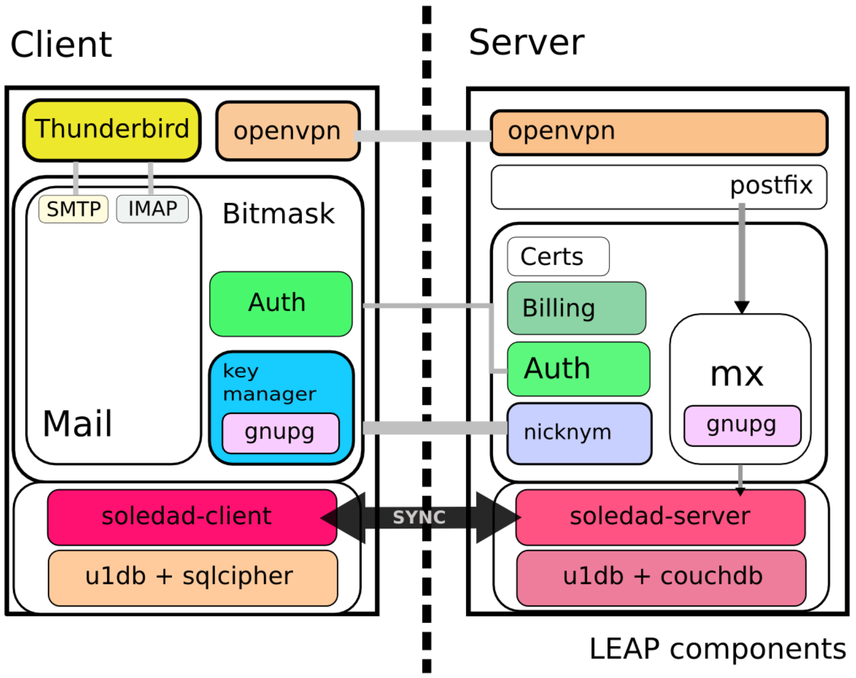


Figure 5.1: Components of LEAP Email Architecture

the existing client key material via a compromised update.

5.1.1 LEAP Platform

The LEAP platform consists of a command line tool and a set of complementary puppet modules. The recipes allow an organization to easily operate one or more clusters of servers to provision LEAP-enabled services. With the LEAP command line tool, a system administrator can rapidly deploy a large number of servers, each automatically configured with the proper daemons, firewall, encrypted tunnels, and certificates. The LEAP platform recipes define an abstract service provider, with recipes for complementary services that are closely integrated together. To create an actual infrastructure, a system administrator creates a "provider instance" by creating simple configuration files in a filesystem directory, one for each server. A system administrator will not need to modify the LEAP Platform recipes, although they are free to fork and merge as desired. The "provider instance" directory tree should be tracked using source control and is a self-contained encapsulation of everything about an organization's server infrastructure (except for actual user data).

LEAP Data Storage

One design goal of the LEAP platform is for a service provider to act as an "untrusted cloud" where data are encrypted by the client before being sent to the server, and we push as much of the communication logic to the client as possible. There are a few cases where the server must have knowledge about a user's information, such as when resolving email aliases or when processing support requests. In the current implementation, data storage is handled by CouchDB. Every user has a personal database for storing client encrypted documents, like email and chat messages.

The unencrypted information stored on the server needed to resolve email, including the database for routing incoming and outgoing email, is similar to any traditional email provider, with the one exception that user accounts don't have traditional passwords. Mail is received via a Soledad synchronization session (detailed in Section 5.1.2) and authenticated using Secure Remote Password (SRP).[11] Mail is sent using SMTP with SASL authentication using client certificates [12]. In detail, the unencrypted database of user information maintained by the service provider includes:

- *username:* The login name for the user. This is not necessarily the user portion of 'user@domain'.
- *SRP verifier:* Akin to a hashed password, but used in SRP 'zero-knowledge' dual pass mutual authentication between client and server.
- *uuid:* Random internal identifier for internal usage.
- *identities:* One or more 'user@domain' identities, with corresponding public keys and separate authentication credentials for SMTP (stored as a fingerprint to an X.509 client certificate). Each identity has its own authentication credentials so that the email headers show that the user authenticated with the SMTP server using their identity username, not their real username. Each identity includes delivery information, either to a uuid or to a third party email address that messages should be forwarded to.

For email delivery, the receiving MX (Mail exchanger record) servers do not have access to the entire database. They only have read-only access to 'identities.' This allows the implementation of the Nicknym protocol (described in Section 5.1.4) for resolving pseudonyms. Additionally, there are several non-encrypted databases containing the minimal information needed to connect user accounts to optional support tickets and even billing details. The LEAP platform includes a web application for user and administrator access to these non-encrypted databases, although future research will hopefully be able to minimize if not eliminate this information.

5.1.2 Soledad

Soledad ("Synchronization Of Locally Encrypted Data Among Devices") is responsible for client-encrypting user data, keeping the data synchronized with the copy on the server and on all the other devices of each user, and for providing local applications with a simple API for document storage, indexing, and search that is akin to CouchDB and related document-centric NoSQL databases. The document that is saved and synchronized with Soledad can be any structured JSON document, with binary attachments. Soledad is implemented on the LEAP client to store email messages, the user's public and private OpenPGP keys, and a contact database of validated public keys. Soledad is based on U1DB, but modified to support the encryption of both the local database replica and every document before it is synchronized with the server.[13] Local database encryption is provided by a block-encrypted SQLite database[14] via SQLcipher.[15] Documents synchronized with the server are individually block encrypted using a key produced via an HMAC of the unique document id and a long storage secret. In order to prevent the server from sending forged or old documents, each document record stored on the server includes an additional client-computed MAC derived from the document id, the document revision number, and the encrypted content. The server time-stamps each update of the database, so that Soledad's MAC and HMAC keys used to encrypt the client database can only send the server new databases. Each time the LEAP client is online (both after re-connecting with the LEAP platform and after each pre-set time interval, the client re-synchronizes the messages and key material.

In addition to synchronizing public-private key materials and a contact list of validated keys, Soledad is used to encrypt and synchronize email. This has the benefit not storing some sensitive metadata on the server and allowing for searchable locally encrypted database of messages. For efficiency, a single email is stored in several different documents (for example, for headers, for attachments, and for the nested MIME structures). While in transit between LEAP-enabled SMTP servers and the LEAP client, there are three different forms of encryption that a single message is subject to:

1. Encrypted by sender, or on arrival by recipient's service provider using OpenPGP or S/MIME.

[11] *http://www.ietf.org/rfc/rfc2945.txt*
[12] *https://tools.ietf.org/rfc/rfc4422*
[13] *https://one.ubuntu.com/developer/data/u1db/*
[14] *https://sqlite.org/*
[15] *http://sqlcipher.net/*

2. Decrypted from and re-encrypted in an SQLcipher database using AES block encryption.
3. Individually re-encrypted for storage on a service provider that supports the LEAP platform using block encryption with a nonce.

5.1.3 LEAP Client

The *LEAP client* (also known as Bitmask[16]) is a cross-platform application that runs on a user's own device and is responsible for all encryption of user data. It currently includes the following components: *Bitmask VPN*, *Soledad* (multi-device user data synchronization), *Key Manager* (Nicknym agent and contact database), and *email proxy* (opportunistic email encryption). The client must be installed in the user's device before they can access any LEAP services (except for user support via the web application). Written in Python (with QT, twisted, OpenVPN, SQLcipher), the LEAP client currently runs on Linux and Android, with Windows and Mac being under development.[17] When a user installs a LEAP client, a *first-run* wizard walks the user through the simple process of authenticating or registering a new account with the LEAP provider of their choice, using the Secure Remote Password (SRP) protocol so that a cleartext copy of the password never reaches the server.

Note that when a user authenticates with the client, via a username and password, these credentials as provided are used to both authenticate with the service provider (via SRP) and also to unlock locally encrypted secrets (via Soledad). In the future, LEAP is interested in supporting best-of-breed standards beyond SRP for user authentication beyond passwords, including multi-factor authentication. However, any multi-factor authentication scheme that relies on a shared secret between the user (or a device in the user's possession) and the server would not help when unlocking locally encrypted secrets. If a user had a multi-factor trusted hardware dongle that supported deterministic signatures, then this information could be mixed in with the user's username and password to achieve multi-factor authentication. For example, devices that support the OpenPGP card protocol are able to store secrets which could be mixed into the authentication process of LEAP to provide second-factor authentication (albeit a second factor that the provider has no knowledge of).[18]

One threat would be that an active server attacker would compel a LEAP-enabled server to push a malicious update to the clients to compromise their keys. This threat applies equally to any browser or plug-in based approach, and in fact to the installation of *any* software. LEAP employs mitigation strategies to prevent this attack. When distributed through the self-contained bundles, the client has auto-updating capabilities, using The Update Framework (TUF) to update LEAP code and other library dependencies as needed using the same Thandy library as deployed by Tor [27].[19] Unlike other update systems, TUF updates are controlled by a timestamp file that is signed each day. This ensures that the client will not miss an important update and cannot be pushed an old or compromised update by an attacker. Updates to the LEAP client via TUF require signatures from multiple keys, held by LEAP developers in different jurisdictions. Lastly, LEAP has started work on a system of reproducible builds, which is working in an automated fashion for Android and in the future should apply to all other platforms.[20]

VPN

The goal with LEAP's VPN service is to provide an automatic, always on, trouble-free way to encrypt a user's network traffic. The VPN service encrypts all of a user's traffic and works hard to prevent data leakage from DNS, IPv6, and other common client misconfigurations that are not tackled by OpenVPN via a strict egress firewall. Currently OpenVPN is used for the transport. OpenVPN was selected because it is fast, open source, and cross-platform. In the future, LEAP plans to add support for Tor as an alternate transport. We believe LEAP is the only VPN that autoconfigures and auto-restarts when connectivity is lost. When started, the LEAP client discovers the LEAP-enabled service provider's proxy gateways, fetches a short-lived X.509 client certificate from the provider if necessary, and probes the network to attempt to connect. If there are problems connecting, the LEAP VPN client will try different

[16] *https://bitmask.net/*
[17] The Android version tends to lag behind development compared to the Linux version due to the design having to be re-coded in Java.
[18] *https://openpgpcard.org/*
[19] Thandy is available here: *https://gitweb.torproject.org/thandy.git*.
[20] See work by Debian on reproducible builds that LEAP is applying to its code: *https://wiki.debian.org/ReproducibleBuilds*.

protocol and port combinations to bypass common ISP firewall settings since VPN access is typically blocked crudely by simple port and protocol rules rather than deep packet inspection. In terms of deep packet inspection, obfsproxy[21] integration is under development to hide the VPN connection to an observer. By default, the LEAP client will auto-connect the VPN service the next time a user starts the computer if the encrypted proxy was switched on when the user the client quit or the machine was shutdown. If network connectivity is lost while the proxy is active, the LEAP client will automatically attempt to reconnect when the network is again available. A firewall is also activated before launching the VPN service, providing a fail-close mechanism that limits the unprotected access to the network in case of client malfunction or crashes. Due to its stringent security requirements, the LEAP VPN does not work when the user is behind a captive network portal.

5.1.4 Nicknym Key-Management

One of the main features of the LEAP system is to provide strong authentication of public keys in a way that is easy for users. To do this, LEAP relies on a protocol called Nicknym in the form of *username@domain* (just like an email address). Nicknym maps user nicknames to public keys. With Nicknym, the user is able to think solely in terms of nicknames, while still being able to communicate with a high degree of security (confidentiality, integrity, and authenticity). Another goal of Nicknym is to, unlike the OpenPGP 'Web of Trust' mechanism, not reveal the social graph of the user to the public.[22]

5.2 LEAP for Email Encryption Example

The LEAP Encrypted Email service is designed to client-encrypt messages whenever possible, be compatible with existing mail user agents, provide strong authentication of recipient public keys, allow communication with existing email providers, and be as user friendly as possible. Additionally, when mail is relayed to other LEAP providers, the LEAP platform will automatically establish and require opportunistic encryption for the SMTP transport. LEAP does not support forward secrecy of email messages. While only ciphers with forward secrecy are allowed for SMTP with StartTLS, which offers some degree of forward secrecy from a third-party network observer, SMTP with StartTLS forward-secret ciphers do not offer forward secrecy from the email provider of the sender or recipient. In the future, LEAP plans to adopt some version of Axolotl double-ratchet used in the Signal Protocol[23] between LEAP-enabled messaging users, where the user's client generates a pool of 'pre-keys' that can be used to create forward secret ECDH key derivation on the first message.

5.2.1 Setting up a new device

When a user installs a LEAP client, the LEAP client asks the user for a username and master passphrase, and to select a LEAP-enabled provider. The first time that a user authenticates a new device against that LEAP-enabled provider after installing a LEAP client, the keymanager on the LEAP client will attempt to perform a Soledad synchronization. If the user has created a new account and so no valid keypair is found, the LEAP client will generate a public-private OpenPGP keypair on the user's device. After such generation is completed, the keypair will be symmetrically encrypted with a key derived from the master passphrase, and the wrapped key will be uploaded to the remote Soledad replica in the server, in order to let the user add new devices and synchronize them with Soledad (as described in Section 5.1.2). For advanced users, this behavior of automatically backing up a symmetrically encrypted copy of the user keypair will be changed to opt-out behavior, so that the experienced user is given the opportunity to manually transfer the key material to other devices under his control by any means of his choice.

[21] *https://www.torproject.org/projects/obfsproxy.html.en*

[22] Note this is a high-level overview, more information is found here: *https://leap.se/nicknym*.

[23] *https://www.whispersystems.org/blog/advanced-ratcheting/*

5.2.2 Receiving Mail

For incoming email, messages are received by the service provider's MX servers, encrypted to the current user's public key, and stored in the user's database in an incoming message queue. The LEAP client then fetches the incoming message queue as part of a periodic Soledad synchronization, decrypting each message and saving it in the user's inbox, stored in the local Soledad database.

The mail module exposes the stored messages through a local IMAP server, so that the messages can be accessed using any standard MUA. A plugin specific for Thunderbird, which is the agent used for the first tests, is also provided that will configure an account to listen on the configured local ports for the IMAP (and SMTP) proxies, and take some precautions such as disabling the disk cache. It is planned that in the near future this plugin can communicate with the mail local backend via a IPC (Inter-process Communication) channel in order to display specific data about the message encryption and signature validation.

As with the outgoing and incoming mail services, a generic protocol-agnostic API for the account is also exposed at the code level, so that trusted third party applications can access all the message data and LEAP functionality: the Pixelated Project will probably be the first project to integrate with the LEAP client app ecosystem, exposing messages and keymanager capabilities through a web-based UI running locally.[24]

5.2.3 Mailbox Sync

Since email is distributed to the client and stored via the Soledad API, any changes to the mailbox will eventually be synchronized to all devices. The mutable parts of the messages and the attachments are kept in separate documents, so that the sync overhead is kept low. Future releases of Soledad will allow for selective synchronization so that header documents can be synchronized first, leaving the ability to download attachments in the background or on demand, which will be specially interesting for mobile.

5.2.4 Sending Mail

For outgoing email, the LEAP client runs a thin SMTP proxy on the user's device, bound to *localhost*, and the mail user agent (MUA)[25] is configured to bind outgoing SMTP to *localhost*. When this SMTP proxy receives an email from the MUA, it issues queries to a local keymanager (Nicknym agent) for the user's private key and public keys of all recipients. The message is then signed, and encrypted to each recipient. If a recipient's key is missing, email goes out in cleartext (unless a user has configured the LEAP client to send only encrypted email). Finally, the message is relayed to the provider's SMTP relay. The approach outlined here is similar to the approach taken by Garfinkle [20] and Symantec,[26] although these systems do not include key discovery, key validation, encryption of incoming messages, secure storage, or synchronization of email among devices.

5.3 Current State and Future Work

As of May 2016, the current LEAP architecture provides a VPN service (currently in beta-testing via *bitmask.net*) and end-to-end encrypted email service (in alpha-testing). The LEAP platform, Soledad, Nicknym, and the basic Key Manager are currently complete. However, there is still ongoing work on greater scalability and reliability for the LEAP platform's encrypted data-storage. On the side of the LEAP client, LEAP is pursuing greater compatibility with existing IMAP clients, improved usability, and better network probing for the VPN. In co-operation with Thoughtworks, we are working on a custom user-interface called Pixelated[27] to be bundled with the LEAP client for users looking for alternatives to existing email clients. Immediate goals also include porting LEAP from Android and

[24] *https://pixelated-project.org//*

[25] Such as Thunderbird, Evolution, or Outlook.

[26] *http://www.symantec.com/desktop-email-encryption*

[27] The source code for Pixelated is available here: *https://github.com/pixelated-project/*.

Unix-based environments (Linux and MacOS) as well as eventually iOS and Windows environments. Work is ongoing to improve the key validation rules (including key verification revocation) and support validation with multiple network perspectives. In addition, due to the NEXTLEAP EC project,[28] LEAP plans to support for CONIKS for key validation[24],and eventually the use of the Axolotl protocol between LEAP-enabled providers as a higher-security alternative to SMTP with perfect forward secrecy, back-ups for stolen key material, and deploying the latest working systems for reproducible builds.

Importantly for PANORAMIX, the LEAP codebase will add mix-networking for messages in transit both in between LEAP providers and clients to prevent metadata collection by passive global adversaries (to address many of the critiques of decentralized architectures via working with PANORAMIX [26]) via use of the PANORAMIX API and any other needed integration work. In the future, the LEAP codebase may expand its basic federated infrastructure to also provide hosting for other end-to-end encrypted and traffic-analysis resistant services needed by users, such as chat and Voice-over-IP. All of these services may benefit from the privacy-preserving properties provided by PANORAMIX.

At this moment, email providers such as *Greenhost* provide centralized email services with tens of thousands of highly sensitive users such as activists who are likely targets of surveillance. Likewise, many ordinary users and even governments and enterprises want to migrate from centralized silos that are easily compromised by programs such as PRISM. Therefore, it is critical that technical solutions be provided that work today with existing heavily-used protocols such as SMTP to combat surveillance.

[28] *http://nextleap.eu*

Chapter 6

End-user Usability

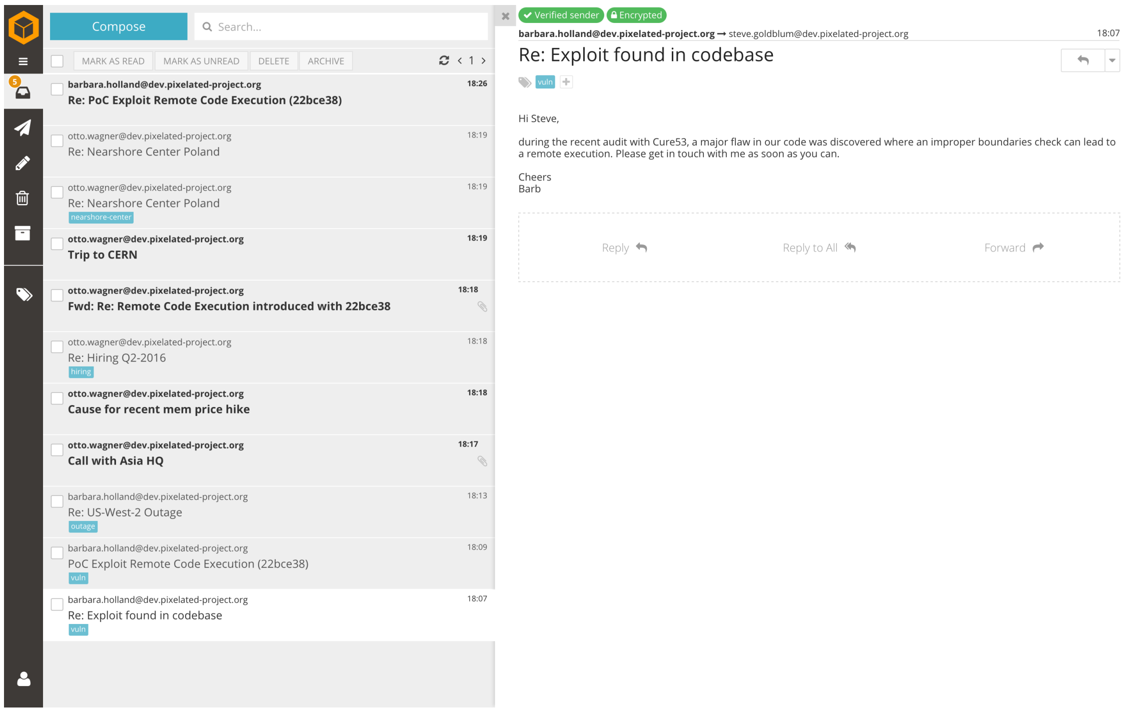
The goal of end-user usability should be to make PANORAMIX-enabled unmappability be “invisible” to the end-user. The only case it should be visible is if the security and privacy properties of the system are changed due to user action. The goal of the LEAP system is to enable encrypted email without the end-user having to know about encryption and keys. In the same vein, the user should have to know ultimately nothing about the mix networking or anonymity.

With the help of SimplySecure usability expert Gus Andrews[1] and working with Thoughtworks, there were three user tests of Pixelated, an encrypted, browser-based email client integrated with the open source LEAP codebase at the Internet Freedom Festival in March 2016. Participants were primarily European and North American adults and one African participant. The participants had used encrypted email before. An interface screenshot of Pixelated is given in Figure 6.1. Generally, users make sense of and navigate around the screen easily, even with slightly unusual features such as the compose window being located on the right. Users quickly orientate to and make sense of dialogs about attachment size limit. They also figure out the menu easily, and use both it and the logo to expand and collapse the left-hand column. Saving of drafts is working seamlessly when users navigate away from their draft.

The current interface sends a single message both encrypted and unencrypted, with no warnings: This sending of messages in a “mixed mode” is a security risk that will also affect the use of PANORAMIX-enabled mixed-networking. One test participant who trains journalists in digital security expressed alarm that Pixelated would allow the same message to be sent to one recipient encrypted and another recipient in the clear. “This seems completely insecure,” he said, noting that attackers could easily see the unencrypted message in transit. He expressed surprise that Pixelated had not stopped and notified him that it was sending a message in the clear. Further user tests should be done on whether users are noticing and responding to any mixed-mode notifications the Pixelated client is showing and whether other clients can be adapted to handle mixed-mode messages gracefully. User tests indicate that when you tell users an email system will encrypt and then give them the task of sending a message, they do not think about checking whether a message will send encrypted unless you stop them and ask them to look for cues about encryption. Pixelated could follow the Thunderbird plug-in Enigmail’s model: Enigmail does not permit users to hit “send” and transmit the same message encrypted and unencrypted at the same time.

When using a PANORAMIX-enabled mix network with email, the system becomes potentially even more complicated. In addition to having to worry about encryption, a user should know if their email is going to users that have PANORAMIX-enabled mix networking enabled and so are privacy-preserving or if their email is being sent to providers that are not PANORAMIX-enabled. This is further complicated by the use of encryption, as we can imagine that some recipients will have encryption but not PANORAMIX mix-networking enabled. The simplest modification would be to follow the example of the Bitmask VPN from the LEAP codebase as shown in Figure 6.2 and 6.3, and use a simple icon such as a “red/green” light when messages are being sent to one or more recipients that are not using PANORAMIX. A single light could be added to the client’s task bar interface, as typical with a VPN, in order to show that the traffic being sent preserves privacy. For emails, this kind of light could be added to the send box with a “red” lght activated when a recipient may not have PANORAMIX enabled (as detected via a membership node), although usability testing should be deployed to determine if these visual signals are noticed. Then a dialogue that explicitly

[1] *https://simplysecure.org/*



barbara.holland@dev.pixelated-project.org 18:26
Re: PoC Exploit Remote Code Execution (22bce38)

otto.wagner@dev.pixelated-project.org 18:19
Re: Nearshore Center Poland

otto.wagner@dev.pixelated-project.org 18:19
Re: Nearshore Center Poland
nearshore-center

otto.wagner@dev.pixelated-project.org 18:19
Trip to CERN

otto.wagner@dev.pixelated-project.org 18:18
Fwd: Re: Remote Code Execution introduced with 22bce38

otto.wagner@dev.pixelated-project.org 18:18
Re: Hiring Q2-2016
hiring

otto.wagner@dev.pixelated-project.org 18:18
Cause for recent mem price hike

otto.wagner@dev.pixelated-project.org 18:17
Call with Asia HQ

barbara.holland@dev.pixelated-project.org 18:13
Re: US-West-2 Outage
outage

barbara.holland@dev.pixelated-project.org 18:09
PoC Exploit Remote Code Execution (22bce38)
vuln

barbara.holland@dev.pixelated-project.org 18:07
Re: Exploit found in codebase
vuln

✓ Verified sender 🔒 Encrypted

barbara.holland@dev.pixelated-project.org → steve.goldblum@dev.pixelated-project.org 18:07

Re: Exploit found in codebase

vuln +

Hi Steve,

during the recent audit with Cure53, a major flaw in our code was discovered where an improper boundaries check can lead to a remote execution. Please get in touch with me as soon as you can.

Cheers
Barb

Figure 6.1: Pixelated Screenshot

notes which one or more of the senders or recipients could have their "identity revealed by receiving this email if they are being monitored" or some guiding text. The difficult usability challenge of the deployment of PANORAMIX is finding some text and visual icons that can adequately express to a user the privacy properties that mix-networking provides.

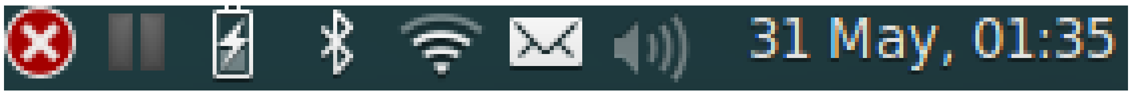


Figure 6.2: Bitmask "Red Light" Screenshot

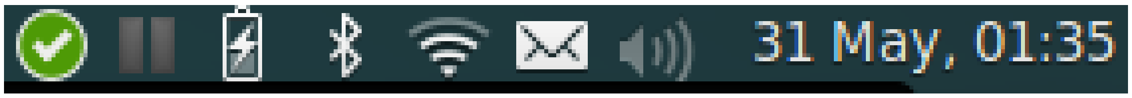
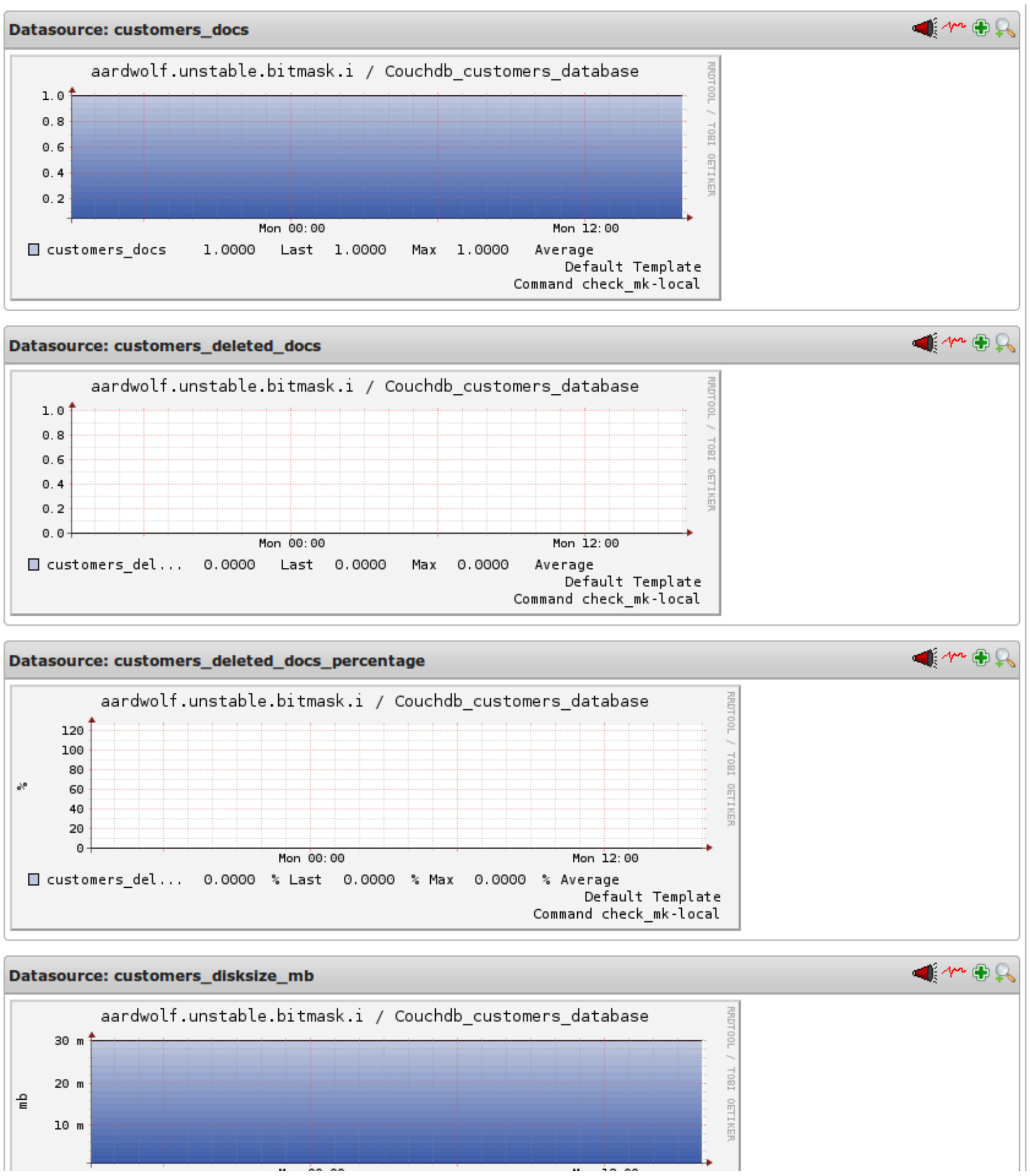


Figure 6.3: Bitmask "Green Light" Screenshot

Chapter 7

System Administration Usability

Currently, the LEAP codebase does not provide an easy-to-use interface for system administrators, relying instead on the use of the command prompt. Over the next year, we hope to provide an easy-to-use graphical interface to help system administrators understand and handle their LEAP provider, with a focus on understanding how the use of PANORAMIX-based mix networking is affecting their service. In particular, we'd like to add bandwidth consumed by cover traffic and the average delay caused by the mix network message shuffling in addition to system administrator statistics such as total number of online users, spam, uptime/downtime, diskspace contraints etc. The plan is to add these metrics to a LEAP-enabled Nagios interface. This interface is shown below in Figure 7.2. We would also like to add the state of the entire PANORAMIX network and its topology in Figure 7.1, including uptime and downtime of particular servers in the network that may be needed for mixing.



Datasource: customers_docs

aardwolf.unstable.bitmask.i / Couchdb_customers_database

```
□ customers_docs      1.0000   Last   1.0000   Max   1.0000   Average
                                                     Default Template
                                                 Command check_mk-local
```

Datasource: customers_deleted_docs

aardwolf.unstable.bitmask.i / Couchdb_customers_database

```
□ customers_del...    0.0000   Last   0.0000   Max   0.0000   Average
                                                     Default Template
                                                 Command check_mk-local
```

Datasource: customers_deleted_docs_percentage

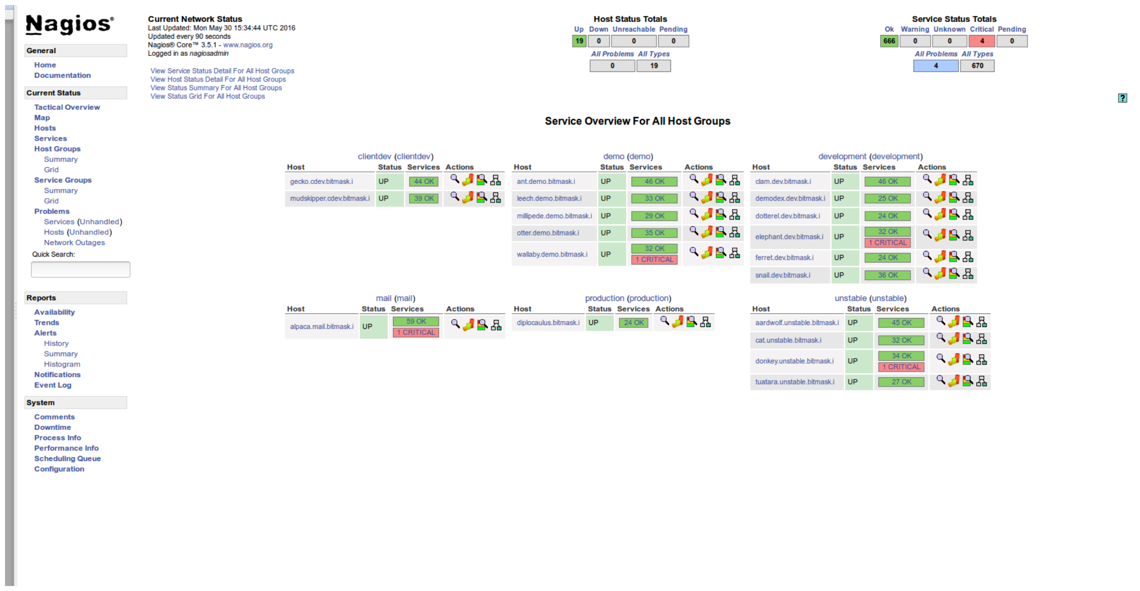
aardwolf.unstable.bitmask.i / Couchdb_customers_database

```
□ customers_del...    0.0000  % Last   0.0000  % Max   0.0000  % Average
                                                        Default Template
                                                    Command check_mk-local
```

Datasource: customers_disksize_mb

aardwolf.unstable.bitmask.i / Couchdb_customers_database

Figure 7.1: Nagios Network Overview Screenshot



Current Network Status
Last Updated: Mon May 30 15:34:44 UTC 2016
Updated every 90 seconds
Nagios® Core™ 3.5.1 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Host Status Detail For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
19	0	0	0

All Problems	*All Types*
0	19

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
666	0	0	4	0

All Problems	*All Types*
4	670

Service Overview For All Host Groups

clientdev (clientdev)

Host	Status	Services	Actions
gecko.cdev.bitmask.i	UP	44 OK	
mudskipper.cdev.bitmask.i	UP	39 OK	

demo (demo)

Host	Status	Services	Actions
ant.demo.bitmask.i	UP	46 OK	
leech.demo.bitmask.i	UP	33 OK	
millipede.demo.bitmask.i	UP	29 OK	
otter.demo.bitmask.i	UP	35 OK	
wallaby.demo.bitmask.i	UP	32 OK 1 CRITICAL	

development (development)

Host	Status	Services	Actions
clam.dev.bitmask.i	UP	46 OK	
demodex.dev.bitmask.i	UP	25 OK	
dotterel.dev.bitmask.i	UP	24 OK	
elephant.dev.bitmask.i	UP	32 OK 1 CRITICAL	
ferret.dev.bitmask.i	UP	24 OK	
snail.dev.bitmask.i	UP	36 OK	

mail (mail)

Host	Status	Services	Actions
alpaca.mail.bitmask.i	UP	59 OK 1 CRITICAL	

production (production)

Host	Status	Services	Actions
diplocaulus.bitmask.i	UP	24 OK	

unstable (unstable)

Host	Status	Services	Actions
aardwolf.unstable.bitmask.i	UP	45 OK	
cat.unstable.bitmask.i	UP	32 OK	
donkey.unstable.bitmask.i	UP	34 OK 1 CRITICAL	
tuatara.unstable.bitmask.i	UP	27 OK	

Figure 7.2: Nagios Network Properties Screenshot

Chapter 8

Empirical Mix-net Parameters

8.1 Empirical Data

In order to estimate the needed parameters for mix networking, statistics need to be collected in order to determine the amount of cover-traffic and other parameters of mix networking. The following needed to be collected:

- The date and time (with second accuracy) that the message was sent, and that the message was received.
- The size of the message.
- If the message is not text, their types and their sizes.
- Number of recipients.
- An identifier for the mail item.
- Whether the message is a reply and if so, the identifier of the message being replied to.
- Whether or not the mail was sent to a mailing-list or not.

4 hours of email traffic from a LEAP-enabled provider were logged. On the server, the information above was anonymized in terms of identifiers and contents, resulting in a 52 gigabytes of data in a SQL database for analysis. As part of the future deliverable or a revision of this one, we will provide the exact parameters needed to parameterize the mix network. However, due to the difficulty of doing the computations in a privacy-preserving manner, we will not release sensitive user data at this point in PANORAMIX. The statistics have been given to PANORAMIX project members University College London for analysis as part of their design in Workpackage 5.

8.2 Privacy-Preserving Data Analytics using Succint Sketches

As stated earlier, one of the hardest problems facing privacy-enhanced email is how collect statistics in a manner that preserves the privacy of the users. Based on a modification of work using succinct sketches to do privacy-aware aggregate statistics for Tor using PrivEx, PrivEx can provide statistics over LEAP-enabled providers in order to guarantee their anonymity. The following sections in this chapter are taken and modified of work on PrivEx presented by Melis, Danezis, and De Cristofaro at NDSS 2016[25].

In general, PrivEx interested in scenarios where providers need to train models based on aggregate statistics gathered from many data sources, and our goal is to do so without disclosing fine-grained information about single sources. In theory, we could turn to existing cryptographic protocols for privacy-friendly aggregation: using homomorphic encryption or secret sharing untrusted aggregators can collect encrypted readings but only decrypt the sum [5, 4, 23, 16, 28, 3]. However, these tools require each data source to perform a number of cryptographic operations, and transmit a number of ciphertexts, *linear* in the size of their input, which makes them impractical when sources contribute large streams.

Also, differential privacy could be used to let aggregators add noise to datasets so that other parties may perform statistical queries while the probability of identifying single records is minimized [8]. However, differential privacy alone would not protect the privacy of single data sources w.r.t. the aggregators themselves. Although recent work such as RAPPOR [17] supports, via input perturbation, differentially-private statistics collection with an untrusted aggregator, it actually requires millions of users in order to obtain reasonably accurate answers.

Our insight is to combine privacy-preserving aggregation with data structures supporting succinct data representation, namely, *Count-Min Sketch* [7] and *Count Sketch* [6] (introduced in Section 8.2.2). Private aggregation is performed over the sketches, rather than the raw inputs. Despite an upper-bounded error in the aggregate is introduced, this allows us to reduce communication and computational complexity (for the cryptographic operations) from *linear* to *logarithmic* in the size of the inputs. We implement them in JavaScript to support seamless web application deployment and portability to multiple browsers as well as Android.

On the other hand, our first-of-its-kind protocol for median statistics of Tor hidden services (and hopefully PANORAMIX-enabled LEAP servers) uses additively homomorphic threshold decryption, relying on a set of non-colluding authorities. It is developed in Python so that it can be integrated on Tor Hidden Service Directories. We also show how to integrate differential privacy guarantees by adding noise to leaked *intermediate* values during the median estimation process which does not involve non-linear operations.

8.2.1 Cryptographic Background

Computational Diffie Hellman Assumption. Let $\mathbb{G}$ be a cyclic group of order q ($|q| = \tau$, for security parameter τ), with generator g. We say that the Computational Diffie Hellman (CDH) problem is hard if, for any probabilistic polynomial-time algorithm $\mathcal{A}$ and random x, y drawn from $\mathbb{Z}_q$:

$$\Pr\left[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y) = g^{xy}\right]$$

is negligible in the security parameter τ.

Decisional Diffie Hellman Assumption. Let $\mathbb{G}$ be a cyclic group of order q ($|q| = \tau$), with generator g. We say that the Decisional Diffie Hellman (DDH) problem is hard if, for any probabilistic polynomial-time algorithm $\mathcal{A}'$ and random x, y, z drawn from $\mathbb{Z}_q$:

$$\left| \Pr\left[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^z) = 1\right] - \Pr\left[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1\right] \right|$$

is negligible in the security parameter τ.

Pairwise Independent Hash Functions. Let H be a family of *random-looking* hash functions mapping values from a domain [D] to a range [R]. H is *pairwise independent* iff $\forall x \neq y \in [D]$ and $\forall a_1, a_2 \in [R]$: $\Pr_{h \in H}\left[h(x) = a_1 \wedge h(y) = a_2\right] = \frac{1}{R^2}$.

8.2.2 Count-Min Sketch and Count Sketch

Count-Min Sketch [7] is a data structure that can be used to provide a succinct sublinear-space representation of multi-sets. An interesting property is that they enable aggregation of the multi-sets represented by two or more sketches using a linear operation on the sketches themselves. Prior uses of Count-Min Sketch include summarizing large amounts of frequency data for sensing, networking, natural language processing, and database applications [1].

Definition 1 (*Count-Min Sketch*)**.** *A Count-Min Sketch with parameters (ϵ, δ) is a two-dimensional array (table) X, with width w and depth d. Given parameters (ϵ, δ), set $d = \lceil \ln T/\delta \rceil$ and $w = \lceil e/\epsilon \rceil$, where T is the number of items to be counted. Each entry of the table is initialized to zero. Then, d hash functions $h_j : \{0,1\}^* \rightarrow \{0,1\}^w$, are chosen uniformly at random from a pairwise-independent family $\mathcal{H}$.*

Update Procedure. To update item i by a quantity c_i, c_i is added to one element in each row, where the element in row j is determined by the hash function h_j. The update is denoted as (i, c_i). More precisely, to update the count for item i to $c_i \in \mathbb{N}$, for each row j of X, set:

$$X[j, h_j(i)] \leftarrow X[j, h_j(i)] + c_i$$

Estimation Procedure. To estimate the count $\hat{c_i}$ for item i, we take the minimum of the estimates of c_i from every row of X:

$$\hat{c_i} \leftarrow \min_j X[j, h_j(i)]$$

Error Upper Bound. Given estimate $\hat{c_i}$, it holds:

1. $c_i \leq \hat{c_i}$
2. $\hat{c_i} \leq c_i + \epsilon \sum_{j=1}^{T} |c_j|$ with probability $1 - \delta$.

(where c_i is the true counter).

Count Sketch [6] is a data structure which provides an estimate for an item's frequency in a stream. Count Sketch has the same update procedure as Count-Min Sketch, but differs in the estimation. Specifically, given the table X built on the stream, the row estimate of c_i (which is the counter of item i) for row j is computed based on two buckets: $X[i, h_j(i)]$ and $X[i, h'_j(i)]$, where $h'_j(i)$ is defined as:

$$h'_j(i) := \begin{cases} h_j(i) - 1 & \text{if } h_j(i) \bmod 2 = 0 \\ h_j(i) + 1 & \text{if } h_j(i) \bmod 2 = 1 \end{cases}$$

The estimate of c_i for row j is then

$$\left(X[j, h_j(i)] - X[j, h'_j(i)]\right)$$

To estimate the count $\hat{c_i}$ for item i, we take the median of the estimates of c_i from every row of X:

$$\hat{c_i} \leftarrow \underset{j}{median} \left(X[j, h_j(i)] - X[j, h'_j(i)]\right)$$

Both Count-Min and Count Sketch are linear: the element-wise sum of the sketches representing two multi-sets yields the sketch of their union.

8.2.3 Differential Privacy

Differentially private mechanisms allow a party publishing a dataset to make sure that only a bounded amount of information is leaked. Output perturbation mechanisms modify a statistic on a dataset D, prior to its release, using a randomized algorithm A, so that the output of A does not reveal too much information about any particular row in D.

Definition 2 (*ϵ-Differential privacy* [13])**.** *A randomized algorithm A satisfies ϵ-differential privacy, if for any two neighbor datasets D_1 and D_2 that differ only in one row, and for any possible output R of A, it holds:*

$$\Pr[A(D_1) = R] \leq e^{\epsilon} \cdot \Pr[A(D_2) = R]$$

Note that ϵ here is used differently than in the Count-Min Sketch's definition. Although this somewhat overloads the notation for ϵ, it is actually clear from the context if it relates to the data structure or to the differential privacy setting.

Laplace Mechanism. To study Tor, we use the differentially private Laplace mechanism [14], which perturbs the output of a function F. Given F, the Laplace mechanism transforms F into a differentially private algorithm, by adding independent and identically distributed (i.i.d.) noise (denoted as η) into each output value of F. The noise η is sampled from a Laplace distribution $Lap(\lambda)$ with the following probability density function: $Pr[\eta = x] = \frac{1}{2\lambda} e^{|x|\lambda}$. Dwork [13] proves that the Laplace mechanism ensures ϵ-differential privacy if $\lambda \geq \frac{S(F)}{\epsilon}$, with $S(F)$ denoting the sensitivity of F, defined as:

$$S(F) = \max_{D_1, D_2} ||F(D_1) - F(D_2)||_1$$

where $||\cdot||_1$ denotes the L1 norm, and D_1 and D_2 are any two neighbor datasets. Intuitively, $S(F)$ measures the maximum possible change in F's output when we modify one arbitrary row in F's input.

8.2.4 Implementation and Evaluation

We implement and evaluate the proposed scheme aiming to: (i) estimate the trade-off between size of the sketch and the accuracy of the median computation, (ii) evaluate the cost of cryptographic computation and communication overheads, and (iii) assess the trade-off between the accuracy of the median and the quality of protection that may be achieved through the differentially private mechanism.

For our evaluation, we instantiate AH-ECC using the NIST-P224 curve as provided by the OpenSSL library and its optimizations by Käsper [22]. Our implementation of the cryptographic core of the private median scheme amounts to 300 lines of Python code using the *petlib* OpenSSL wrapper[1], and another 350 lines of Python include unit tests and measurement code. All experiments have been performed on a Xubuntu Trusty (Ubuntu 14.04.2 LTS) Linux VM, running on a 64 bit Windows 7 host (CPU i7-4700MQ, 2.4Ghz, 16GB RAM). Our Python implementation is easily pluggable as part of the Tor infrastructure and does not require changes within the Tor (C-based) core functionalities.

We first illustrate the performance and accuracy of estimating the median using this technique with both sketch parameters ϵ and δ equal to either 0.25 or 0.05 against the London Atlas Dataset[2]. The error rate is computed as the absolute value of difference between the estimated and true median divided by the true median.

Further results are presented on an experimental setup that uses as a reference problem the median estimation in a set of 1,200 sample values, drawn from a mixture distribution: 1,000 values from a Normal distribution with mean 300 and variance 25, and 200 values drawn from a Normal distribution with mean 500 and variance 200. This reference problem closely matches the settings of both the Tor project and, we project, PANORAMIX-enabled email both in terms of the range of vales (assumed to be within $[0, 1000]$) and the number of samples [15].

Quality vs. Size. Figure 8.1 illustrates the trade-off between the quality of the estimation of the median algorithm and the size overhead of the Count Sketch. The size overhead (green slim line) is computed as the number of encrypted elements in the sketch as compared with the number of elements in the range of the median (1,000 for our reference problem). The estimation accuracy (blue broader line) is represented as the fraction of the absolute deviation of the estimate from the real value over the real sample median (light blue region represents the standard deviation of the mean over 40 experiments for each datapoint). Thus both qualities can be represented as percentages.

The trade off between the size of the sketch and the accuracy of the estimate is evident: as the sketch size reaches a smaller fraction of the total possible number of values, the error becomes larger than the range of the median. Thus, Count Sketch with parameters $\epsilon, \delta < 0.025$ are unnecessary, since they do not lead to a reduction of the information that needs to be transmitted from each client to the authorities; conversely, for $0.15 < \epsilon, \delta$ the estimate of the median deviates by more than 20% of its true value making it highly unreliable.

For all subsequent experiments, we consider a Count Sketch with values $\epsilon = \delta = 0.05$, leading to $d = 3$ and $w = 55$. As outlined in Figure 8.1, this represents a good trade-off between the size of the Count Sketch (16.5% of transmitting all values) and the error.

True Size and Performance. When implemented using NIST-P224 curves, the reference Count Sketch may be serialized in 10,898 bytes. Each Count Sketch takes 0.001 sec to encrypt at each HSDir, and it takes 1.456 seconds to aggregate 1,200 sketches at each authority (0.001 sec per sketch). As expected, from the range of the reference problem, 10 decryption iterations are sufficient to converge to the median (therefore $\xi = 10$). The number of homomorphic

[1]`https://github.com/gdanezis/petlib`
[2]`http://data.london.gov.uk/dataset/ward-profiles-and-atlas`

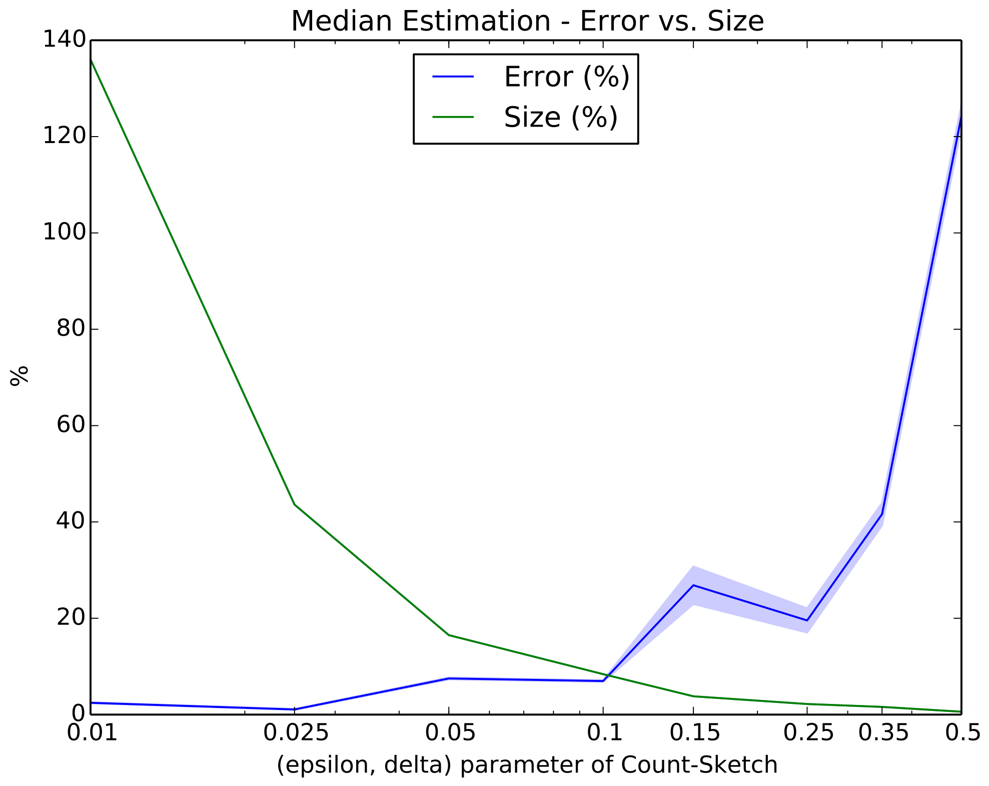


Figure 8.1: Count Sketch size versus estimation quality.

additions for each decryption round is linear in the range of the median and their total computational cost is the same order of magnitude as a full Count Sketch encryption. It is clear from these figures that the computational overhead of the proposed technique is eminently practical, and the bandwidth overhead acceptable.

Quality vs. Differential Privacy Protection. Figure 8.2 illustrates the trade-off between the quality of the median estimation and the quality of differential privacy protection. The x-axis represents the ϵ parameter of the differentially private system, and the y-axis the absolute error between the estimate and the true sample median. Differential privacy with parameter $\epsilon = 0.5$ can be provided without significantly affecting the quality of the median estimate. However, for $\epsilon < 0.5$ the volume of the error grows exponentially (note the log scale of the x-axis). While the exact value of a meaningful ϵ parameter is often debated in the literature, we conclude that the mechanism only provides a limited degree of protection, and no ability to readily tune up protection: utility degrades very rapidly as the security parameter ϵ decreases.

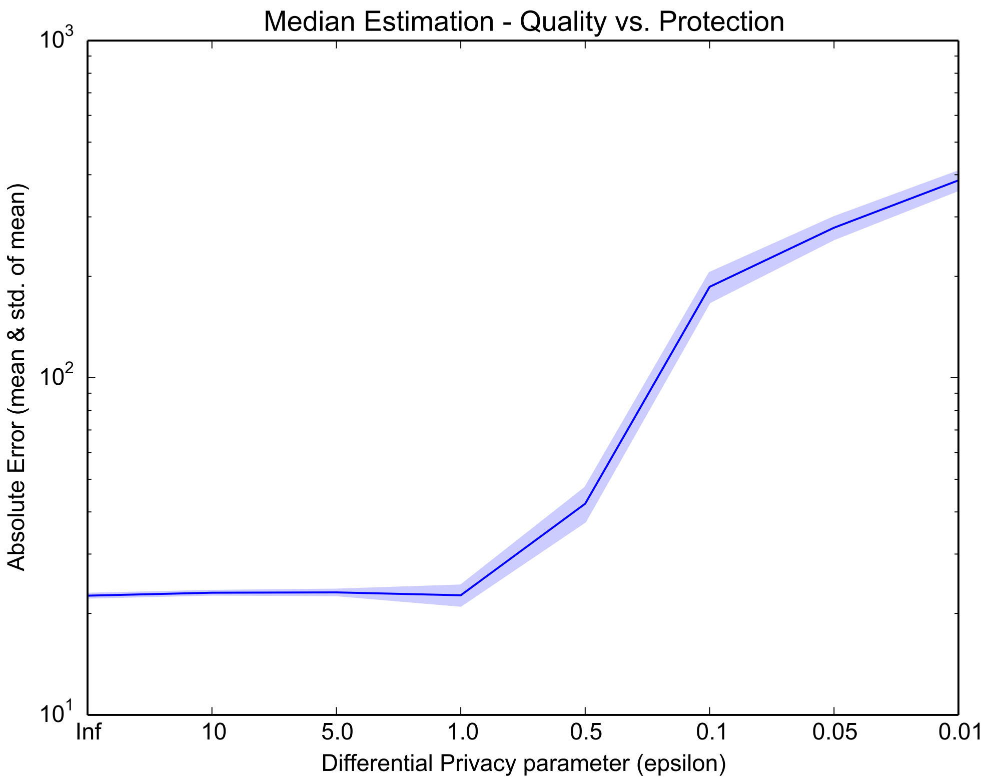


Figure 8.2: Quality versus differential privacy protection.

Chapter 9

Anonymity Analysis

There are several design decisions in how we implement mix networking and cover traffic for LEAP-enabled email that have an effect on the privacy properties. Although a full anonymity analysis will only be possible over simulated systems parameterized by the statistics gathered over email, a number of general principles are observable.

An envisioned mix networking system is sketched in Figure 9.1. This shows a message from Sender 2 out of n senders through their provider, who serves as an input node into a PANORAMIX-enabled mix network that delivers the message with third-party anonymity to Recipient 1. After transferring between various mix nodes, the message is output into one of y providers. Each provider then has a number of different recipients, with a total of m recipients of the message. Quite crucially, the anonymity set of a typical e-voting assumes $n = m$. However, we can see that the input n and the output m vary. Worse, m itself may be subdivided into y uneven sets due to differences between providers. Even assuming that an attacker cannot distinguish between y, so the anonymity set for a given message is then $M = \{1, ...m\}$ for each time period. Yet as multiple messages are sent and z recipient clients may leave the system such that $Z = \{1, ...z\}$, the anonymity set will decrease such that the anonymity set is $M \setminus Z$. So the central problem of messaging mix networking is how to deal with the anonymity damage loss of recipients cause. Furthermore, by adding that senders are also going offline using an information-theoretic analysis, it is clear that the real-world messaging use-case will have to make new design decisions in order to preserve anonymity in contrast to systems like [30] and e-voting systems.

1. As traffic analysis can be done over both message size, message frequency, and message time. PANORAMIX should employ constant-size messages with padding as needed, dividing a single email into multiple message as necessary in order to achieve constant size. PANORAMIX should employ cover traffic in order to disguise message frequency if needed. PANORAMIX should use mix networking to deliver messages over constant time intervals. An analysis of real-world network turbulence should be taken into account as regards anonymity as well.

2. One way to violate the privacy of users via traffic analysis is to intercept the messages delivered between MTAs. At minimum, the above be done to provide provider unlinkability from passive adversaries.

3. A local passive observer can monitor all the incoming connections to a provider MSA and all outgoing connections from the MDA. In order to deal with attacks on the client, if client software is used we should employ the same anti-traffic analysis techniques to provide unlinkability of a user and their provider.

4. Simply re-purposing an existing e-voting mix networking system would have email providers moving their traffic in and out of a static mix networking system. However, as denial-of-service attacks could disable the connection between MTAs and a static mix networking system, it would be preferable to integrate the mix networking into the email servers themselves, with other mix net servers used in other services (such as e-voting) being indistinguishable from MTAs to a third-party.

5. Provider reputation is necessary in dynamic mix networking systems. For users that already have established

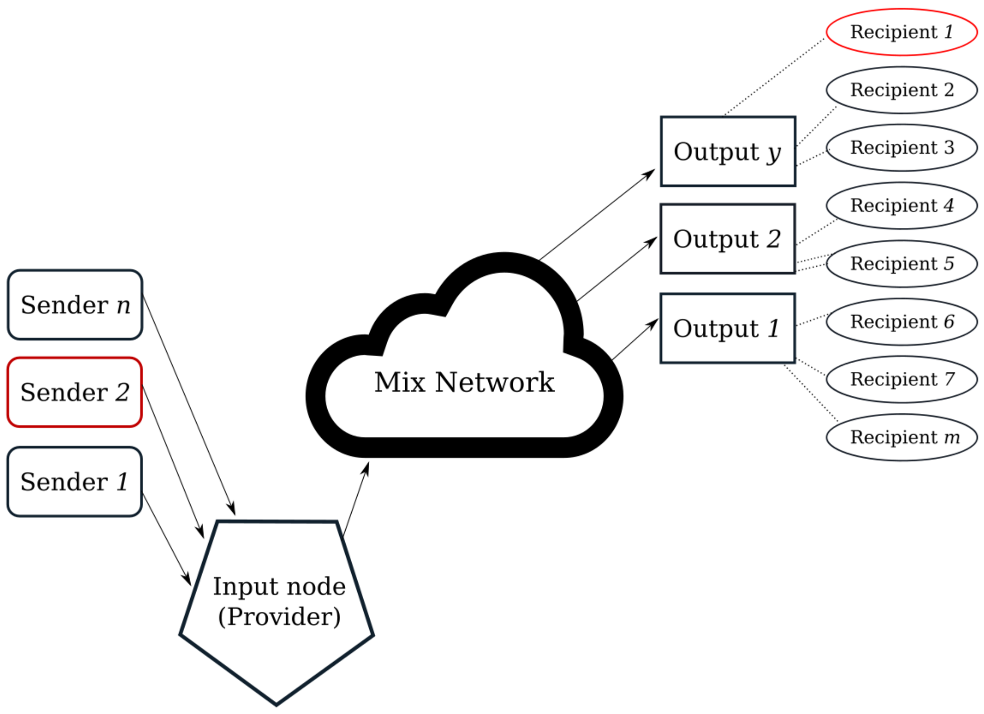


Figure 9.1: Anonymity Set Example

trust, pre-authorization is preferable but this is an advanced option that should be pursued later after MORBLs and other reputation systems are tried.

6. In order to mitigate abusive users, the first version of PANORAMIX should primarily provider third-party anonymity. Recipient and sender anonymity as well as protection from active attackers on the server or malicious servers should be done only after third-party anonymity is achieved as they are more difficult and could lead to issues regarding spam and abuse.

7. Users should have a simple interface to distinguish between emails that are privacy-preserving and those that are not, such as a "green/red" light interface.

8. Latency must be acceptable to users.

9. How interaction with open email providers that do not use PANORAMIX damage anonymity must be studied further.

10. Previous mix networking systems assume clients are always connected [30]. In email systems, clients are not always connected and so this assumption may cause a loss of privacy for users, but is likely impossible to prevent. It is precisely this property of messaging that causes major issues with traditional mix networking systems and messaging, and effects the anonymity set.

Bibliography

[1] Count-Min Sketch and its applications. `https://sites.google.com/site/countminsketch/`, 2015.

[2] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, et al. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17. ACM, 2015.

[3] I. Bilogrevic, J. Freudiger, E. De Cristofaro, and E. Uzun. What's the Gist? Privacy-Preserving Aggregation of User Profiles. In *ESORICS*, 2014.

[4] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Mobiquitous*, 2005.

[5] T.-H. H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *FC*, 2012.

[6] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *ICALP*, 2002.

[7] G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *Journal of Algorithms*, 2005.

[8] G. Cormode, C. Procopiuc, D. Srivastava, and T. T. Tran. Differentially private summaries for sparse data. In *ICDT*, 2012.

[9] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *IEEE Symposium on Security and Privacy*, pages 2–15. IEEE Computer Society, 2003.

[10] G. Danezis and A. Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *in Proceedings of 6th Information Hiding Workshop (IH 2004*, pages 293–308, 2004.

[11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation Onion Router. Technical report, DTIC Document, 2004.

[12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. *Proceedings of the 13th USENIX Security Symposium*, 2, 2004.

[13] C. Dwork. Differential Privacy. In *ICALP*, 2006.

[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, 2006.

[15] T. Elahi, G. Danezis, and I. Goldberg. PrivEx: Private Collection of Traffic Statistics for Anonymous Communication Networks. In *ACM CCS*, 2014.

[16] Z. Erkin and G. Tsudik. Private Computation of Spatial and Temporal Power Consumption with Smart Meters. In *ACNS*, 2012.

[17] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *ACM CCS*, 2014.

[18] H. Fakhoury. Fbi overreaches with may first - riseup server seizure. InfoSec Island, 2012. http://www.infosecisland.com/blogview/21186-FBI-Overreaches-with-May-First-Riseup-Server-Seizure.html.

[19] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko. Security by any other name: On the effectiveness of provider based email security. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 450–464. ACM, 2015.

[20] S. L. Garfinkel. Enabling email confidentiality through the use of opportunistic encryption. In *Proceedings of the 2003 annual national conference on Digital government research*, dg.o '03, pages 1–4. Digital Government Society of North America, 2003.

[21] S. Gaw, E. W. Felten, and P. Fernandez-Kelly. Secrecy, flagging, and paranoia: adoption criteria in encrypted email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 591–600, New York, NY, USA, 2006. ACM.

[22] E. Käsper. Fast Elliptic Curve Cryptography in OpenSSL. In *FC*, 2012.

[23] K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly Aggregation for the Smart-grid. In *PETS*, 2011.

[24] M. S. Melara, A. Blankstein, J. Bonneau, M. J. Freedman, and E. W. Felten. Coniks: A privacy-preserving consistent key service for secure end-to-end communication. *IACR Cryptology ePrint Archive*, 2014:1004, 2014.

[25] L. Melis, G. Danezis, and E. De Cristofaro. Efficient private statistics with succinct sketches. In *Isoc NDSS*, 2016.

[26] A. Narayanan, V. Toubiana, S. Barocas, H. Nissenbaum, and D. Boneh. A critical look at decentralized personal data architectures. *CoRR*, abs/1202.4503, 2012.

[27] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine. Survivable key compromise in software update systems. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 61–72, New York, NY, USA, 2010. ACM.

[28] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-Preserving Aggregation of Time-Series Data. In *NDSS*, 2011.

[29] E. Sparrow, H. Halpin, K. Kaneko, and R. Pollan. LEAP: A next-generation client VPN and encrypted email provider. In *Workshop on Surveillance and Technology*, 2015.

[30] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[31] A. Whitten and J. D. Tygar. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8*, SSYM'99, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.