Acta Cybernetica **27** (2025) 93–112.

Scalix Mix Network*

Ádám Vécsi[ab] and Attila Pethő[ac]

Abstract

Mix networks have now advanced to a level where they can compete in the domain of low-latency anonymous communication, owing to their "strong" anonymity design advantage. However, a few bottlenecks still exist, primarily because users are required to select the complete message path. This characteristic of mix networks hinders the implementation of load balancing and necessitates the use of an extensive shared database. To address this issue, we introduce Scalix, which presents a new topology featuring load balancers, modified path selection, and a mix package based on identity-based encryption and attribute-based encryption. Additionally, Scalix can serve as an anonymous return channel with minimal modifications.

Keywords: identity-based cryptography, attribute-based cryptography, mix network, anonymity

1 Introduction

When we talk about secure communication, we often mean encrypting messages with a cryptographic method to protect sensitive information. But communication also involves metadata, which can be valuable to observers. Metadata is information about the communication itself, such as who sent it, when it was sent, and where it was sent from. In this work, we focus on protecting the identities of the parties involved in communication by providing anonymity through a mix network.

The most well-known tool for anonymous communication is Tor [10], which uses onion routing to provide low-latency anonymous communication. However, this system only provides anonymity when the adversary can only observe a part of the system and not the entire network. If the messages go through servers that are not observed, the communication will be anonymous. VPN services are another popular solution, but they provide only superficial anonymity that hides the user's IP address, geolocation, and identity from requested sites and the ISP.

*The research was supported by the 2018-1.2.1-NKP-2018-00004 Security Enhancing Technologies for the Internet of Things project.

[a]Department of Computer Science, Faculty of Informatics, University of Debrecen, Hungary
[b]E-mail: vecsi.adam@inf.unideb.hu, ORCID: 0009-0003-5813-6111
[c]E-mail: petho.attila@inf.unideb.hu, ORCID: 0000-0002-9764-1570

DOI: 10.14232/actacyb.307671

For anonymity against a strong adversary that can observe the entire network, mix-based architectures are the best choice. Such architectures are useful in applications like voting, exam and assessment systems [14, 18, 13]. A mix network is a system that includes multiple stages of mixes, where every stage receives multiple messages, performs some cryptographic transformation for each message, and permutes them. After every mix, tracking the path of the messages gets more complex, achieving untraceability. However, mix networks used to have a trade off of high latency in communication. This issue seems to be improving rapidly thanks to Loopix [17], which is built on the Sphinx [7] mix format and can achieve low latency in communication, narrowing the delay to milliseconds.

Despite the promising benchmarks for Loopix, we believe there are two bottlenecks of the protocol. The first aspect involves load balancing among the nodes in each stage, which relies on theoretical random generation. This approach can lead to the possibility of an overloaded node if it is selected more frequently, or an underloaded node if it is chosen less frequently. The system lacks control over this outcome. The second is that the sender must pick the full path of the message, which requires a shared database with information about every mix node and could become costly to maintain with many mix nodes.

Our protocol addresses these issues by applying identity-based cryptography (IBC) and attribute-based cryptography (ABC) methods. In IBC, the public key is a known identifier string of an entity, such as an email address or phone number. ABC holds similar novelties but with even more flexible keys, allowing fine-grained cryptographic access control. Despite using IBC, our protocol allows the use-case as an anonymous return channel, enabling senders to communicate with a receiver in a way that the receiver will not know the sender's identity but can still reply to the messages. For more details, we refer to Golle and Jakobsson [11].

Our architecture aims to provide low-latency anonymous communication for users against adversaries who can monitor the entire network. We also aim to improve efficient scalability with reduced data sharing and avoid possible bottlenecks by implementing load balancers in the system. Our goal is to keep all the anonymity properties of Loopix. The sender-receiver third-party unlinkability ensures that it is impossible for an adversary to connect the honest sender of a message to its honest receiver. Sender online unobservability means that an adversary can't decide if an online honest sender is communicating with any receiver or not. Meanwhile, receiver unobservability means the inability of an adversary to decide if any sender is communicating with a specific online or offline honest receiver.

The rest of the paper is organized as follows. Section 2 introduces the core building blocks of our system, which are identity-based cryptography and attribute-based cryptography, and presents the main idea of mix networks. The related work is found in Section 3, which mainly focuses on Loopix, a mix network that provided a good foundation for our system. Section 4 presents the details of our construction, and finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Mix network

Mix networks are designed to provide strong anonymity for communicating parties by performing multistage cryptographic transformations and permutations on messages called mixing. This mix operation changes the appearance of messages and their order of transmission, making it difficult for even a strong adversary who can observe all communications in the network to trace messages.

The first mixnet was created by Chaum [6], using a single cascade of n mix nodes where each message was sealed multiple times, addressing each mix node and the receiver of the communication. While this system was able to provide anonymous emailing for those who prioritize anonymity, it suffered from high latency due to the required permutation on each node and the need to wait for a batch of messages before forwarding them. Additionally, the design was disadvantageous since a single malfunctioning node could cause the entire service to stop working. These shortcomings led to the emergence of other anonymity protocols like Tor [10], which provided low-latency with slightly weaker anonymity.

However, recent research has made mixnets more promising. There have been advancements in latency and fault tolerance, with different types of topologies and mix strategies. Stratified topology is the most widespread topology used in mixnet research due to its scalability and fault tolerance. As Figure 1 shows, it is organized into layers where nodes communicate with the nodes in the next layer, allowing horizontal scaling of the system if message traffic requires it. Diaz, Murdoch, and Troncoso's work [9] provides a clear analysis of mixnet topologies and suggests that stratified and restricted stratified topologies are the best choices for balancing anonymity and latency tradeoffs.

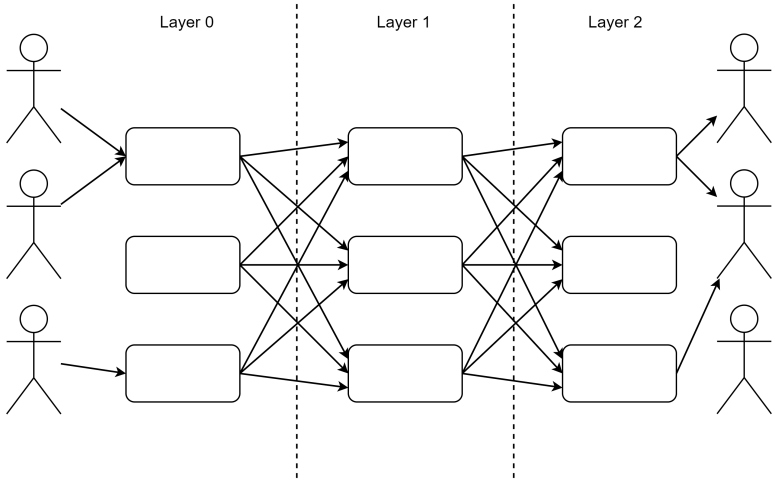

Figure 1: Stratified topology

The other area of mixnet research is mix strategies for nodes. There are various approaches to this topic, including the use of a message threshold, which requires nodes to wait until they receive a certain number of messages before forwarding them. However, this approach may not be suitable for low-latency networks as it could result in bottlenecks if nodes do not receive enough messages in time. Another approach, which had practical use too [21], is the message pool, which requires nodes to wait until they have a certain number of messages in their pool before performing permutation and then with a certain probability forwarding them to the next node. This could also lead to high latency if a message is unlucky and doesn't get forwarded. There are other ways to perform mixing [16], but the Stop and Go strategy [15] is the most promising for low-latency mix networks. In this strategy, the sender determines the amount of delay for each mix node before releasing the message, ensuring low-latency while still providing anonymity. However, if a node receives only one message, it won't strengthen anonymity. This issue can be addressed by using cover messages in the correct manner.

Overall, mix networks continue to evolve and improve, with new advancements being made in both topology and mix strategies.

2.2 Identity-based Cryptography

The original concept behind IBC was coined by Shamir in 1984 [20], who managed to build an identity-based signature scheme based on factorization. However, identity-based encryption (IBE) remained an unsolved problem until Boneh, and Franklin created their pairing-based scheme in 2001 [4], providing feasible performance for practical use.

The uniqueness of IBC lies in the fact that its public key is a string that identifies an entity in a particular domain. One may think about an email address, a username, or a phone number. This novelty directly connects with the core idea of the IBC, which was to simplify certificate management and eliminate the need for certification authorities. In the public key infrastructure scenario, public keys and user identities are bound together with certificates. With IBC, however, there is no need for such certificates since the public key corresponds directly to the user identity.

Furthermore, the public key may contain more information than just the identity of the user. This extension of the public key with domain-specific data enables a wide spectrum of advanced use cases where fine-grained access control is necessary.

Since this protocol family eliminates the need for certification authorities, it requires a trusted third party responsible for the user key generation, called the private key generator (PKG). The PKG responds to every extraction request based on the user identity, system parameters, and the master secret. Typically, in the IBC model, only the PKG knows the master secret. Otherwise, all user's private keys would be in danger. A reference implementation in C and WebAssembly can be found in [22].

2.3 Attribute-based cryptography

Attribute-based cryptography (ABC) is an extension of the idea of fuzzy identity-based cryptography [19] that provides a solution for fine-grained access control through the use of access structures. These structures allow the use of logical operators between attributes to define the authorization policy, such as (("Public Corruption Office" AND ("Knoxville" OR "San Francisco")) OR (management-level $>$ 5) OR "Name: Charlie Eppes"). ABC allows the targeting of a specific group of people with cryptographic access policies.

There are two types of ABC schemes: key-policy [12], in which the policy is associated with the user's secret keys and the ciphertexts with sets of descriptive attributes, and ciphertext-policy [3], in which the secret key is associated with sets of descriptive attributes and the ciphers with the policies. Ciphertext-policy provides the encryptor with control over who can access the data, while key-policy requires the encryptor to trust that the key-issuer has issued appropriate keys.

ABC also solves the problem of the encryptor not knowing the exact identities of all the people who should access the data. With ciphertext-policy attribute-based encryption, identities that the encryptor knows can be included, and the rest of the permitted people can be given by the attribute policy.

The key generation for ABC is similar to that of identity-based cryptography. As it can be seen on Figure 2, a trusted central authority creates decryption keys for users using a master secret key and additional descriptive information. With these keys, users can decrypt the ciphertexts if their attributes satisfy the policy associated with the ciphertext.

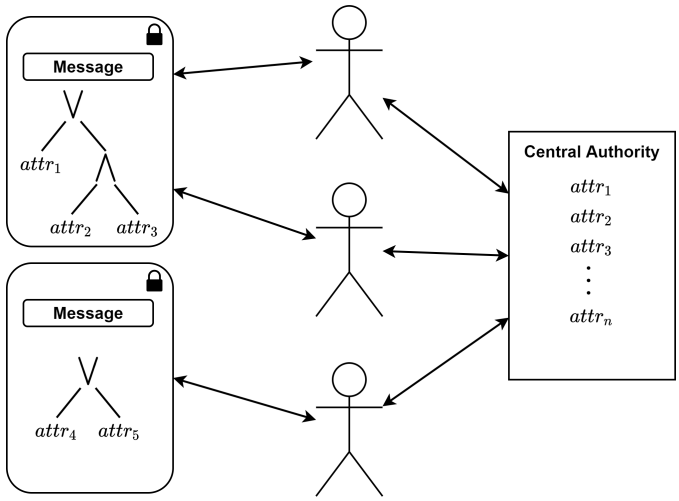

Figure 2: Ciphertext-policy attribute-based encryption

Multi-authority attribute-based encryption

Traditional attribute-based cryptography suffers from several flaws when it comes to the design of a single attribute authority. This can cause a bottleneck in terms of performance, as a single authority has to provide service to all the requests. Additionally, having a single authority creates a security risk, as one trusted party is responsible for all decryption key generation, which leads to the key escrow problem and requires a lot of trust.

To address these issues, Chase introduced a solution, multi-authority attribute-based encryption (MA-ABE) [5]. This concept allows for distributed management of attributes with independent authorities, and any number of authorities can be utilized, even if some of them are corrupt.

By increasing the number of authorities, the performance issue can be resolved since they can operate simultaneously, as depicted in Figure 3. Moreover, if the authorities are covering different parts of the attribute universe, the security will be increased, as the attributes are managed in a distributed way between authorities.

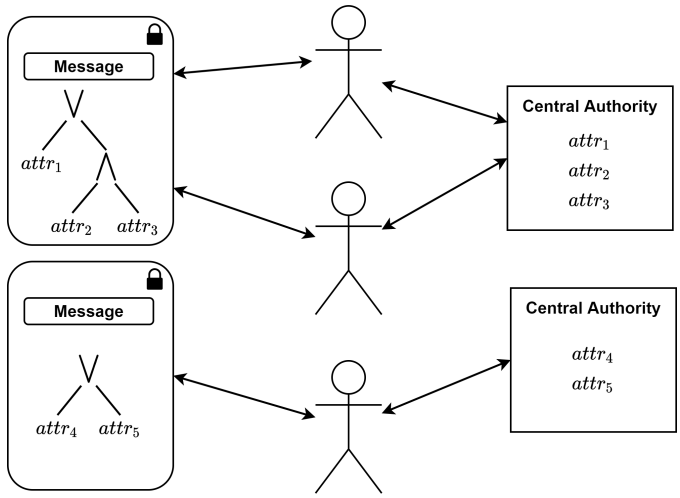

Figure 3: Multi-authority ciphertext-policy attribute-based encryption

3 Related work

3.1 Loopix

Loopix [17] is an academic mixnet that focuses on improving low-latency mix networks. It introduces several innovative features that, when combined, result in a low-latency solution for mix networks.

Loopix is built on a stratified topology, which includes a new entity called Provider. The idea of Providers comes from real-world messaging, where service

providers act as intermediaries between end-users. Similarly, all messages in Loopix go through Providers, which also act as storage for messages when end-users are offline. Providers are in a long-term relationship with users, and one provider can serve multiple users. The Provider layer is in connection with all the mix nodes from the first and the last layer to send and receive messages. The topology of Loopix can be seen in Figure 4.

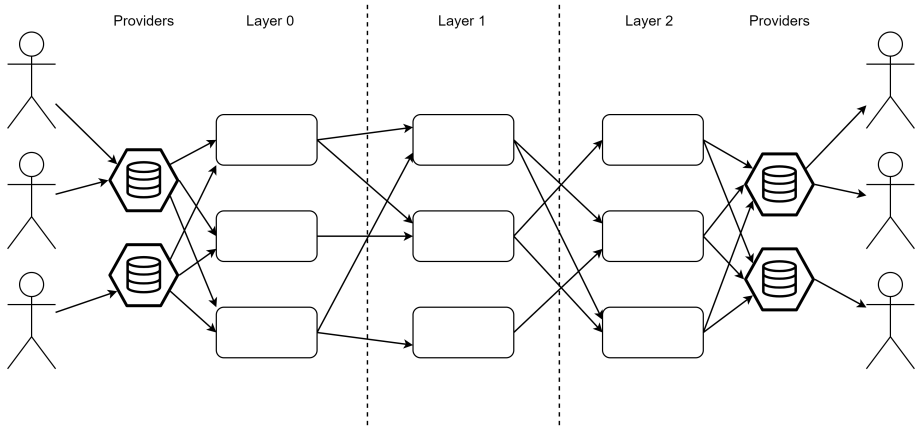

Figure 4: Loopix's topology

Mix networks' security and the latency of their messages are strongly linked, as all mix strategies depend on the number of messages flowing through the system. Loopix generates cover traffic to ensure message delivery and prevent bottlenecks of mix strategies. It includes three categories of cover traffic.

First, we need to understand how the message sending works in the system. When a user wants to send an anonymous message, they must place it into a buffer. The system periodically checks this buffer and sends any message it contains. If the buffer is empty, the system generates the first category of cover traffic and sends it to the user's Provider. This cover traffic looks like a real message, so it goes through the mixnet to a random Provider, which then discards it. With this periodic message and cover traffic, the system hides the information when the users are communicating, as real messages and this type of cover traffic are indistinguishable.

Users also send a second type of cover message that loops back to them, specifying themselves as the recipient. This category of messages provides cover for users when they receive real messages.

Finally, each mix node injects its own loop cover traffic that goes through the system to a random Provider and back to the mix node itself.

With these guaranteed messages, Loopix can use a simplified form of the Stop and Go mix strategy, which greatly contributes to low-latency communication. It is called Poisson mix, named after the fact that users and mixes send messages and cover traffic according to a Poisson process. Additionally, messages are indepen-

dently delayed using an exponential distribution. These factors make the system modelable in steady states with a Poisson distribution, allowing us to calculate the number of messages mixed together at any time.

With these innovations and the goal of achieving low-latency mix networks, Loopix has influenced multiple current mixnet solutions, such as Katzenpost [2], Nym [8], and Scalix (our system).

4 Scalix

Many mixnet protocols rely on maintaining a database of information about mix nodes, which allows message senders to select the route of their message. However, as the number of mix nodes scales, the size of the database required to store this information also increases, making it more costly to maintain. Additionally, scaling the number of users results in more queries on this database, making maintenance even more challenging and expensive. To address this issue, our model utilizes identity-based (IB) and attribute-based (AB) methods to provide a solution.

4.1 Topology

Loopix gave us many great ideas and a working low-latency mixnet. Our aim was to build upon this existing solution and extend it with our ideas. As seen in Figure 5 scalix's topology is similar to that of Loopix. However, we included load balancers (marked as $AA + LB$) between each layer of mix nodes, to guarantee the proper distribution of messages between mix nodes herewith maximizing the permeability.

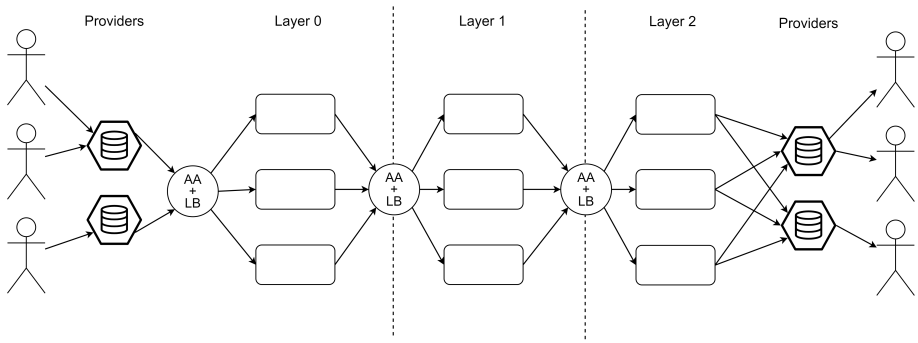

Figure 5: Scalix's topology

Attribute authority and load balancer nodes

Figure 5 depicts nodes labeled as $AA + LB$ between each mix layer, which serve two purposes in the system - attribute authority (AA) and load balancer (LB).

In our system, users target the $AA + LB$ nodes instead of individual mix nodes to determine the path through which their packets will be mixed. This ensures that packets always pass through an $AA + LB$ node before entering the mix layer, which is responsible for selecting the mix node to receive the packet based on load balancing methodologies.

The attribute authority functionality of these nodes serves a security purpose by allowing the use of decryption keys as single-use keys with minimal computation and without sharing unnecessary information with the encryptor user. To accomplish this, the attribute authority generates the single-use part of the decryptor key and sends it to the selected mix node. Consequently, even if the encryptor targets a whole layer of mix nodes with their encryption, only the mix node selected by the $AA + LB$ node can perform the mix operation. Further details on how this mechanism works are available in section 4.2.

Providers

The providers, which are the final layer of the mix network before the users, have similar functions as those defined in Loopix [17], and we have not introduced any new services to these entities.

4.2 ABE requirements

In our protocol, we utilize ABE that supports multiple authorities, constant-size ciphertext, and user revocation.

The use of multiple authorities aims to reduce the trust placed on any single authority while improving the system's overall performance.

The constant-size ciphertext is crucial for ensuring that packets are of the same size at all points in their path and indistinguishable from other packets in the mix network.

The requirement for user revocation is closely linked to the load balancer's role in selecting the mix node for a particular layer. If every mix node in a layer can perform the mix operation, the packet may be vulnerable. However, by incorporating the concept of single-use keys into user revocation, we can target an unknown node.

Encrypt to an unknown target in a group

Figure 6 illustrates how this concept works in our system. Each node has a long-term attribute, $attr_{layer}$, which specifies the layer to which it is assigned upon registration, as well as a single-use attribute, $attr_{single}$, that is active for only one decryption of a specific packet and is revoked instantly.

When a message sender creates a packet, he encrypts it with a policy that requires both attributes. This ensures that even if the packet is leaked, it cannot be decrypted because no one satisfies the $attr_{single}$ part of the policy.

Once the packet is created, it is sent to the $AA + LB$, which selects a mix node and assigns the $attr_{single}$ to it. After the mix node completes its work, the attribute is revoked.

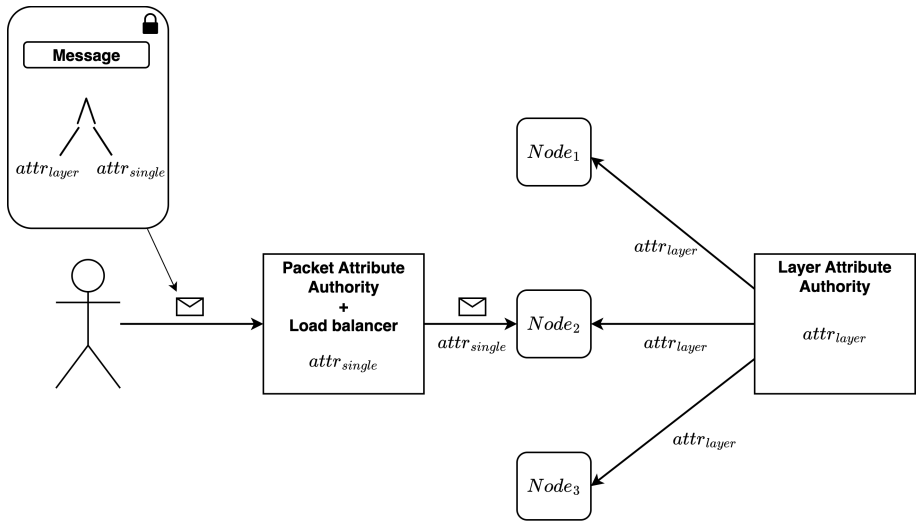

Figure 6: Encrypt to an unknown target in a group

To implement this concept in practice, we use the scheme proposed by Zhang et al. [23], which satisfies the requirements of multiple authorities, constant-size ciphertext, and user revocation.

In the scheme U denotes the set of all attributes. During the setup phase of the scheme, an injective τ encoding is chosen, which sends each of the m attributes $at \in U$ onto an element $\tau(at) = x \in \mathbb{Z}_p^*$. In the proposed use of the scheme, the attribute authorities would generate decryption keys using τ as the attribute's encoding function, since the documents are encrypted using that. Then, when a user gets revoked, they change the encoding function to τ' and create an update key, with which the ciphertexts and the decryption keys of the non revoked users can be updated.

In our case, the default τ is used only for the encryption and for the $attr_{layer}$ decryption key part generation. This way, the encryptors can encrypt targeting a specific layer, but no one can decrypt, since $attr_{single}$ is not generated using τ.

As mentioned earlier, an $AA + LB$ node decides which mix node gets the encrypted packet in our system. Since this node handles the layer's single-use attribute, which should be included in the encryption phase, it can compute to a τ' encoding function the update key, update the packet, and send out the corresponding decryption key part to the picked mix node. After this, the $AA + LB$ node should choose a different encoding function to protect the next packet from the recently selected mix node, operating with single-use keys, which means instant user revocation.

4.3 Packet

Requirements

We have set the following requirements for our packet design to fulfil.

- *Secrecy*: During the mixing process, packets should only reveal the information necessary for each layer, in order to prevent malicious mix nodes from gaining access to extra information. This helps to ensure anonymity and protection against malicious nodes.
- *Indistinguishability*: To prevent adversaries from tracking the route of messages, packets in the network should appear random and maintain a consistent length throughout their journey.
- *Infinite loop resistance*: If the number of hops is not limited, adversaries could flood the network with messages, causing significant delays or even preventing the system from operating. Therefore, limiting the path length is necessary to prevent such attacks.
- *Support for flags*: The packet must be designed to provide information to nodes that can aid their operation or enable additional features, such as drop flags that enable the use of cover traffic.
- *Expandability*: "The packet should also be expandable to accommodate additional information that may be required for future features, without requiring changes to the packet building method.

Notations

Since our method of encrypting to an unknown target is based on ABE, to understand the construction of the packet, first we will introduce the encryption function and its notations from [23].

To encrypt a message M the sender sets an access policy, which in our case as introduced in Section 4.2 is the necessity of the layer attribute and the single-use attribute. The layer attribute should be identical for every node, with distinct value for each layer. On the other hand, the single-use attribute must be a different attribute for each layer. This way one layer's $AA + LB$ node can only generate the decryption key part for the intended layer (for the single-use attribute of that layer).

With encryption, we always target one layer. In our case, two AAs handle the decryption key. Let I_{AA} be the set of these two AAs, and $aid \in I_{AA}$ represent any of the AAs. Additionally, let $\mathbb{G}$ and $\mathbb{G}_T$ denote two multiplicative groups with the same prime order p. The value $v_{aid} \in \mathbb{G}_T$ is introduced in the global public key of the protocol. $A_{aid} \in \mathbb{Z}_p^*$ are chosen randomly by the central authority, and the set of these values forms the global master secret key. AA aid randomly selects $B_{aid}, Y_{aid} \in \mathbb{Z}_p^*$, which serve as the local master secret key for AA aid. (In the referred paper, $A_{aid}, B_{aid}, Y_{aid}$ are denoted as $\alpha_{aid}, \beta_{aid}, \gamma_{aid}$, respectively.

However, we have changed the notations here to avoid any potential confusion or collisions.) n_{aid} denotes the upper bound of the size of allowed decryption policy for AA aid, and in this setup, D_{aid} should consist of only one element from $\mathbb{Z}_p^*$, therefore $D_{aid} = \{d_{aid,1}\}$ and $n_{aid} = 2$ in our case. Similarly, S_{aid} should also be a set with only one attribute, resulting in $s_{aid} = 1$. In addition, the notation $D_{aid,i}$ represents the set $\{d_{aid,1}, d_{aid,2}, ..., d_{aid,i}\}$. Finally, h is a generator element of $\mathbb{G}$, and $u_{aid} \in \mathbb{G}$ represents a public key component of aid.

Important to mention that the encryption supports threshold access structure too, but in our case that property is not needed. Therefore t_{aid} should be the same value as s_{aid} (which is 1 in our case).

The sender picks a random $\kappa \in \mathbb{Z}_p^*$ and can perform the encryption which's result should be the following for $aid \in I_{AA}$:

$$CT = \begin{cases} C = M \prod\limits_{aid \in I_{AA}} v_{aid}^{\kappa} \\ C_{1,aid} = h^{\kappa \cdot \frac{A_{aid}}{B_{aid}} \cdot X} \\ C_{2,aid} = u_{aid}^{-\kappa} \end{cases}$$

where

$X = (Y_{aid} + \tau(at))(Y_{aid} + d_{aid,1})$ and $at \in S_{aid}$.

The notations of the packet are the following.

Let l be the number of mix nodes that a Scalix mix message will traverse before delivered to the receiver.

Let CH_i be the ciphertext parts denoted as $C_{1,aid}$ and $C_{2,aid}$ after attribute-based encryption for the ith layer.

H_i is the mixing header for the ith layer.

Let n be the size of CH_i and m be the size of H_i for $i \in 1, 2, ..., l$.

Let $r = (l-1) \cdot n$ and $s = (l-1) \cdot m$.

Our system also uses two hash functions to produce a packet:

- $h_1 : \mathbb{G}_T \to \{0,1\}^r \{0\}^n$
- $h_2 : \mathbb{G}_T \to \{0,1\}^{l \cdot m}$

We are using identity-based encryption (IBE), multi-authority ciphertext-policy attribute-based encryption (MA-ABE) and Advanced Encryption Standard (AES) as building blocks of our system.

M denotes a plaintext for each occurrence. Let $E_{IB}(ID, M)$ denote the encryption method of the IBE scheme, where ID is a public key. $E_{AB}(ATTR, M)$ is the encryption method of the MA-ABE with constant-size ciphertexts [23], where $ATTR$ is the access policy (The constant-size ciphertext is an important property, so the packets will be indistinguishable by their size). Finally, $E_{AES}(K, M)$ denotes an AES encryption with the key K.

The most common access policy notations are in the form of L_i which is a policy that contains a specific layer and a single-use attribute. R is the receiver's identity and P_R is the receiver's provider's identity.

Building a packet

Our packets are built in an onion structure, where each layer only reveals the information necessary for the node to perform the mix operation.

The necessary data to perform a mix are the amount of delay, the address where to forward the packet and the drop flag. We made it possible with the use of ABE and XOR cipher. The packet is built from three parts, the header, the cipherheader and the body. The header holds all the mix information listed above, the cipherheader has the CH_i part of the ABE cipher and the body is the rest of the ABE cipher.

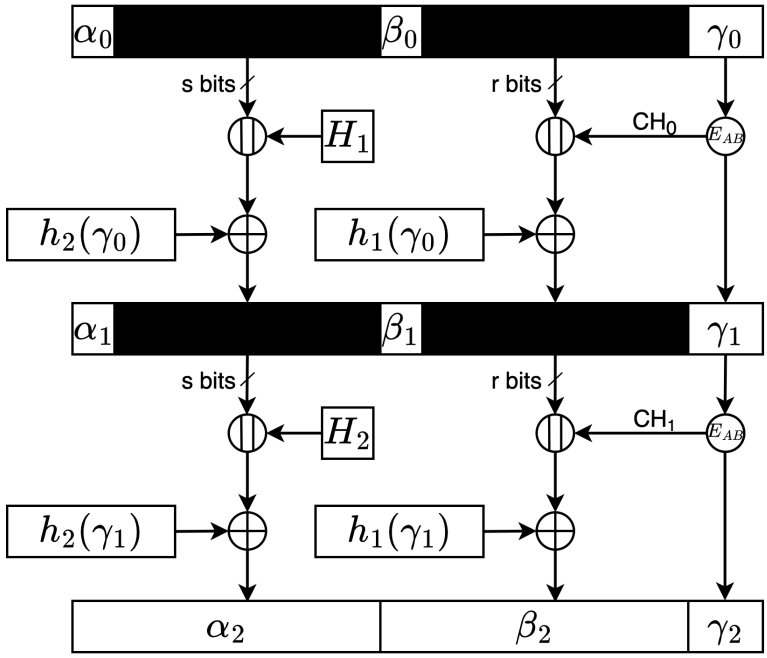

Figure 7: The construction of the Scalix packet for two mix layers

Figure 7 presents how our packet is built for the layers that it will traverse through. The core ($0th$) layer is for the receiver's provider, so it does not hold any mix information.

$\gamma_0 = E_{AES}(K, M)||E_{IB}(R, K)||E_{IB}(P_R, R)$
$\beta_0 = \{0,1\}^{l \cdot n}$ random bits
$\alpha_0 = \{0,1\}^{(l \cdot m)-1}$ random bits, and the last bit is the drop flag.
For each $i \in \{1, 2, ..., l\}$:
$\gamma_i = E_{AB}(L_i, \gamma_{i-1}) \setminus CH_{i-1}$
$\beta_i = \beta_{i-1[n..l \cdot n]}||CH_{i-1} \oplus h_1(\gamma_{i-1})$

$\alpha_i = \alpha_{i-1[m..l \cdot m]} || H_i \oplus h_2(\gamma_{i-1})$

How this construction keeps the information secure and reveals only for the right layer will be discussed more in Section 4.4.

The size of our packet is constant. This is possible by using constant-size ciphertext ABE, which makes the body part (γ_i) the same size every time and also the cipherheader fragments (CH_i). Since the mix information is the same for every layer, the size of it is easy to be standardized and make it constant in the system. In addition, since the number of mixes are set by the system to l, and now we know all the fragments are constant, n and m bits for CH_i and H_i respectively. β_i and α_i can be set to a constant size of $l \cdot n$ and $l \cdot m$ respectively.

The path length cannot be more than l. Since α_i is $l \cdot m$ bits long and every time a new layer is added to the packet, we cut the first m bits of the header and then we concatenate the new part, after l steps we will start to cut useful information, keeping the path lenght l, just making it unable to reach its intended destination.

Also, since the information the header holds is defined by the system and the only requirement is that it be standardized with constant size m, it also gives the option of easy expansion making it flexible for future features if needed.

4.4 Mix operation

During the mixing operation, a mix node can only extract the intended information and no more. As a result, dishonest mix nodes are unable to compromise the anonymity of the system as long as there is at least one honest mix node present.

Our system uses the Poisson mix strategy defined in the paper of Loopix [17]. The layer's extraction operation works as shown on Figure 8.

If a received packet is not of uniform length, it indicates that the previous mix node did not follow the mixing method. In such cases, the packet should be discarded.

For each $i \in \{l, l-1, ..., 1\}$, assuming the node is authorized for decryption (the attributes are correct):

The mix node finds at the end of β_i the CH_{i-1} part of the ABE cipher. Using CH_{i-1} and γ_i it can perform a decryption, receiving γ_{i-1}.

Once it obtained γ_{i-1} it should XOR β_i with $h_1(\gamma_{i-1})$. Then the result will be right shifted with n bits to throw away the used CH_{i-1} and concatenate it to n random bits, to keep it the same size. At the end of this sequence it will receive β_{i-1}.

To acquire H_i, the mix node will XOR α_i with $h_2(\gamma_{i-1})$. The last m bits of the result is H_i. After that it should right shift the result of the XOR with m bits and concatenate it to m random bits. Once its done, the result is α_{i-1}.

At the end it should just wait for the amount of the delay and after that forward the result to $AA + LB_{i-1}$.

In the case when the node is not authorized for decryption, it cannot perform the attribute-based decryption, therefore can't obtain any information. This also protects the inner layers from the current mix node.

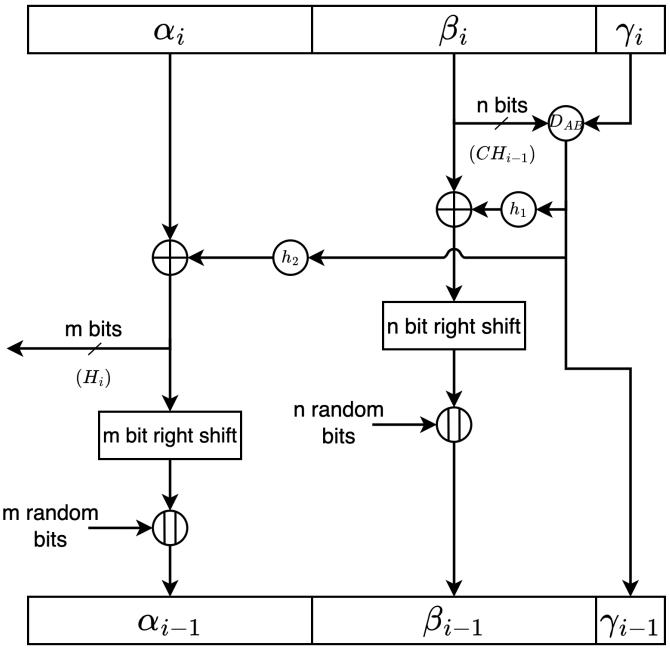

Figure 8: The mix operation of the Scalix packet

4.5 Mixnet as reusable anonymous return channel

We have mentioned that one of our goals is to support the use of our system as a reusable anonymous return channel, which allows the recipients of anonymous messages to send anonymous replies without knowing who they are replying to. This provides complete anonymity to the sender. A mix network solution for this purpose was proposed by Golle and Jakobsson in [11]. In order to offer this service in our system, we need to modify the content of the core of the packet body.

The original was: $\gamma_0 = E_{AES}(K, M)||E_{IB}(R, K)||E_{IB}(P_R, R)$

Let S be the notation of the sender's ID and P_S is the sender's provider's ID.

If the core of the body part is

$\gamma_0 = E_{AES}(K, M)||E_{IB}(R, K)||E_{IB}(P_R, R)||E_{IB}(S, K)||E_{IB}(P_S, S)||E_{AES}(K, P_S)$,

and the rest of the packet building and the mixing operation stays the same, Scalix will function as a reusable anonymous return channel.

If the receiver wants to reply, he will encrypt his message with the received K encryption key and attach $E_{IB}(S, K)||E_{IB}(P_S, S)$ to it which were included in γ_0. This way building the original part of a mix packet. γ_0 also includes the AES encryption of P_S, which has to be included in the packet header to be routed to the correct provider, since a provider gives service to multiple users, this won't break the anonymity of the sender. The rest of the construction is the same as explained

earlier.

Assuming the IBE used is secure, it is not possible to decrypt S without the provider's decryption key. When the receiver of γ_0 obtains P_S, it narrows down the set of possible senders. However, since every provider serves multiple senders and the system is designed such that each user sends messages periodically and receives cover messages, the sender can remain anonymous.

4.6 Proof of security

Given that our architecture is based on Loopix, our aim is to achieve similar security properties against the same threat model. Specifically, our goal is to prevent adversaries from compromising the anonymity of the communicating parties by attempting to link message senders with the correct receivers.

To achieve their objective, adversaries are capable of observing all traffic within the network. Some may also compromise $AA+LB$ nodes or mix nodes or providers, allowing them to take full control. Furthermore, some adversaries are able to participate as a compromised sender or receiver.

Since our mix strategy is the same as that used in Loopix, for security analysis we refer to [17]. However, our topology includes a group of nodes called $AA + LB$, which requires additional security evaluation.

In terms of packet security, we are partially following the principles outlined in Sphinx [7], which include correctness, integrity, and security. We have excluded wrap-resistance, as we believe that it is not a useful property for mix networks, since the adversary is already capable of monitoring the entire network and will therefore not perform a wrapping attack.

$AA + LB$ nodes

If the $AA + LB$ node is compromised, it can be subject to three possible attacks: forwarding messages improperly, ceasing to forward any messages, or attempting to read or modify the packet.

In the first scenario, if the $AA + LB$ node is compromised, it will not function properly as a load balancer and will select certain packets to not be forwarded in a balanced way. However, due to our usage of the Poisson mix strategy, the system can be modeled in its steady states with a Poisson distribution. This allows us to calculate the number of messages that are mixed together at any given time. As a result, mix nodes can establish a range for the number of messages that they expect to receive. If the number of received messages is not within this range, they can report the $AA + LB$ node, and the service provider can take steps to fix the malicious node.

In our current topology, if an $AA+LB$ node stops forwarding messages, it could create a bottleneck in the system since these nodes represent a single point of failure. Consequently, any messages routed through the malfunctioning node would not be delivered. However, we use multi-authority attribute-based encryption, which can

be combined with distributed load balancing techniques. This way the $AA + LB$ would act as a distributed system, eliminating the risk of a single point of failure.

In the third attack, the malicious $AA + LB$ node would act as an honest node while attempting to extract as much information from the packet as possible. However, in the normal case, this node is weaker than a malicious mix node because it cannot decrypt the attribute-based cipher and, therefore, has no means of gathering or altering information. Let's assume that the malicious $AA+LB$ node manages to obtain a decryption key, which allows it to access the information intended for that layer's mix node. With this, the node could potentially modify the delay information or the forward address of the packet. However, changing the forward address would be ineffective since it would be received by a layer that cannot decrypt the attribute-based encryption part and therefore cannot forward the packet. On the other hand, modifying or reading the delay information could allow the adversary to track the packet for one hop. Nonetheless, as proven in [17], a single honest Poisson mix provides a measure of sender-receiver unlinkability.

The $AA + LB$ node is subject to overload attacks as well. Nevertheless, our system does not restrict the $AA + LB$ node to a single server. A potential solution is to distribute the ABE protocol's parameters and keys across multiple servers and utilize a dynamic distributed cooperative load balancing algorithm. This approach can help mitigate the impact of the attack. In a cooperative load balancing algorithm, the servers work together collaboratively to balance the load. They share information and cooperate in the decision-making process to ensure an optimal distribution of workload. This can involve exchanging data about server capacities, current loads, and performance metrics [1].

Correctness

It is evident that our system works correctly if there is no adversary and all the entities are honest. That is, all the $AA+LB$ nodes produces the correct decryption key parts and forwards the packets to the mix nodes, which processes the packet correctly and finally the correct message is sent to the correct receiver.

Integrity

Our system has a predetermined maximum number of layers that a mix packet can travel through during its path, which is set during the setup. As detailed in Section 4.3, if an adversary attempts to create a longer path by adding more layers, information loss will occur in the header and cipherheader parts of the packet. This causes the length to remain at the maximum amount, but results in the message being lost and the final destination being changed with each additional layer added.

Sender-receiver third-party unlinkability, sender and receiver unobservability and mix security

The proofs of sender-receiver third-party unlinkability, the sender and receiver unobservability and mix security are based on the fact that our system is utilizing

Poisson mix and employing the same cover message strategies as described in [17]. As the proofs of these properties are already given in the paper of Loopix, we refer to the description provided there.

Message Indistinguishability

If we assume that the MA-ABE scheme used in our system is secure (meaning that an adversary without the private key cannot perform decryption), any attempt to extract useful information from a packet would be futile. This is because the information is hidden and even the number of hops traveled by the packet is concealed, since the packet size remains consistent and the bits appear random to any inspector. Moreover, since all packets are designed to have the same size, it is impossible to distinguish a real message from a cover message.

5 Conclusion

In this paper, we have introduced a mix network that incorporates practical load balancing and significantly reduces the size of the shared database of node information. We have utilized identity-based and attribute-based encryption as effective tools to achieve these objectives. These encryption methods have been particularly useful in scenarios where we need to encrypt to entities whose identity we know as well as entities whose participation in the communication process is uncertain. Our solution is also unique in this field of cryptography as it provides an identity-based mix network that can be utilized as a reusable anonymous return channel, thus bridging a gap in this cryptographic family.

References

[1] Alakeel, A. A guide to dynamic load balancing in distributed computer systems. *International Journal of Computer Science and Network Security*, 10(6):153–160, 2009. URL: http://paper.ijcsns.org/07_book/201006/20100619.pdf.

[2] Angel, Y., Danezis, G., Diaz, C., Piotrowska, A., and Stainton, D. Katzenpost Mix Network specification, 2019. URL: https://github.com/Katzenpost/docs/blob/master/specs/mixnet.rst.

[3] Bethencourt, J., Sahai, A., and Waters, B. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*. IEEE, 2007. DOI: 10.1109/sp.2007.11.

[4] Boneh, D. and Franklin, M. Identity-based encryption from the weil pairing. In *Advances in Cryptology — CRYPTO 2001*, Lecture Notes in Computer Science, page 213–229. Springer Berlin Heidelberg, 2001. DOI: 10.1007/3-540-44647-8_13.

[5] Chase, M. Multi-authority attribute based encryption. In *Theory of Cryptography*, pages 515–534. Springer Berlin Heidelberg, 2007. DOI: 10.1007/978-3-540-70936-7_28.

[6] Chaum, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981. DOI: 10.1145/358549.358563.

[7] Danezis, G. and Goldberg, I. Sphinx: A compact and provably secure mix format. In *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009. DOI: 10.1109/sp.2009.15.

[8] Diaz, C., Halpin, H., and Kiayias, A. The Nym network — The next generation of privacy infrastructure, 2021. Whitepaper, URL: https://nymtech.net/nym-whitepaper.pdf.

[9] Diaz, C., Murdoch, S. J., and Troncoso, C. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Privacy Enhancing Technologies*, pages 184–201. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-14527-8_11.

[10] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router, 2004. DOI: 10.21236/ada465464.

[11] Golle, P. and Jakobsson, M. Reusable anonymous return channels. In *Proceeding of the ACM Workshop on Privacy in the Electronic Society*. ACM Press, 2003. DOI: 10.1145/1005140.1005155.

[12] Goyal, V., Pandey, O., Sahai, A., and Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and communications Security*, pages 89–98. ACM Press, 2006. DOI: 10.1145/1180405.1180418.

[13] Huszti, A. A homomorphic encryption-based secure electronic voting scheme, 2011. DOI: 10.5486/pmd.2011.5142.

[14] Huszti, A. and Pethő, A. A secure electronic exam system. *Publicationes Mathematicae Debrecen*, 77:299–312, 2010. DOI: 10.5486/pmd.2010.4682.

[15] Kesdogan, D., Egner, J., and Büschkes, R. Stop-and-Go-MIXes providing probabilistic anonymity in an open system. In *Information Hiding*, Lecture Notes in Computer Science, pages 83–98. Springer Berlin Heidelberg, 1998. DOI: 10.1007/3-540-49380-8_7.

[16] Oujani, A. Tools and protocols for anonymity on the internet, 2011. URL: https://www.cse.wustl.edu/~jain/cse571-11/ftp/anonym/.

[17] Piotrowska, A. M., Hayes, J., Elahi, T., Meiser, S., and Danezis, G. The Loopix Anonymity System. In *26th USENIX Security Symposium*, pages 1199–1216, Vancouver, BC, 2017. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska.

[18] Rjaskova, Z. Electronic voting schemes. Master's thesis, Comenius University, Bratislava, 2002.

[19] Sahai, A. and Waters, B. Fuzzy identity-based encryption. In *Advances in Cryptology — EUROCRYPT 2005*, Lecture Notes in Computer Science, pages 457–473. Springer Berlin Heidelberg, 2005. DOI: 10.1007/11426639_27.

[20] Shamir, A. Identity-based cryptosystems and signature schemes. In *Advances in Cryptography (CRYPTO 1984)*, Lecture Notes in Computer Science, page 47–53. Springer Berlin Heidelberg, 1984. DOI: 10.1007/3-540-39568-7_5.

[21] Tuckley, C., Arneson, E., Kirk, A., Crook, S., and Palfrader, P. Mixmaster. URL: http://mixmaster.sourceforge.net/.

[22] Vécsi, A., Bagossy, A., and Pethő, A. Cross-platform identity-based cryptography using WebAssembly. *Infocommunications Journal*, 11(4):31–38, 2019. DOI: 10.36244/icj.2019.4.5.

[23] Zhang, X., Wu, F., Yao, W., Wang, Z., and Wang, W. Multi-authority attribute-based encryption scheme with constant-size ciphertexts and user revocation. *Concurrency and Computation: Practice and Experience*, 31(21), 2018. DOI: 10.1002/cpe.4678.