INVITED PAPER

A Survey on Mix Networks and Their Secure Applications

These networks are designed to use multiple-stage cryptography and permutation to protect communicated information such as medical records, personal opinions and electronic voting results.

By Krishna Sampigethaya, *Student Member IEEE*, and Radha Poovendran, *Senior Member IEEE*

ABSTRACT | Anonymity is a subdiscipline of information hiding, required in a number of applications, such as in electronic voting. For network communications, anonymity can be provided by a mix network (mixnet). A mixnet is a multistage system that uses cryptography and permutations to provide anonymity. The basic idea of a mixnet has evolved into a number of different classes. In addition to presenting the existing mixnet classifications, this paper classifies mixnets based on the verification mechanisms employed for robustness. The construction of mixnets is presented under a common framework to provide insight into both the design and weaknesses of existing solutions. Basic forms of attack on mixnets and the corresponding robustness solutions are reviewed. Comparison with other solutions for anonymity and suggestions for interesting future research in mix networks are also provided.

KEYWORDS | Anonymity; applied cryptography; mixnets; privacy; protocols

I. INTRODUCTION

A. Overview

Advances in communication technology over the years have broken geographical barriers, making communication networks ubiquitous and seamlessly accessible. Widespread use of these networks by the public has led to the development of network-based personalized services involving sensitive and private information. However, data traffic on these networks can be easily tapped, making privacy an increasing concern. Confidentiality of medical records, restricted access to personal opinions on sensitive societal issues, and prevention of online user profiling by market researchers are among the concerns most frequently raised by users deprived of online privacy.

Any communication can be considered private if it involves confidential or personal data. For preserving privacy of communications, it is essential that the data is not accessed by any unauthorized entity. Equally important is that the data is not linkable to its user with the possible exception the authorized receiver of the data. While access to the data can be restricted using cryptographic encryption, to provide unlinkability of the data to the sender requires anonymity for the communication. Anonymity in a communication context, also known as untraceability, prevents tracing back from a receiver to the sender. To give a popular example of the interrelation between anonymity and privacy, consider a social application such as secure electronic voting. Untraceability between the voter and his/her encrypted vote received by the electoral authority provides anonymity for the communication and, hence, preserves the privacy of the voter.

In the year 1981, motivated by the need for anonymity in network communications, an anonymous communication channel, based on cryptography and permutation, called mix network (mixnet), was proposed in [1]. A mixnet is a multistage system that accepts an input batch of quantities and produces an output batch containing the cryptographically transformed, permuted input batch. The change of appearance and the random reordering of the batch by the mixnet prevents trace back from output to input, hence achieving untraceability between the input and output batches. This seminal work spurred significant interest in the development of anonymous channels for communication networks and for network applications requiring privacy. In this paper, we provide a tutorial review of mix networks constructed since the

Manuscript received August 1, 2005; revised August 1, 2006. This work was supported in part by NSF Grant ANI-0093187-002 and ARO Grant W911NF-05-1-0491.

The authors are with the Network Security Lab (NSL), Department of Electrical Engineering, University of Washington, Seattle, WA 98195-2500 USA (e-mail: rkrishna@u.washington.edu; rp3@u.washington.edu).

Digital Object Identifier: 10.1109/JPROC.2006.889687

mixnet in [1]. While the existing body of literature on mixnets is extensive, we restrict our coverage to maintain clarity of exposition and to satisfy space constraints.

B. Applications

One of the most important applications of mixnets is in secure electronic voting. In e-voting protocols, a mixnet anonymously communicates the ballots from the voter to the electoral authority, hence providing ballot secrecy [2]. For error-free elections, problems related to voter fraud and coercion have to be resolved, and ballot/tally verifiability must be provided. The mixnet accommodates for incoercibility to an extent, while enabling verification of voter eligibility, ballot integrity, and tally accuracy [3]. Moreover, using mixnets in remote e-voting, it becomes possible to nullify the requirement for geographical proximity for voting and to avoid the use of problematic provisional and absentee ballots [4]. Hence, mixnets can play a pivotal role in protecting the integrity and enhancing the accessibility of electronic elections.

Other applications such as anonymous e-mail [5]–[8], employ mixnets to satisfy flexible and distributed deployment on public networks. Mixnets have also been used for anonymous telecommunications [9], [10], and anonymous internet communications [11]–[14], involving delay-sensitive two-way interactions between two or more entities over a public network. However, in such anonymous communication applications, misuse of anonymity for illegal purposes is always a concern [7], [15]. In addition to detection of such misuse, it is important for the mixnet to be able to show that it was not an intentional aid to any inappropriate activity. By designing such a mechanism in the mixnet, reliable anonymous communication providers can be encouraged to provide service.

Location privacy in wireless networks is another area benefiting from mixnets [16]–[18]. Mixnet solutions to privacy with radio frequency identication (RFID) tags was recently reported in [19]. As the list of applications continues to grow, the construction of mixnets is being driven into distinct classes. Together, these classes constitute a broad spectrum of working mechanisms that tradeoff properties of the mixnet.

C. Mix Networks

The design of a mixnet is based on providing anonymity for a batch of inputs, by changing their appearance and removing the order of arrival information. As shown in Fig. 1(a), the main component of a mixnet is the stage, also known as the *mix*, that performs mixing on a batch of inputs. Note that the inputs may arrive at the stage at different times, as seen in Fig. 1(b). The mixing operation involves a cryptographic transformation using either decryption or encryption, that changes the appearance of inputs, followed by a permutation on the batch of transformed inputs. The mixed batch is then forwarded in parallel by the stage at time T_{out} to the next destination, as shown in Fig. 1(b). The batching and the permutation together hide the order of arrival information of the inputs. Note that if the batch size is l, by observing the mixed output batch from a stage, one can only guess the correspondence with an input with probability, $1/l$. For example, in Fig. 1 where $l = 5$, the probability of guessing an input will be at best, 1/5. Hence, an increase in the number of inputs to the stage increases the anonymity provided by the stage.

A mix network consists of several interconnected stages depending on the robustness of anonymity required. Each stage performs mixing on its inputs, and the mixed batch is then forwarded to the next stage in the mixnet or directly to their destinations. The interconnection of the stages determines the mixnet topology, and based on the topology of the mixnet, there can be a cascade mixnet or a free-route mixnet, as illustrated in Fig. 2(a) and (b), respectively.

A cascade mixnet consists of stages connected in a fixed, sequential order, as shown in Fig. 2(a). The first stage in the cascade mixnet receives a batch of inputs, performs mixing, and then transfers in parallel the mixed

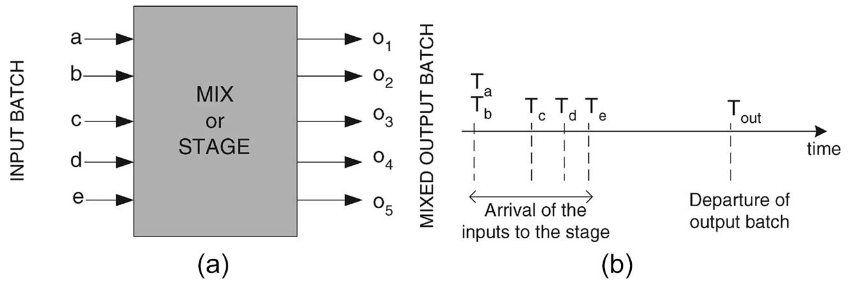


Fig. 1. ***(a) Mixing by stage changes appearance of inputs and also removes order of arrival information. Output batch is a permutation of the transformed input batch. (b) Different times of arrival of inputs at the mix/stage and exit of mixed output batch.***

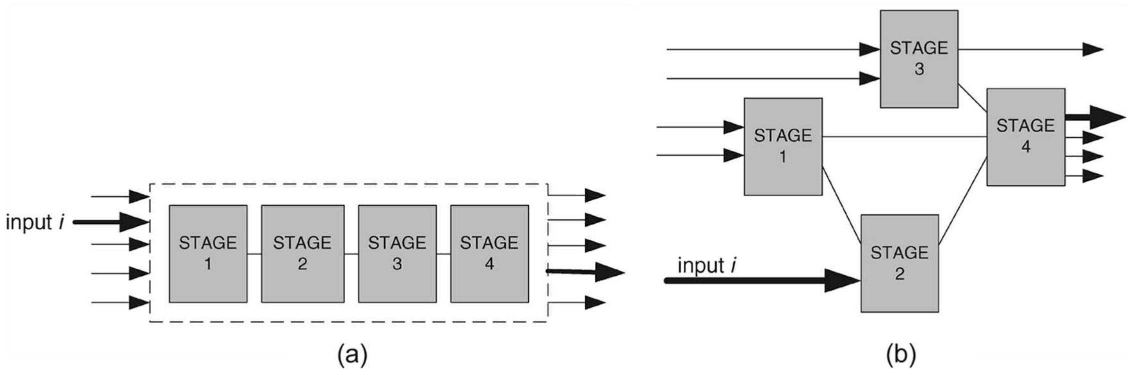


Fig. 2. ***Mixnet Topologies. (a) Cascade topology containing a fixed sequence of four stages. Anonymous communication path of five inputs includes all four stages. (b) Free-routing topology containing four interconnected stages. Each of five inputs can have from one to four stages included in communication path. For instance, an input i has stage 2 and stage 4 only in its anonymous path.***

batch to the connected second stage. The second stage then repeats mixing and forwarding, and the process continues until the final stage outputs the untraceable inputs. Hence, in a cascade mixnet, all the inputs traverse the same path. On the other hand, in a free-route mixnet there can be several anonymous paths available for each of the inputs. As seen in Fig. 2(b), stage 2—stage 4 forms an anonymous path for an input *i*. Stage 2 receives input *i*, waits for an input from stage 1, performs mixing on them, and forwards them in parallel to stage 4. The topologies are covered later in Sections IV and VII.

In the mixnet communication model, multiple senders communicate anonymously with one or more receivers. Each sender may communicate with a separate receiver as in an e-mail application, or multiple senders may communicate with a single receiver as in electronic voting. It is also possible that a sender communicates with multiple receivers, as in multicast applications. A mixnet provides anonymous communication, as long as there are two or more senders using the mixnet, with the batch size being two or more in at least one stage of the mixnet. Further, if the mixnet is used for a two-way communication, then the receiver must be able to reply to the anonymous sender.

D. Basic Attacks and Adversary Model in Mixnets

In order to evaluate the security properties of a mixnet, we have to define an adversary model. The adversary can launch passive or active attacks on the mixnet. A passive attack is often called traffic analysis attack [20], where the adversary observes traffic going into and out of the stages of the mixnet. The objective of such an attack is to correlate inputs to corresponding outputs at the stages, and, hence, breach the anonymity provided by the mixnet. An active attack is called traffic manipulation attack [20], where the objective of the adversary includes corrupting the inputs to the stages, hence attacking the integrity of the mixnet while also enabling tracing of the corrupt inputs. The manipulation can be in form of addition, deletion, modification, or delay of traffic.

The adversary may control one or more compromised stages to actively manipulate the traffic received or passively trace the messages at these stages. A mixnet may also contain faulty stages that may fail to conduct mixing operations, and thus not produce any output batches. Further, in this paper, we assume a powerful adversary as in [1]. A powerful adversary is capable of tapping all the channels of the public communication networks and can passively eavesdrop and trace all the communications to and from the mixnet as well as between the mixnet stages. By making this assumption, we ignore any possible anonymity gained by the inputs before or after traversing the mixnet.

E. Previous Work on Comparison and Classification of Mixnets

Over the last two decades, there has been active research on mixnets and the literature is abundant [21]. In [22] and [86], the different mixnet topologies have been investigated and compared. In [23], the different batching mechanisms and the resulting anonymity and performance tradeoffs have been studied. An entire body of mixnet research literature stems from the secure electronic voting application, where mixnets have to be verifiable and robust against compromised stages launching attacks on anonymity as well as integrity. In [2], an overview of the mixnets proposed for secure e-voting application has been provided.

Recent works such as [23]–[27] have classified mixnets based on cryptographic transformation at the stages, the topology, the batching strategy of the stages, and reliability. Yet, they are not extensive in presenting the mechanisms involved in the various classes of mixnets. One of the main contributions of this paper is that it provides an extensive as well as in-depth coverage of mixnet designs and presents them under a unified framework. The paper also provides a classification of mixnets, based on verifiability mechanisms. Mixnets employing verifiability mechanisms are particularly applicable in secure e-voting protocols.

F. Paper Outline

Section II lists the requirements that mixnets need to meet for secure applications. Section III presents the types of mixnets based on the cryptographic transformations used at the stages. In Section IV, the cascade topology of mixnets is covered in detail. Section V classifies the verifiability mechanisms used in cascade mixnets. Section VI describes a method to decrease the latency incurred in cascade mixnets. Section VII covers free-route mixnets and the related robustness measures employed in them. In Section VIII, the use of reputation in mixnets is presented. In Section IX, a comparison of mixnets is provided along with open problems. Section X presents our conclusions and suggestions for interesting future research in mixnets.

II. MIX NETWORK PROPERTIES

The mixnet properties are determined by the requirements of various applications. These requirements can be categorized into security, performance, and implementation requirements.

A. Security Requirements

Mixnet applications need to provide certain security guarantees. Based on such guarantees and the adversary model described previously, the following security properties can be listed for the mixnet.

1) *Anonymity*: The mixnet is designed to provide untraceability (unlinkability) as the basic property between senders and receivers [101]. No one should be able to trace any input through the stages of the mixnet. The untraceability provided by a mixnet is captured by an anonymity set [81], [85], and the level of anonymity can be quantified by using entropy of the anonymity set probability distribution [102], [103].
2) *Integrity*: The messages from the senders have to be anonymized by the mixnet without any corruption of the data. An integrity check is provided in most mixnets by verification mechanisms and checksum techniques.
3) *Verifiability*: The mixnet must provide some means for verification of the correctness of the messages in the mixed output batch.
4) *Robustness against attacks*: The mixnet must be robust against passive, as well as active, attacks by an adversary. Various mechanisms are employed in a mixnet (both by the communicating sender and receiver, as well as the stages) to ensure robustness, as seen in Sections IV–VII. Note that the verifiability property is related to robustness against some types of active attack, since verifiability ensures honest participation of each stage.
5) *Fault-tolerance*: The mixnet must be able to tolerate a certain number of faulty stages among the participating stages during its operation.

B. Performance Requirements

The mechanisms used to address the security properties in a mixnet determine the performance of the mixnet. The performance metrics used to evaluate mixnets can be listed as follows.

1) *Latency*: The processing at each stage and the verifiability and other robustness mechanisms (see Section VII) in the mixnet take a finite amount of time, hence, adding to delay in the communications. Low latency is crucial for real-time applications requiring anonymity. However, most of the security properties contradict latency, especially under low-traffic conditions.
2) *Throughput*: This is a measure of the number of actual sender messages a mixnet can output per unit of time. It provides an estimate of the overhead due to the mixnet. The overhead can be mainly due to any traffic padding or dummy messages employed for robustness by the mixnet.

C. Implementation Requirements

While providing security and performance guarantees, the mixnet must also be implementable. More specifically the following requirements must be addressed.

1) *Scalability*: In a mixnet, the level of anonymity can be enhanced by increasing the number of participants (increase in the batch size) and/or increasing the number of stages in the anonymous path. However, scalability of the mixnet, with respect to the number of inputs and stages, is mainly limited by the latency requirement.
2) *Efficiency*: The mix network protocol must minimize complex computations and communications. The computational complexity is measured mainly in terms of modular exponentiations or number of cryptographic operations required in the protocol computations. Communication complexity is often measured in terms of the number of interactions needed between entities participating in the protocol.

Before describing mixnets and related mechanisms, we note that the essential cryptographic primitives and protocols used in the design of mixnets are presented in the Appendix. In this paper, we consistently sacrifice mathematical rigor and implementation specifics [28] for clarity of the main concepts. Also note that modular arithmetic is assumed throughout our exposition [28], [29]. Table 1 provides the notation used in this paper.

Table 1 Standard Notation in Our Paper

Notation	Description
i	A sender using the mixnet
j, stage j	A stage of the mixnet
n	Number of stages in the mixnet
t	Threshold number of stages
m	A message
A_e	Address of an entity e. A_s, A_x denote address of a sender, receiver, respectively
l	Number of inputs in a batch, i.e. the batch size
π_j	Random permutation by a stage j
R, r	Random strings
$x\|\|y$	Concatenation of strings x and y
K_e, d_e	Public and private key pair of an entity e
k_e	Symmetric key of an entity e
$E_K(m, r)$	Encryption of a message m with a public key K, using a random string r
$D_K(c)$	Decryption of a ciphertext c using the private key K^{-1}, corresponding to the public key K.
	Note that $D_K(c) = D_K(E_K(m)) = m$
$E_k[m]$	Encryption of a message m with a symmetric key k
$D_k[c]$	Decryption of a ciphertext c with the symmetric key k
p	A large prime
Z_p^*	$\{1, 2, ..., n-1\}$
g	A generator or primitive element of the group Z_p^*, i.e. $g^{p-1} \equiv 1 \ mod \ p$
$\mathcal{H}(x)$	Cryptographic hash of a string x
$\mathcal{H}_k(x)$	Keyed hash of a string x, under the key k

III. MIXING OPERATION IN MIXNETS

In this section, the various types of mixnets based on the mixing operation employed at the mixnet stages are described. Recall that the mixing operation provides anonymity by changing the appearance and removing the order of arrival information of the inputs. The change in appearance of the inputs is due to a cryptographic operation. The batching and the permutation of transformed inputs hide any time of arrival information. The types of cryptographic operations that can be employed at the stages of the mixnet are decryption and encryption. Consequently, there is a decryption mixnet consisting of stages that decrypt and permute batches and a reencryption mixnet where the stages reencrypt and permute batches. We detail these mixnets and their variants as follows.

A. Decryption Mixnet

This category of mixnets was initiated in [1]. In decryption mixnets, the sender is required to encrypt the message with the keys of the stages. Hence, a stage can change the appearance of its inputs by decrypting with its key. This can be accomplished with the use of public key cryptosystems such as RSA [30], seen in Appendix A. Note that the use of symmetric keys [29] is also possible if the sender shares a pairwise key with each stage of the mixnet. A decryption mixnet using only the public keys of the stages is as follows. A sender i encrypts a message m with the public key of each of the n stages in its anonymous communication path as

$$\begin{aligned}
&\text{forwarding onion}_n \\
&\quad = E_{K_x}(m)\|r_n, \\
&\text{forwarding onion}_{n-1} \\
&\quad = E_{K_n}(A_x\|\text{forwarding onion}_n)\|r_{n-1}, \\
&\quad \vdots \\
&\text{forwarding onion}_j \\
&\quad = E_{K_{j+1}}(A_n\|\text{forwarding onion}_{j+1})\|r_j, \\
&\quad \vdots \\
&\text{forwarding onion}_1 \\
&\quad = E_{K_1}(A_2\|\text{forwarding onion}_2)\|r_1, \\
&E_K(m, r) \\
&\quad = A_1\|\text{forwarding onion}_1
\end{aligned} \tag{1}$$

where $K = (K_1, K_2, \ldots, K_n)$ are the public keys of n stages, and $A_1, A_2, \ldots, A_n$ are the addresses of the n stages.

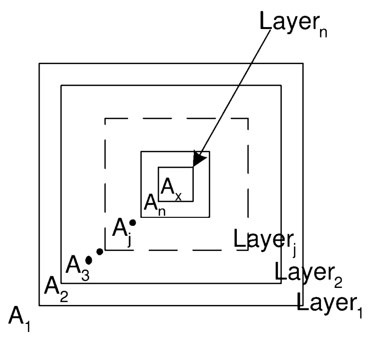


Fig. 3. ***Visualization of typical structure of n-layered onion. Inside each layer we find next stage's address, forwarding onion, and also additional control information.***

$r = (r_1, r_2, \ldots, r_n)$ are random strings that are used to randomize the encryption of each layer (see Appendix A.1). A_x is the address of the receiver. The key K_x is the receiver's public key.

The resulting quantity in (1) can be visualized as an n-layered onion [11], with the forwarding onions nested in each of the n layers, that is broadcasted by the sender. This is illustrated in Fig. 3. The sender onion in (1) is given as

$$E_K(m,r) = A_1 \| E_{K_1}(A_2 \| E_{K_2}(A_3 \| \ldots E_{K_{n-1}}(A_n \| E_{K_n}(A_x \| E_{K_x}(m) \| r_n) \| r_{n-1}) \ldots \| r_2) \| r_1). \quad (2)$$

Fig. 4 gives a graphical illustration of the decryption performed on the sender onions by the mixnet. Each stage j in the path peels off a layer from the onion, i.e., decrypts using its private key K_j^{-1} as

$$D_{K_j}\left(E_{K_j}(A_{j+1} \| \text{forwarding onion}_{j+1})\right). \quad (3)$$

After decrypting more onions received (from other senders and/or stages), the stage j permutes them using a random permutation $\pi_j : l \rightarrow l$, where l is the batch size. This concludes mixing by stage j. The resulting quantities are the forwarding onions which have been reduced in size and constitute the mixed output batch of stage j. These onions are then forwarded simultaneously to their respective next stages. In the case of sender i, the next stage is based on the address A_{j+1}. The mixing is repeated by the remaining stages until, finally, the last stage n outputs the decrypted quantity $E_{K_x}(m)$, which is sent to the receiver at the address A_x.

In the case of two-way anonymous communication where the receiver has to reply to the anonymous sender, the sender must also include return path information (RPI) and a key k_s, along with m. The RPI looks similar to the sender onion in (2) except that only the address of the sender A_s will be encrypted in the onion as follows:

$$\text{RPI} = A_n \| E_{K_n}(k_n \| A_{n-1} \| E_{K_{n-1}}(k_{n-1} \| A_{n-2} \| \ldots E_{K_2}\left(k_2 \| A_1 \| E_{K_1}\left(k_1 \| A_s \| r'_1\right)\right) \| r'_2\right) \ldots \| r'_{n-1}\right) \| r'_n\right) \quad (4)$$

where $r'_1, \ldots, r'_n$ are random strings and k_j, $j = 1, 2, \ldots, n$ are symmetric keys shared between sender and stage j. For the sender to remain anonymous to the receiver, it is necessary that k_s must not identify the sender. Assuming k_s is a symmetric key, the receiver at A_x obtains the following from the mixnet:

$$E_{K_x}(m \| \text{RPI} \| k_s). \quad (5)$$

The receiver decrypts the message and may send a reply m' as

$$E_{k_s}[m'] \| \text{RPI}. \quad (6)$$

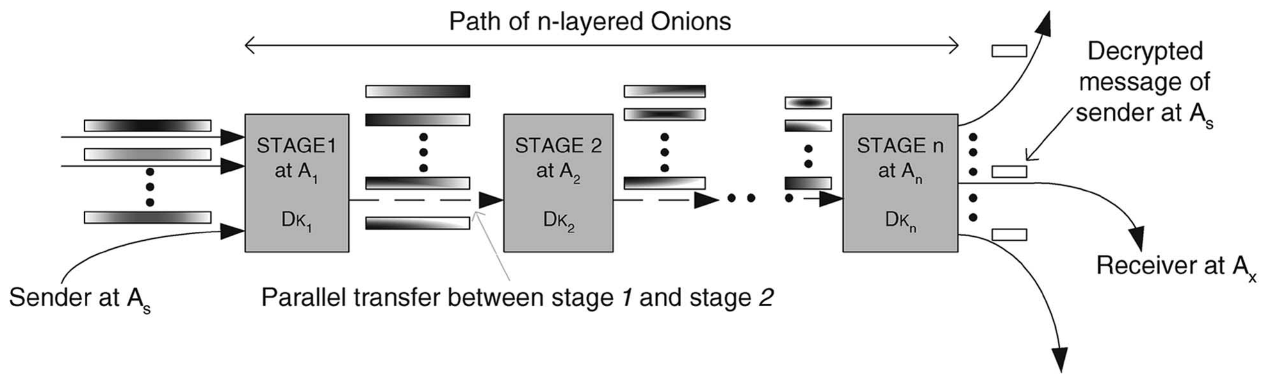


Fig. 4. ***Illustration of onion decryption, in decryption and hybrid mixnets. Length of onions indicates their size, and we can notice the decrease in size of onions as stages are traversed. Decrypted message received by receiver at A_x cannot be linked back to sender at A_s.***

Stage n receives the encrypted reply, peels a layer off the attached RPI onion to obtain k_n, and reencrypts $E_{k_s}[m']$ with k_n to change its appearance. The remainder of the mixing process occurs as before in the forward path (from sender to the receiver). Finally, stage 1 transmits the reencrypted reply to the sender at A_s. Hence, a two-way anonymous communication can be achieved using the decryption mixnet. Note that it is not necessary for the return path to include the same stages as the forward path.

An alternative approach to two-way communication in decryption mixnets has been proposed in [9] and [10]. Here, the sender first sets up a path from stage 1 to stage n and establishes a common parameter with the receiver and the last stage n. The receiver also sets up a path through the mixnet, from stage 1 to stage n. During this setup, the common parameter is obtained by the last stage n and is used to link the sender established path with the receiver established path. In this approach, uninterrupted two-way anonymous communication can occur once both paths are established [9], [10].

B. Hybrid Mixnet

In the previous decryption mixnet, expensive public key operations by the sender are performed on the increasing size of the onions. A more efficient variant of the decryption mixnet is the hybrid mixnet [1], [11], [24], [31], which uses public key operations as well as symmetric key operations and achieves efficiency. In a hybrid mixnet, the forwarding onion obtained by a stage j from stage $j-1$ is

$$\text{forwarding onion}_j = E_{K_j}(k_j)\|E_{k_j}[A_{j+1}\|\text{forwarding onion}_{j+1}]. \tag{7}$$

As in the decryption mixnet, stage j peels a layer by decrypting with private key K_j^{-1}. But here, on decryption, the stage j obtains a symmetric key k_j that it uses to decrypt the forwarding onion to stage $j+1$. Hence, the complete sender i onion in the hybrid mixnet is given as

$$\begin{aligned} E_K(m,r,k) = A_1\|E_{K_1}(k_1)\|\\ E_{k_1}[A_2\|E_{K_2}(k_2)\|\\ E_{k_2}[A_3\|E_{K_3}(k_3)\|\ldots\|\\ E_{k_{n-1}}[A_n\|E_{K_n}(A_x\|E_{K_x}(m)\|r')]\ldots]] \end{aligned} \tag{8}$$

where $K=(K_1,K_2,\ldots,K_n)$ are the public keys of the stages, $k=(k_1,k_2,\ldots,k_{n-1})$ are the symmetric keys chosen by sender, and r is a random string. Note that the hybrid mixnet is efficient since the expensive public key encryption is only required for the symmetric keys which are nonincreasing in size. The relatively efficient symmetric key operations are used for the increasing-in-size onion.

The hybrid mixnet in [11] encrypts a key material for each stage, instead of the symmetric key. The hash of the key material is used as the symmetric key. On the other hand, the hybrid mixnet in [24] encrypts a public key K for each stage, in place of the symmetric key k. The public key enables use of zero-knowledge (ZK) proof [32] (described in Appendix E) for robustness as seen later in Section V-B2. Variants of the decryption and the hybrid mixnet can include additional control information that can be used by the stages. The control information can be a set of keys used for future communications [11] or for enabling faulty stages to be skipped [85] and cryptographic checksums [6], [8]. These aspects will be covered later in Sections IV and VII.

C. Reencryption Mixnet

The decryption and hybrid mixnets in the previous two sections that are based on public key cryptosystems, such as the RSA [30], have the following weaknesses.

1) The size of the onions decreases as the stages are traversed.
2) The sender is able to trace its onion by appearance as it traverses the stages.
3) The sender is required to encrypt for each stage.
4) The decryption is performed in a predetermined (by sender onion) sequence of stages.

Note that the second weakness can be an advantage under certain circumstances, since the sender can verify that the stages of the mixnet are mixing its input [1]. However, the adversary can still misuse the advantage to trace inputs through the mixnet as described later in Section IV-C2. All the weaknesses listed are addressed in a more efficient decryption mixnet proposed in [33], using the ElGamal public key cryptosystem [34] (described in Appendix A.2). Here, a sender i only needs to perform a single encryption for all the n stages as

$$E_K(m,r) = (g^r\|(A_x\|m)K^r) \tag{9}$$

where g is the generator [34], r is a random string, and K is the public key of the mixnet, computed from the public keys of the stages as

$$K = \prod_{j=1}^{n} K_j = \prod_{j=1}^{n} g^{d_j} = g^{\sum_{j=1}^{n} d_j} \tag{10}$$

where $K_j = g^{d_j}$ and d_j, are the public and the private key, respectively, of stage j. In this type of decryption mixnet, there is no necessity for any predetermined sequence of stages for decryption. Any stage j can randomly decrypt

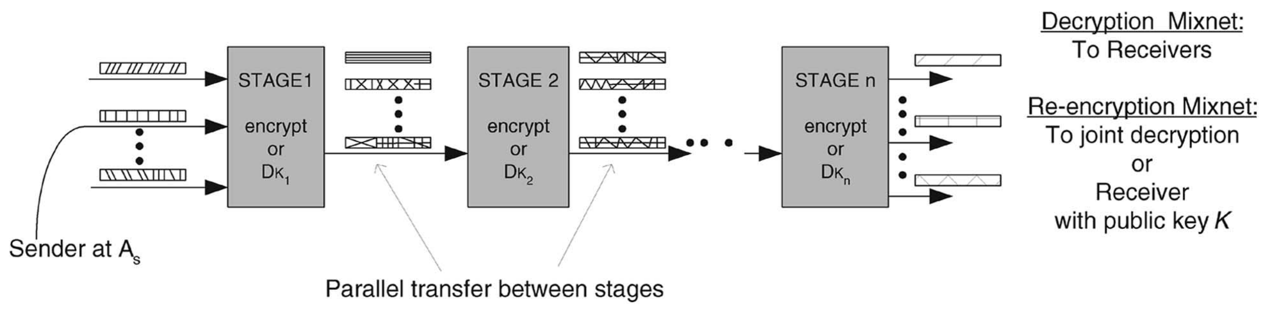


Fig. 5. ***Illustration of ElGamal public key cryptosystem-based mixnet. If it is a decryption mixnet, then each stage decrypts with its private key K_j^{-1} and random strings. Mixnet output batch will be sent to the corresponding destinations. But in a reencryption mixnet, stages reencrypt inputs with random strings. Mixnet output batch will be either decrypted jointly by the group of stages or will be sent to receiver with public key K.***

the sender i input, using its private key d_j and random string r_j as

$$\begin{aligned} &D_{K_j}(E_K(m,r)) \\ &= g^r g^{r_j} \| (A_x \| m)(K^r)(g^r)^{-d_j} \left(\prod_{a=1,a\neq j}^{n} g^{d_a} \right)^{r_j} \\ &= g^{r+r_j} \| (A_x \| m) \left(g^{\sum_{a=1,a\neq j}^{n} d_a r} g^{d_j r} \right) \left(g^{-d_j r} \right) \left(\prod_{a=1,a\neq j}^{n} g^{d_a r_j} \right) \\ &= g^{r+r_j} \| (A_x \| m) \prod_{a=1,a\neq j}^{n} g^{d_a (r+r_j)}. \end{aligned} \tag{11}$$

Note that in the first step of (11), the stage j uses the first component of its input in (9) to obtain $(g^r)^{-d_j}$ and uses the product of public keys of the stages that are yet to decrypt to obtain $(\prod_{a=1,a\neq j}^{n} g^{d_a})^{r_j}$. After decryption of more inputs to form a batch, the stage j broadcasts the mixed batch to the remaining $n-1$ stages. The process repeats, with another stage performing the decryption, until finally all n stages have decrypted using their private keys to obtain $g^{r+\sum_{j=1}^{n} r_j} \| (A_x \| m)$, as one of the outputs of the last stage. Note that the size of the decrypted inputs remains the same at all the stages and that the sender will not be able to recognize its input once the first stage performs random decryption. Fig. 5 illustrates this type of mixnet.

However, the ElGamal-based decryption mixnet in [33] has only limited use because of its inherent weaknesses.

1) *Unlike* other decryption/hybrid mixnets, the next stage addresses or any other control information *cannot* be included with the message in ElGamal-based decryption mixnet, and it does not have a hybrid variant.
2) On the other hand, *like* other decryption mixnets, an ElGamal-based decryption mixnet is still dependent on each stage j to decrypt the input batch. Such a mixnet design cannot tolerate failures in terms of any faulty stages and requires the use of additional mechanisms for robustness as described later in Section IV-B.

The second weakness is avoided in the reencryption mixnet design [33], also based on the ElGamal cryptosystem. The main idea is to utilize the reencryption property of ElGamal encryption (described in Appendix A.3). The senders encrypt their messages with the public key of the mixnet, as in (9), and to change the appearance of the inputs a stage simply reencrypts them with random strings. As before, no predetermined sequence of stages is required for the reencryption mixnet. The following cryptographic operation can be performed first, by any stage j, on the sender i input in (9)

$$\begin{aligned} E_K(m, r+r_j) &= (g^r g^{r_j} \| (A_x \| m) K^r K^{r_j}) \\ &= (g^{r+r_j} \| (A_x \| m) K^{r+r_j}) \end{aligned} \tag{12}$$

where r_j is a random string used by stage j for reencryption of sender i input. After a batch of inputs are reencrypted and permuted, stage j broadcasts the mixed batch to the remaining stages, for further mixing. The mixing terminates at a stage n. The reencryption mixnet is also illustrated in Fig. 5, where after mixing the group of n stages may perform a decryption phase to jointly decrypt each quantity of the mixed output batch as

$$\begin{aligned} D_K\left(E_K\left(m, r+\sum_{j=1}^{n} r_j \right) \right) &= \frac{(A_x \| m) K^{r+\sum_{j=1}^{n} r_j}}{\left(g^{r+\sum_{j=1}^{n} r_j} \right)^d} \\ &= \frac{(A_x \| m) K^{r+\sum_{j=1}^{n} r_j}}{K^{r+\sum_{j=1}^{n} r_j}} \\ &= A_x \| m \end{aligned} \tag{13}$$

where $K = g^d$ is the public key, and d is the private key that can be shared by the n stages using a (t, n) threshold scheme [35] (described in Appendix C). Note that as in the ElGamal-based decryption mixnet, in the reencryption mixnet the sender is *not* able to trace its message as it traverses the stages, since it does not have the knowledge of the random strings used by the stages to reencrypt. Also, the size of the inputs remains the same. However, in the reencryption mixnet, not all stages are needed to reencrypt, though they may still need to participate in the decryption phase to contribute their shares of the decryption key d. Yet, this slight improvement in the independence of the stages makes the reencryption mixnet design better, compared to its decryption mixnet counterpart in [33].

Note that in the reencryption mixnet design, the stages do not have to share the private key if there is only *one receiver* communicating with the senders. The public/private keys can then be those of the receiver, with the public key K known to the stages. The mixnet output is broadcast to the receiver which decrypts with private key d. This type of design is used in applications such as e-voting, where a single authority receives the mixnet output.

Two-way anonymous communication in a reencryption mixnet is possible, if the sender includes the necessary information. As proposed in [36], the sender i broadcasts three ElGamal encryptions to the mixnet as

$$E_K(m, r) \| E_K(A_s \| K_s, r') \| E_K(A_x \| K_x, r'') \tag{14}$$

where K is the public key of the mixnet, A_s and A_x are addresses of the sender and receiver, respectively, and K_s, K_x are the ElGamal public keys of the sender and receiver, respectively. r, r', r'' are random strings. Any stage j first reencrypts the three components using a random string r_j. Hence, the mixnet output batch will contain three reencrypted components in each input as

$$E_K\left(m, r + \sum_{j=1}^{n} r_j\right) \Big\| E_K\left(A_s \| K_s, r' + \sum_{j=1}^{n} r_j\right) \Big\| \\ E_K\left(A_x \| K_x, r'' + \sum_{j=1}^{n} r_j\right). \tag{15}$$

The first component contains the message, and the second component is essentially the RPI. The third component containing the address A_x and the public key K_x of the receiver is used to ensure privacy of the message m as follows. After mixing, the stages of the mixnet jointly decrypt the third reencrypted component in (15) to obtain the address A_x and public key K_x of the receiver. The stages then perform switching encryption [37] on the first component in (15) as

$$E_{K_x}\left(m, r + \sum_{j=1}^{n} r_j\right). \tag{16}$$

The switching encryption is a technique used to blindly decrypt m, i.e., without revealing it, while simultaneously encrypting m under the public key K_x of the receiver. The receiver at A_x obtains the following from the mixnet:

$$E_{K_x}\left(m, r + \sum_{j=1}^{n} r_j\right) \Big\| E_K\left(A_s \| K_s, r' + \sum_{j=1}^{n} r_j\right). \tag{17}$$

The receiver can then reply using the sender's RPI as

$$E_K(m', R) \| E_K(A_x \| K_x, R') \| E_K\left(A_s \| K_s, r' + \sum_{j=1}^{n} r_j\right) \tag{18}$$

where m' is the reply message and R, R' are random strings chosen by receiver. The protocol then proceeds as in the forward path.

Unfortunately, neither of the previous reencryption mixnets is capable of handling multiple receivers without requiring the stages to share the private key and perform the decryption phase after mixing to obtain the address of the receivers. Such a dependency is avoided in an improved variant, called universal reencryption mixnet, proposed in [19]. The universal reencryption mixnet accommodates multiple receivers, without requiring a separate decryption phase after mixing, as described in the following.

D. Universal Reencryption Mixnet

In this mixnet the sender i broadcasts two ElGamal encryptions, one containing the message, and the other containing the public key of the receiver used to encrypt the message, as follows:

$$E_K(m, r) \| E_K(1, r') = (g^r \| mK^r) \| \left(g^{r'} \| K^{r'}\right). \tag{19}$$

Next, a stage j performs the following cryptographic operation:

$$\begin{aligned} &\left(g^r g^{r' r_j} \| mK^r K^{r' r_j}\right) \| \left(g^{r' r'_j} \| K^{r' r'_j}\right) \\ &\quad = \left(g^{r + r' r_j} \| mK^{r + r' r_j}\right) \| \left(g^{r' r'_j} \| K^{r' r'_j}\right) \\ &\quad = E_K(m, r + r' r_j) \| E_K\left(1, r' r'_j\right) \end{aligned} \tag{20}$$

where r_j, r'_j are random strings chosen by stage j for the sender i. The remaining $n-1$ stages repeat the reencryption operation with different random strings. Note that since the sender includes an encrypted form of the public key K used for message encryption, the stages can perform reencryption without knowing K. Only the message containing the portion of the mixnet output batch is broadcast by the mixnet as

$$E_K(m, R_c) = (g^{R_c} \| m K^{R_c}) \tag{21}$$

where R_c is a combination of the random strings used by the stages to reencrypt sender i input. Hence, the mixnet allows senders to communicate with multiple receivers with different public keys and with no decryption phase required after mixing. However, the receiver has to perform an exhaustive search on every mixnet output batch received for possible messages encrypted under its public key K. This is a weakness of this design, in addition to the inability to provide two-way anonymous communications.

Table 2 provides a summary of the pros and cons related to mixnets based on the cryptographic transformations. A significant drawback in the reencryption mixnet and its variants is that the stage addresses cannot be included in the encryption. This limits its application in free-routing topologies, where the mixnet inputs must include address of stages to define the anonymous path. Nevertheless, the cascade topology can be employed to effectively utilize the advantages of the reencryption type mixnet. The next section presents the various details related to mixnets with cascade topology.

IV. CASCADE TOPOLOGY FOR ANONYMITY

It was noted earlier in Fig. 2(a) that a mixnet with cascade topology (cascade mixnet) consists of a fixed sequence of stages, leading to a single anonymous path for all senders communicating with receivers. Only the *first* stage of the cascade mixnet initiates mixing on *all* the sender input batches. Since the path is already established from the input to the output of the mixnet, there is *no requirement for explicit addresses for the stages* to be included in the sender input. However, the stages in the cascade mixnet are interdependent, since a single stage in the mixnet can compromise or fail its operation.

Table 2 Types of Mixnets Based on Cryptographic Transformations

Mixnet Type	Advantages	Limitations
RSA-based Decryption Mixnet [1]	Can include intermediate stage addresses	Needs multiple encryptions by sender Exponentiations needed on increasing size input Requires fixed sequence of stages for mixing Inputs can be traced by appearance and/or size Requires all stages in the mixnet to perform mixing Cannot efficiently accommodate large size messages
RSA-based Hybrid Mixnet [1], [11]	Can include intermediate stage addresses Exponentiations needed only on fixed size input Can efficiently accommodate large size messages	Needs multiple encryptions by sender Requires fixed sequence of stages for mixing Inputs can be traced by appearance and/or size Requires all stages in the mixnet to perform mixing
ElGamal-based Decryption Mixnet [33]	Needs a single encryption by sender Inputs are not traceable by appearance and/or size No fixed sequence of stages required for mixing	Cannot include addresses of intermediate stages Cannot efficiently accommodate large size messages Requires all stages in the mixnet to perform mixing
ElGamal-based Re-encryption Mixnet [33]	Needs a single encryption by sender Inputs are not traceable by appearance and/or size No fixed sequence of stages required for mixing All stages in the mixnet need not perform mixing	Cannot include addresses of intermediate stages Cannot efficiently accommodate large size messages May need stages to jointly decrypt
ElGamal-based Universal Re-encryption Mixnet [19]	Needs two or more encryptions by sender Inputs are not traceable by appearance and/or size No fixed sequence of stages required for mixing All stages in the mixnet need not perform mixing No joint decryption by stages is needed	Cannot include addresses of intermediate stages Requires receivers to search mixnet output batches for inputs encrypted with their public key No two-way communication Cannot efficiently accommodate large size messages

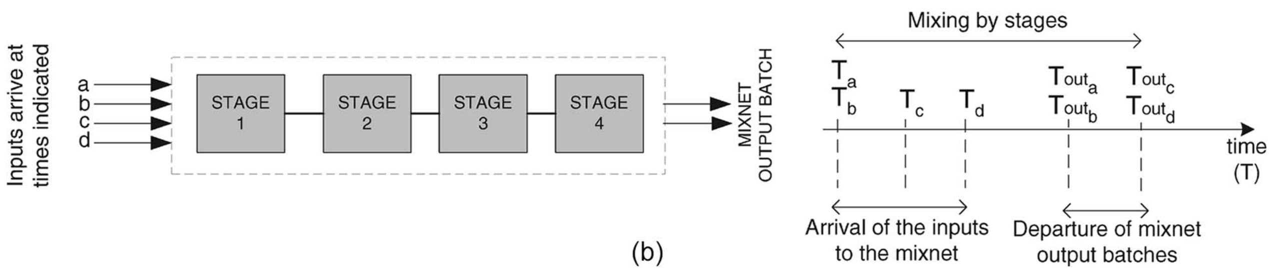


Fig. 6. *Synchronous batching in cascade mixnet with batch size 2.*

A. Synchronous Batching in Cascade Mixnets

An important feature of a cascade mixnet is its batching process. The input batches exit the mixnet in the same temporal sequence in which they are mixed. This is due to the fact that all the batches have to traverse the same path through the mixnet. Hence, the first batch created during a batch period is followed by the second batch, and so on. This is referred to as synchronous batching [38] and is illustrated in Fig. 6, where the inputs arrive at different times but exit in the temporal sequence in batches of two, as indicated.

A typical cascade mixnet protocol can be described as follows. Assume there are l senders and n stages. Each sender i broadcasts an input $I_i^{(0)}$ to the mixnet. After l inputs are received, the batch is formed, and the following is executed by the mixnet.

Basic Cascade Mixnet Protocol

```
Input. $I_i^{(0)}$; $i = 1, \ldots, l$.
For $j = 1, \ldots, n$.
    For $i = 1, \ldots, l$.
        Step 1) Perform cryptographic operation,
        $I_i^{(j)} = f_j(I_i^{(j-1)})$.
    Step 2) Permute all the quantities obtained in Step 1
    using random permutation $\pi_j$.
Output. $O_i$, $i = 1, 2, \ldots, l$, the batch of mixed/anonymized
sender inputs.
```

Note that $f_j(.)$ can be a decryption or a reencryption operation at stage j, as described in the previous section. The previous protocol is the basic cascade mixnet protocol, providing anonymity and integrity when there is *no* active attack. In the case of a reencryption cascade mixnet, an additional subprotocol may be required for the decryption phase, before broadcasting to the receivers. Also, to address the presence of attacks or failures, the protocol will need additional subprotocols, including shared key generation and distribution [35], simulation of faulty stages [39], authentication [1], as well as generation of zero-knowledge proofs [40], as will be seen.

B. Cascade Mixnet Failure and Robustness Measures

Before considering robustness of the cascade mixnet against attacks, we will consider robustness against a faulty stage. A faulty stage in a mixnet will simply fail to perform mixing of its input batch. As seen in Section II, this will affect the fault-tolerance property of the mixnet. Since there is only one path in the cascade mixnet, a faulty stage can cause nonavailability of service. This is particularly true in the case of a decryption cascade mixnet, since the failure of one stage to decrypt its input batch means that the mixnet cannot produce any output. As we have seen earlier, however, in the case of a reencryption mixnet, this is true only if the stages are required to decrypt the final mixed batch. Otherwise, only one of the n stages needs perform reencryption, for the cascade mixnet to successfully produce a mixed batch at the output.

In order to provide fault tolerance in a decryption/hybrid cascade mixnet, the private key of each stage may be shared by the group of n stages, using a (t, n) threshold secret-sharing scheme [35]. In the case of a reencryption cascade mixnet, only the private key corresponding to the public key of the mixnet may be shared by the n stages. Nevertheless, prior to the execution of the cascade mixnet protocol, the stages must participate in a protocol to share the keys. The sharing of keys enables the simulation of the faulty stage by the remaining stages, as will be seen in Section V-B. We note that by using a (t, n) threshold scheme, the mixnet protocol is robust for up to $t - 1$ compromised stages and $n - t$ faulty stages, which represents a tradeoff between security properties.

From a security viewpoint, it has been argued that in applications such as electronic voting, an observable operation failure (such as faulty stage) is better than a failure that may pass undetected. For example, a faulty stage in a cascade mixnet may halt its operation but can be detected and replaced to continue its normal application. However, if the anonymity or the integrity of the cascade mixnet is breached without detection, then it may result in graver consequences. Hence, there exists a vast amount of literature dedicated to finding weaknesses that may exist in a mixnet design and solutions for such weaknesses.

Different types of active attacks on a decryption cascade mixnet were initially reported in [41], and attacks on a reencryption cascade mixnet were reported in [42]. Traffic analysis attacks on anonymity of mixnets have been addressed in detail in [20] and [43]. A general review of the main weaknesses employed in the various attacks, and the resulting mechanisms that are designed to ensure robustness in cascade mixnets, are presented next.

C. Attacks on Anonymity in the Cascade Topology

In cascade mixnet design, anonymity is dependent on the batch containing l inputs and the subsequent mixing by the n stages. For a passive attack, the adversary has to analyze a batch in order to trace any input in the batch, while for an active attack the adversary must control the inputs in the batch.

1) Passive Attacks: The adversary performs traffic analysis in order to conjecture the correspondence between the l inputs and the l outputs in a batch. By observing the inputs to the cascade mixnet, the adversary may get time and appearance information. But after mixing is initiated in the first stage, for tracing an input to the corresponding output, the adversary is forced to choose from the l outputs in the batch. By increasing the batch size l, robustness against this attack can be achieved.

2) Active Attacks: In an active attack, an adversary may control some compromised stages and use them to trace an input passing through. In order to break the anonymity of the cascade mixnet, an adversary must control at least a threshold number of t stages. Hence, by increasing the threshold t (and the number of stages, n) in a cascade mixnet, we can improve robustness against attacks on anonymity.

However, the adversary may try to trace a target input from a target sender by manipulating the mixnet input traffic, such as by including some known $l-1$ inputs in the mixnet input batch. Alternatively, the adversary can include one or more copies or replays of the target input (or the RPI of the target sender in the return path). This attack is not trivial in the case of ElGamal-based cascade mixnets, since identifying any input is not possible between the stages due to the random reencryption of each input at a stage. However, the adversary can still possibly break the anonymity of the unknown message in the target input from the decrypted output batch. In the case of a decryption/hybrid cascade mixnet based on RSA cryptosystem, the inputs are recognizable between intermediate stages; hence, the adversary can trace known inputs, as well as the target input traversing the cascade mixnet. Robustness against these active attacks at the input of the cascade mixnet can be partially achieved by employing authentication of the sender [1] and using zero-knowledge (ZK) proofs [44] as explained next.

D. Authentication of Senders

By authenticating the senders, robustness against manipulation of the mixnet input batch is achieved. The adversary must now control different compromised senders to successfully launch any attack. Authentication can be integrated into the cascade mixnet protocol by making the first stage authenticate the senders along with receipts [1]. The first stage decrypts a layer from each sender onion, signs the forwarding onion, and then sends it as the receipt to the corresponding sender. On obtaining receipt, the sender can confirm that its input has not been corrupted *en route* to the mixnet. The receipt can also be used as a proof in verification, as will be seen in Section V-A.

A more robust approach has been used in [33], where sender authenticated sections on a public bulletin board (described in Appendix B) are employed instead of a trusted first stage in the mixnet. A cascade mixnet using the bulletin board and sender authenticated sections is illustrated in Fig. 7. The senders as well as the stages have authenticated append-only access to designated sections on the board. The inputs are broadcast by the senders to their designated sections on the board. No receipt is needed since the sender can confirm the input posted on the bulletin board. When there is a sufficient number of postings on the board to form a batch, the mixnet performs mixing of the encrypted messages. Each stage obtains its input batch from the board and posts its mixed output batch in a designated section of the board. It must be noted that if an application requires the sender to be anonymous even to the mixnet, then authentication of senders can be performed using identification by pseudonyms [1], [45].

E. Proof of Knowledge of the Message

The use of a publicly accessible bulletin board, however, enables an adversary to more easily replay an input that it wants to trace from the board [42]. This can be addressed by forcing the mixnet to ignore all copies of an input on the board. Yet, the adversary may suitably modify the input so that it does not appear to be a direct copy [41], [42], by using properties of the public key cryptosystem (described in Appendix A.3). The adversary can then replay the duplicate input in the same batch as the original input (or the subsequent batches). Such an input will not be detected by the mixnet, and will be mixed and decrypted as normal. By observing the output batch (or correlating two output batches) for a repeated message m, the adversary may possibly breach the anonymity of the sender of m. This attack is called a duplication attack [42].

The duplication attack can be addressed effectively in ElGamal-based cascade mixnets, by requiring the sender to provide a ZK proof [44] (described in Appendix E). The sender needs to prove that it knows the random secret exponent r used in the ElGamal encryption of the

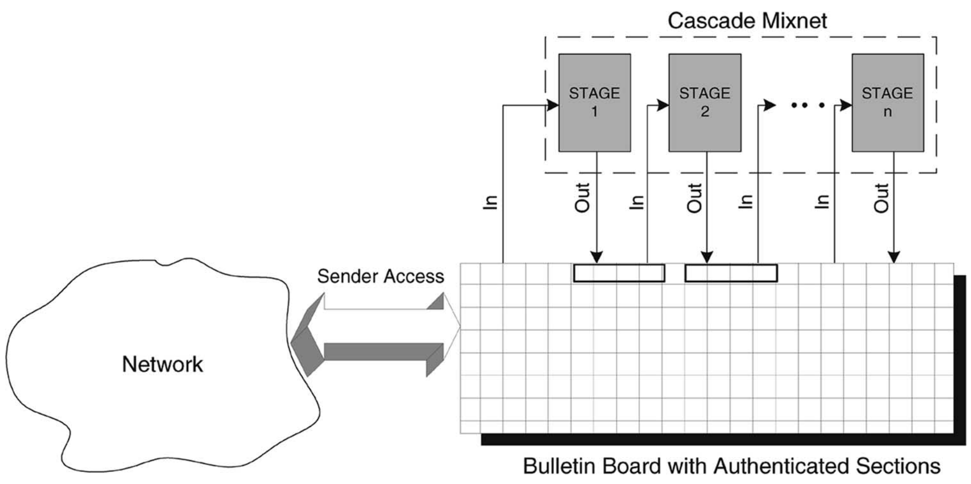


Fig. 7. ***Robust cascade mixnet using public bulletin board and authentication.***

message *m*. However, in the case of RSA-based decryption/hybrid cascade mixnets, a ZK proof may not be feasible [24]; hence, defending against a duplication attack can be a problem. As suggested in [41], a bulletin board must be avoided in an RSA-based decryption/hybrid cascade mixnet, and the senders must time synchronously transmit to the mixnet. This ensures to an extent that the adversary cannot duplicate an input during transmission and send it in the same mixnet input batch. For robustness against replay in subsequent input batches, public key rotation [41] at the stages is required, as seen later in Section VII-G2.

F. Attacks on Integrity in Cascade Topology

The previous solutions based on authentication and sender proofs provide robustness against active attacks at the input of the cascade mixnet. However, as illustrated in Fig. 8, active attacks by compromised stages in the cascade mixnet can still be performed as follows.

1) A compromised stage may simply forward one or more inputs without any transformation.
2) A compromised stage may duplicate (replay) an input and replace another input with this duplicate to trace it at the mixnet output [42].
3) A compromised stage may corrupt a target input, and based on the corruption, the target input can possibly be identified and traced at a subsequent stage or at the mixnet output (tagging attack) [8].
4) While the previous attacks enable tracing of inputs in the mixnet, a worse scenario is when an attack is used to modify the mixnet output. A compromised stage may *delete* one or more inputs

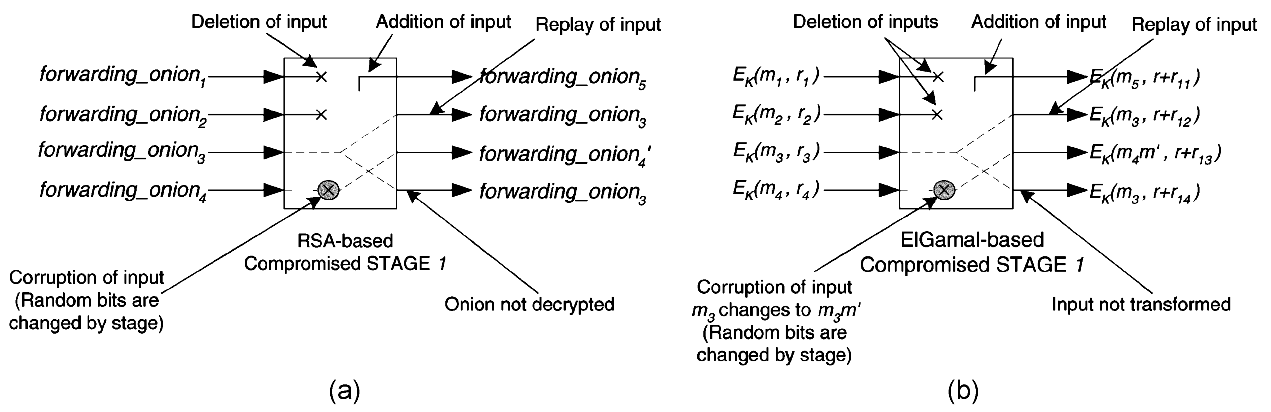


Fig. 8. ***Attacks on integrity of a batch of four sender inputs by a stage in cascade mixnet. (a) Compromised stage 1 in RSA-based decryption/hybrid mixnet. (b) Compromised stage 1 in reencryption mixnet. Note that the same can be extended to ElGamal-based decryption mixnet and also to any compromised stage in the mixnet.***

and *add* inputs containing other messages. For instance, in an e-voting application, an adversary can modify the encrypted votes at a compromised stage of the mixnet, such that the tally is biased towards a contesting candidate. In secure e-voting, it then becomes crucial to provide measures that ensure integrity of the ballots and accuracy of the tally.

All these attacks involving corruption, deletion, and addition of inputs by compromised stages of the mixnet lead us to the analysis of the attacks on integrity in cascade mixnets. A common robustness technique against such attacks, employed in cascade mixnets, is verifiability.

V. VERIFIABILITY MECHANISMS IN CASCADE TOPOLOGY

In mixnets, verifiability involves checking the correctness of the output of the mixnet or the output of each stage. The correctness can be analyzed based on the following criteria.

C1) The input batch has been transformed (decrypted or reencrypted) and permuted honestly.

C2) The messages in the input batch have not been corrupted.

C3) Inputs have not been added/deleted.

In order to ensure a correct mixnet output, the verifiability mechanism employed in the mixnet must satisfy all the three criteria. As shown in Fig. 9, this translates to the region of intersection of all three criteria, C1$\cap$ C2 $\cap$ C3. Based on the criteria satisfied, we can classify cascade mixnets into the following: sender verifiable, stage verifiable (SV), universally verifiable (UV), and conditionally universally verifiable (CUV). We first describe the sender verifiable cascade mixnet and the verifiability mechanism employed in it.

A. Sender Verifiable Cascade Mixnet

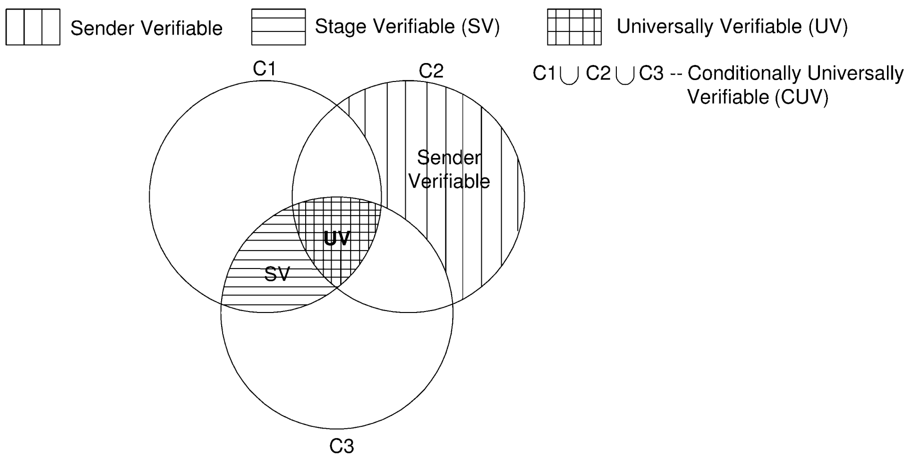
The verifiability mechanism in this mixnet only provides detection of corrupt messages at the mixnet output. The sender encrypts a redundancy as a checksum, such as a string of zeroes, along with the message as

$$E_K(K_s \| D_{K_s}(m \| 0^w), r) \tag{22}$$

where 0^w is a zero string of fixed (publicly known) length w. The public key K_s used by the sender does not reveal any identity information. Only the sender knows the private key K_s^{-1} and, hence, can compute $D_{K_s}(m \| 0^w)$.

After mixing is performed, the mixnet output batch will include the decrypted quantity, $K_s \| D_{K_s}(m \| 0^w)$. By encrypting with K_s, the receiver, as well as any other entity, can verify the integrity of the message m by checking the zero string. Any corruption of m modifies the zero string with a high probability and, hence, can be detected (criterion C2). Sender verifiable cascade mixnet of decryption/hybrid type are proposed in [1], and those of decryption/reencryption type in [33].

However, robustness against compromised stages is a problem in these mixnets. The redundancy cannot detect addition or deletion of inputs or the honest participation of each stage. Each sender i is expected to verify if its message $K_s \| D_{K_s}(m \| 0^w)$ is present in the mixnet output batch (thus called sender verifiable). If a sender i does *not* verify, then a compromised stage in the mixnet can replace the sender i input, without being detected. Hence,



Sender Verifiable
Stage Verifiable (SV)
Universally Verifiable (UV)
C1$\cup$ C2$\cup$C3 -- Conditionally Universally Verifiable (CUV)
C1
C2
C3
Sender Verifiable
UV
SV

Fig. 9. ***Venn diagram indicating classification of verifiable cascade mixnets, based on correctness criteria satisfied. C1) Input batch has been transformed (decrypted or reencrypted) as expected. C2) Messages in input batch have not been corrupted. C3) Inputs have not been added/deleted.***

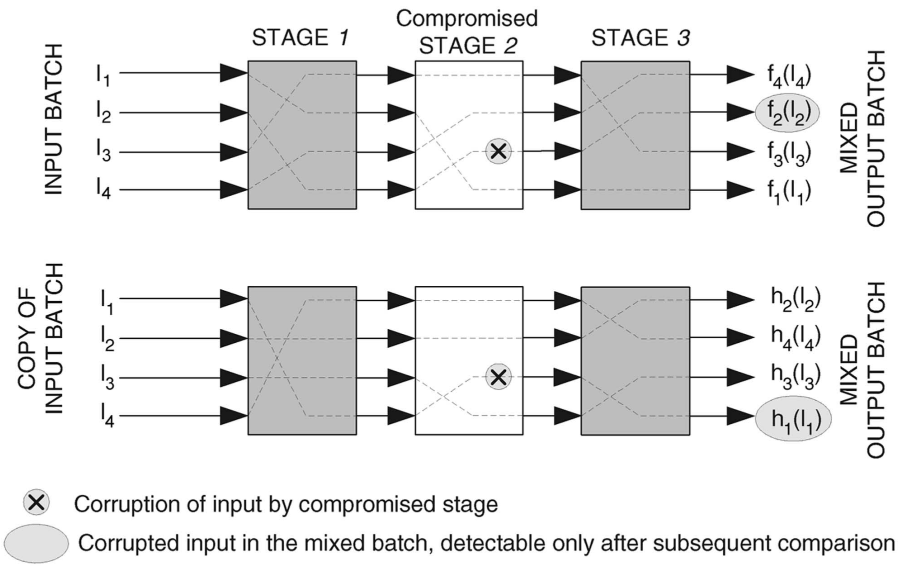


Fig. 10. ***Mixing of input batch and its copy. Mixnet contains three stages, where stage 2 is compromised.*** $f_i(I_i)$, $h_i(I_i)$ ***denote random transformation of the input*** I_i, $i = 1, 2, 3, 4$, ***during mixing of input batch and its copy, respectively. Compromised stage 2 may not locate same input in mixed copy batch obtained from stage 1 and, hence, corrupts a different input. Corrupted inputs are detected in a comparison step involving two mixed output batches.***

as seen in Fig. 9, the sender verifiable mixnet can only satisfy the hashed area in C2 and cannot satisfy the other two criteria. Another disadvantage of this mixnet is that the compromised stages responsible for the corrupted inputs cannot be identified. These weaknesses are addressed in the stage verifiable mixnets.

B. Stage Verifiable (SV) Cascade Mixnet

In these mixnets, the stages verify the mixnet output batch on their own, hence, not requiring the unrealistic sender verification. The stages together execute additional subprotocols to ensure correctness of the mixnet output batch. A minimum number of zero-knowledge proofs, and other cryptographic techniques, are used to provide robustness. The ElGamal-based reencryption type stage verifiable (SV) cascade mixnets are described first.

1) Stage Verifiable Reencryption Cascade Mixnets: A description of the main techniques used in the SV mixnet protocols introduced in [44], [46] is given as follows.

1) *Mixing Copies of the Input Batch*: If the mixnet performs mixing on c copies of an input batch, then it becomes difficult for a compromised stage to corrupt the same input in all the copies (duplicate batches). This is because the mixing operation on the duplicate batches will be different, with random transformations (exponentiation or reencryption) and random permutations. If a compromised stage corrupts an input, then it is unlikely (with increase in c) that it can locate and corrupt the same input in all the remaining duplicate batches. As illustrated in Fig. 10, where $c = 1$, the compromised stage 2 trying to corrupt input related to I_2 in the copy batch corrupts a different input related to I_1. Next, by revealing secrets and comparing the mixed output batches, the mixnet can detect the corrupted inputs $f_2(I_2)$, $h_1(I_1)$. In general, this technique provides robustness if there is a compromise in stage 2 to n of the cascade mixnet.
2) *Repetition of Mixing on the Input Batch*: The mixing of duplicate batches does not provide robustness against a compromised *first stage* (stage 1) of the mixnet. In a cascade mixnet, since the first stage initiates the mixing, it can trivially locate and corrupt the same input in all the duplicate batches. In Fig. 10, stage 1 can corrupt I_2 in the input batch, as well as the copy batch. However, once the input batch and its copies have been mixed, if the mixnet has to repeat mixing on the resulting mixed output batches, then a compromised first stage can be detected. As illustrated in Fig. 11, mixing is performed on an input batch and then the mixnet repeats mixing on the resulting mixed output batch. Therefore, a compromised stage 1 is unable to locate target inputs in the batch received for repeat mixing. Any corruption of inputs (e.g., I_4) during the first mixing of the input batch as well as during repeat mixing can be detected as seen next.

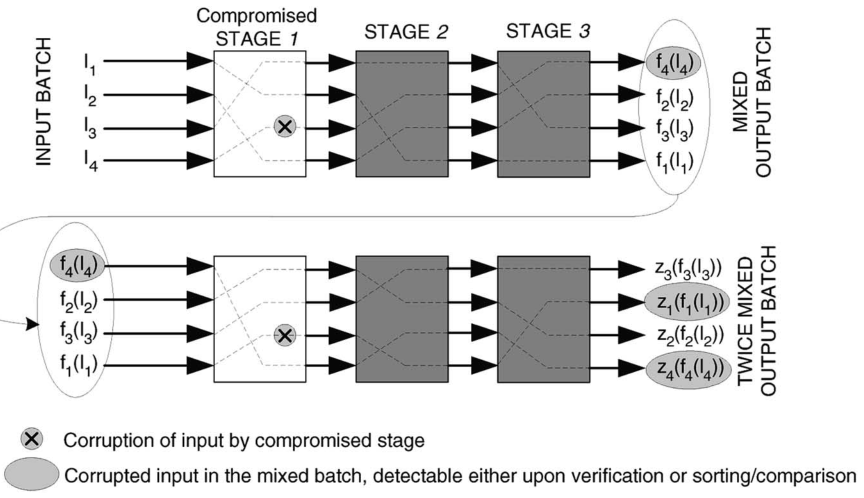


Fig. 11. ***Repetition of mixing of input batch. Corruption of target input I_4 in first mixing is detected upon verification that occurs after repetition of mixing. On the other hand, compromised stage 1 is not able to locate target inputs in batch received from stage 3 for repeat mixing. Corruption of inputs during repeat mixing, e.g., I_1, can be also detected by sorting and comparison protocol. $f_i(I_i)$, $z_i(f_i(I_i))$ denote transformations of I_i, $i = 1, 2, 3, 4$, during first mixing and repeat mixing, respectively.***

3) *Stages Revealing Secrets and Comparison of Batches*: By making the stages reveal secrets used in the mixing operation of the input batch and its copies, it is possible to detect any compromised stages that may perform incorrect mixing. For an illustration of the main idea, consider the mixnet in Fig. 11 to be a reencryption cascade mixnet. After mixing, each stage reveals the secrets used to generate the final transformations $f_i(.)$, $z_i(.)$, $i = 1, 2, 3, 4$. If the compromised stage 1 has corrupted the input (related to I_4) $E_K(m, r + r_1)$ as

$$\begin{aligned} E_K(mm', r + r_1 + r_2) &= (g^{r+r_1+r_2} \| K^{r+r_1+r_2} mm'), \\ &= \left(g^{r+r_1+r_2} \| g^{(r+r_1+r_2)d} mm'\right). \end{aligned} \tag{23}$$

Since g is a generator of the group Z_p^* [28], $m' = g^x \in Z_p^*$ for some x, and (23) can be rewritten as

$$\left(g^{r+r_1+r_2} \| g^{(r+r_1+r_2)d} g^x m\right) = \left(g^{r+r_1+r_2} \| g^{(r+r_1+r_2)d+x} m\right). \tag{24}$$

On revealing the secret exponent r_2, the g^x factor can be detected, hence, leading to detection of the compromised stage 2 having corrupted the input $E_K(m, r + r_1)$. However, revealing the secrets breaches anonymity of input batch and defeats the whole purpose of the mixnet, which is to anonymize the input batch. Therefore, the approach taken is to reveal the secrets *only after* the repetition of mixing on the mixed output batches. But the revealed secrets will then have to be used by the mixnet to obtain sets of quantities related to the output batches resulting from mixing of the input batch and its copies, as well as from repetition mixing [44], [46]. These sets of quantities are sorted and compared, and based on the comparison, if there are *no* anomalies, then the mixed output batches are verified to be correct.

4) *Tracing the Compromised Stages*: On detection of an anomaly in the mixnet output batch, a verified trace back from the mixnet output towards the mixnet input is done. Each stage reveals its secrets and the operation of the stage is verified by the remaining stages. In order to preserve anonymity of the input batch, it then becomes crucial that the mixnet output batch must *not* be decrypted to reveal the message [46]. In an alternate approach, the input batch contains the blinded messages [44]. Hence, even if a compromised stage 1 is present in the mixnet, the anonymity of any corrupted input (possibly corrupted by stage 1) is not lost, despite having traced it from the mixnet output back to the input of stage 1.

5) *Stage Proof Generation and Verification*: All the techniques discussed previously try to ensure that each stage has not corrupted an input (criterion C2) and has not deviated from its normal mixing

operation (criteria C1). Next, to ensure that at each stage the output batch contains the input batch, i.e., to ensure that a compromised stage has not added or deleted any inputs (criteria C3), the ZK proofs are used. After mixing, when no anomalies are detected in the comparison of batches, each stage proves that the product of the input batch and the product of the output batch are the same by using the random exponents employed in repetition mixing [44], [46]. ZK proofs may also be used before mixing, when a stage may be required to prove knowledge of a share of the private key of the mixnet, used in the exponentiation of messages [44]. The proofs generated by each stage j are verified by the remaining stages during the mixnet protocol. If correct, then stage j is verified to be uncompromised. If all stages are uncompromised, then the correctness of the mixnet output batch is verified. However, if a compromised stage is detected, then a recovery subprotocol is initiated as follows.

6) *Reacting to a Compromised Stage Detection*: If *before mixing*, the proof of a stage j is verified to be incorrect, then the stage can be simulated by the remaining stages or simply removed/ignored. Whereas, if incorrectness in the proof is detected (or a compromised stage is traced) after mixing, the compromised stage is removed/ignored and the mixnet protocol is restarted (see Fig. 13).

While the previous techniques have been employed in the ElGamal-based reencryption cascade mixnets of [44], [46], there are some important differences between the two mixnets themselves. In [44], initially the input batch is blindly decrypted to obtain the blinded messages, which ensures privacy of the input batch as described. The decryption of input batch before mixing also provides for computational efficiency, since each stage j will only need to exponentiate a blinded message m_b as

$$\left(m_b^{\prod_{a=1}^{j-1} r_a}\right)^{r_j} \tag{25}$$

where $r_1, r_2, \ldots, r_j$ are the random exponents used by stages $1, 2, \ldots, j$, respectively. In [46], decryption is performed only *after* mixing. The protocol makes use of reencryption of the encrypted messages. More specifically, stage j performs the following multiplications on the inputs obtained from stage $j-1$:

$$\left(g^{R+\sum_{a=1}^{j-1} r_a} g^{r_j} \| m K^{R+\sum_{a=1}^{j-1} r_a} K^{r_j}\right) \tag{26}$$

where R is a random exponent used by the sender. Note that in terms of computations, the stage j can precompute random g^{r_j}, K^{r_j}, $j = 1, 2, \ldots, l$ and perform multiplication with the input batch (containing l inputs) received. But in the case of (25), the stage has to perform exponentiation of its input batch. This makes the online computational complexity of the mixnet in [46] better than that of [44].

The integrity of the mixnet in [44] can be broken, as shown in [47]. A compromised stage can make use of the homomorphic property of ElGamal encryption (in Appendix A.3) and modify *all* the inputs in a batch, without being detected by the mixnet protocol. However, this attack is not possible in [46], since it makes use of dummy inputs. These dummy inputs are generated jointly by the n stages, and only a collusion of all the stages can locate them in the mixnet input batch. But, as pointed out in [48], if the location of the dummy inputs is known to the first stage of the mixnet, then it will be able to break the integrity of the mixnet in [46]. Therefore, the SV reencryption mixnets can only satisfy the correctness criteria C1 and C3.

While the previous mixnets make use of one or more ZK proofs, the SV reencryption cascade mixnet proposed in [47] does not require ZK proofs and, moreover, satisfies all three correctness criteria. Consequently, in Fig. 9, the SV mixnets belong to the intersecting region $C1 \cap C3$ that includes $C1 \cap C2 \cap C3$. However, the approach requires $t-1$ stages participating as verifiers for each stage j in the cascade mixnet. These $t-1$ verifiers check the operation of their assigned stage j in the cascade mixnet. During the mixnet protocol, each stage j in the cascade mixnet performs reencryption mixing and reveals the random exponents to its $t-1$ verifiers, who then check the stage output. If determined to be incorrect, then the output of stage j is ignored and output of stage $j-1$ is transferred directly to stage $j+1$. We note that this protocol, unlike in [44] and [46], does not require a restart upon detection of a compromised stage. For a cascade mixnet containing t stages, if we consider the verifiers, there will be a total of $(t-1) \times (t-1)$ stages in the mixnet. However, robustness is still only $t-1$, since if one verifier in each stage of the cascade mixnet is compromised, then these t compromised verifiers can reveal the secrets and, hence, breach the anonymity of the mixnet.

2) Stage Verifiable Hybrid Cascade Mixnets: The cascade mixnets in [44] and [46], [47] have been proposed only for ElGamal-encrypted input batches. In Section IV-E, it was mentioned that ZK proofs may not be feasible for decryption/hybrid mixnets due to the multiple layer encryptions and symmetric key encryptions. However, in [24] and [49], the ZK proofs are enabled through the use of public keys in the symmetric key algorithms as explained in the following. In [49], a hybrid cascade mixnet is proposed, based on the robustness techniques of [47]. The approach taken in [49] was improved in [24], which makes use of a relatively efficient ZK proof, and robustness is also achieved without revealing any secrets or using additional verifier stages (required in [49]). The techniques used in the hybrid cascade mixnet protocol in [24] are described as follows.

1) *Public Keys for Symmetric Encryption*: The main idea of this technique is to use the public key of a stage to derive the symmetric key that the stage has to use to decrypt a layer from the onion. If $K_j = g^{d_j}$ is a public key of a stage j, then the stage receives from stage $j-1$, a key generator

$$\text{keygen}_{j-1} = g^{r \sum_{a=1}^{j-1} d_a} \tag{27}$$

and an onion of the form

$$\text{forwarding onion}_j = E_{k_j}\left[A_{j+1} \| \text{forwarding onion}_{j+1}\right] \times \| \mathcal{H}_{k'_j}\left(E_{k_j}\left[A_{j+1} \| \text{forwarding onion}_{j+1}\right] \| 0^w\right). \tag{28}$$

Note that the sender initially includes $\text{keygen}_0 = g^r$, where r is a random string, with the forwarding onion_1. The key generator keygen_{j-1} is updated to keygen_j, by stage j, exponentiating it as in (27) with its private key d_j. In fact, this operation is used in the ZK protocol technique explained in the stage proof generation phase of the mixnet protocol. The symmetric keys $\{k_j, k'_j\}$ are derived from keygen_{j-1} using two more private keys from stage j. Hence, each stage has three public keys that are used initially by the sender in the onion encryption.

2) *Message Authentication Code*: As seen in (28), each layer of the onion contains a message authentication code (MAC) [28]. The $\text{MAC}_j = \mathcal{H}_{k'_j}(.)$, provides verification of integrity of each input in the batch received by stage j from stage $j-1$. Note that only stage j is able to verify the MAC, since the key for the MAC can be obtained only with the private key of stage j. Hence, if stage $j-1$ is compromised and corrupts any input, then stage j can detect it by checking the MACs.

3) *Stage Proof Generation, Proof Verification, and Reaction to a Claim of a Compromised Stage*: In addition to MACs, each stage $j-1$ generates a proof of knowledge of the private key used in the exponentiation of the key generator as seen previously. This proof is verified by stage j and ensures that the output batch from stage $j-1$ contains its input batch. A computationally efficient ZK proof [24] is used for this purpose. Hence, the MACs and the ZK proofs together provide robustness against a compromised stage $j-1$. Fig. 12 illustrates the operation of the hybrid cascade mixnet in [24] containing three stages. If a stage j in the hybrid mixnet verifies stage $j-1$ to be compromised, then a threshold of t stages in the cascade mixnet (excluding stage $j-1$) jointly generate the private keys of stage $j-1$ and verify its mixing operation. For instance, in the mixnet of Fig. 13 where $t=2$, if stage 1 is compromised, then stages 2 and 3 would need to verify its operation by reconstructing its private key. If stage $j-1$ is indeed compromised, then it is simulated by the t stages and the mixnet protocol proceeds. Note that the simulation of the compromised stage $j-1$ is required, since decryption with the private key of stage $j-1$ is necessary. But if stage $j-1$ is found to be uncompromised, then there can only be two possibilities: 1) a compromised stage j which is then simulated or 2) one or more inputs to the stage $j-1$ may be incorrect, which initiates tracing backwards until the incorrect inputs are located and deleted from the mixnet input batch, followed by repetition of the mixing.

4) *Simulation of the Last Stage*: Despite the techniques, a compromised stage n of the hybrid cascade mixnet can still break the integrity of the inputs in the batch received from stage $n-1$ without being detected. This is because there is no stage $n+1$ to verify operation of stage n. Therefore, to address this weakness, stage $n+1$ is simulated by the n stages. Joint decryption with the private key of stage $n+1$ is performed to obtain the mixnet output batch. For example, in Fig. 12(c) and (d), stage 4 is simulated by the three stages to finally obtain the mixnet output batch.

Though the hybrid cascade mixnet protocol in [24] satisfies all three criteria (C1 ∩ C2 ∩ C3), the anonymity can still be breached by an active adversary that is allowed to participate as the sender in $n+1$ separate mixing iterations of the mixnet. The attack, as shown in [50], is enabled due to the reaction to the claim of a compromised stage in the mixnet protocol. As seen previously, in this technique, when an error in the output of stage $j-1$ is detected by stage j, the t stages verify the operation of stage $j-1$. During this verification a set of quantities (especially the keys of the MACs in the input batch of stage $j-1$) are revealed to all the stages. Hence, any single compromised stage in the hybrid mixnet can help the adversary to obtain the revealed quantities and, thereby, launch the anonymity attack in [50].

3) *Limitations of Stage Verifiable Cascade Mixnets:* An important tradeoff between the SV reencryption mixnet, and the SV hybrid mixnet is the reaction to a compromised stage. As illustrated in Fig. 13, in the case of a hybrid mixnet, the detected compromised stages have to be simulated, but the protocol is not restarted. However, as seen earlier, in an SV reencryption mixnet the compromised

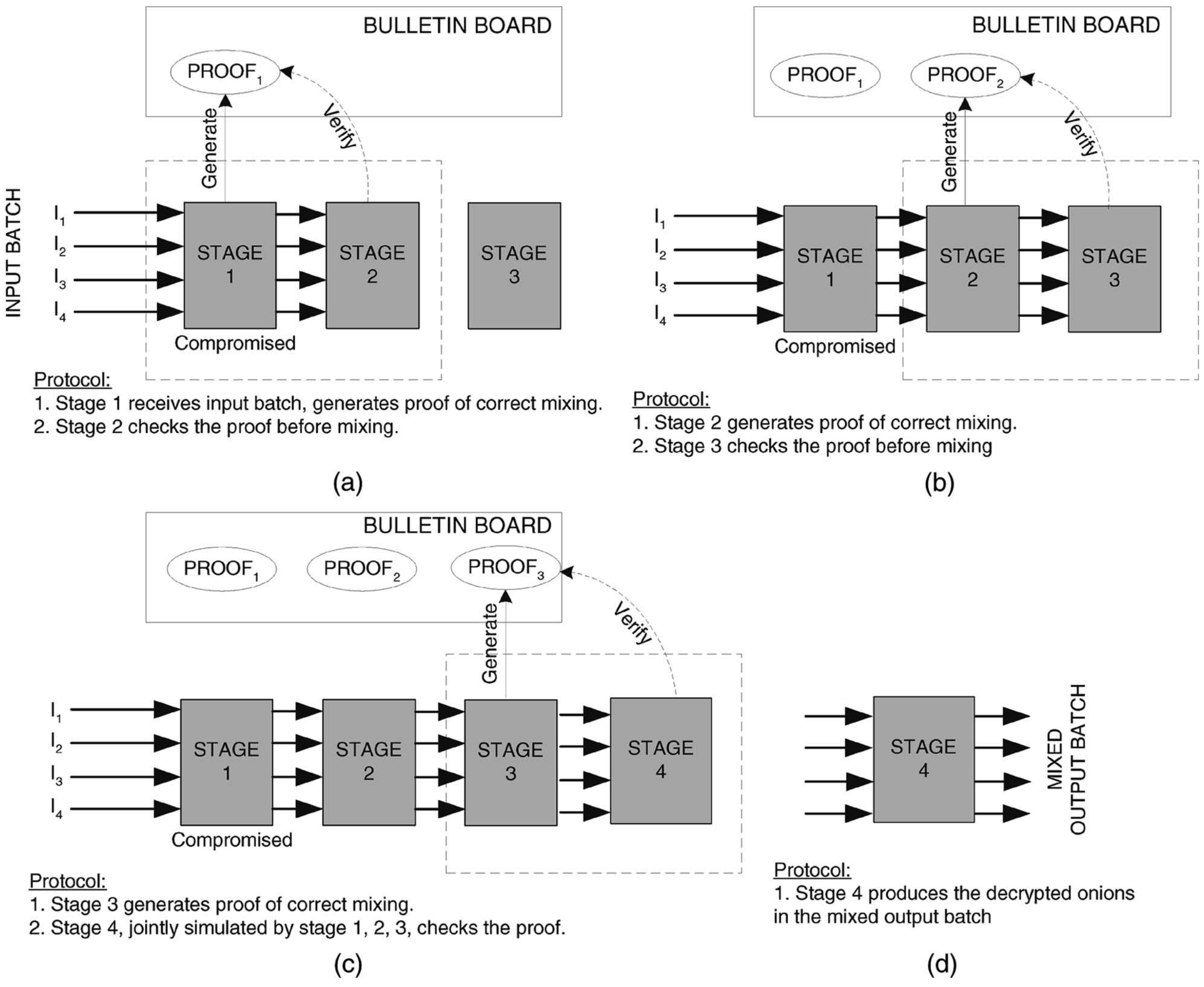

BULLETIN BOARD
PROOF$_1$
PROOF$_2$
PROOF$_3$
Generate
Verify
INPUT BATCH
I_1
I_2
I_3
I_4
STAGE 1
STAGE 2
STAGE 3
STAGE 4
Compromised
MIXED OUTPUT BATCH
Protocol:
1. Stage 1 receives input batch, generates proof of correct mixing.
2. Stage 2 checks the proof before mixing.
(a)
Protocol:
1. Stage 2 generates proof of correct mixing.
2. Stage 3 checks the proof before mixing
(b)
Protocol:
1. Stage 3 generates proof of correct mixing.
2. Stage 4, jointly simulated by stage 1, 2, 3, checks the proof.
(c)
Protocol:
1. Stage 4 produces the decrypted onions in the mixed output batch
(d)

Fig. 12. ***Stage verifiable hybrid mixnet protocol, such as in [24]. (a) Stage 2 receives mixed input batch from stage 1, as well as a proof. Stage 2 verifies the MACs of onions in its input batch, checks proof, and performs mixing. (b) Stage 3 repeats same operations. (c) Stage 4 is jointly simulated by three stages to verify operation of stage 3. (d) Stage 4 performs decryption of the last layer of onions in input batch.***

stage can be simply removed, but it still requires an unavoidable restart of protocol. The only exception is the SV reencryption mixnet in [47] where a protocol restart is avoided by involving the additional $t-1$ verifiers for each stage in the mixnet.

A major drawback of the SV cascade mixnets is their limited robustness against attacks. A SV cascade mixnet is robust only against a maximum of $n-1$ compromised stages, and if sharing of private keys is used in the mixnet, then robustness is only $t-1$. A collusion of t, or all n stages, can corrupt the input batch without being detected in the mixnet protocol. For example, consider the mixnet in Fig. 13, where $n=3$, and threshold $t=2$. If more than one stage $(t-1=1)$ is compromised, then the output batches can contain corrupted inputs without being detected by the mixnet protocols.

For critical, secure applications, including electronic voting, a higher guarantee for the integrity of the mixnet output batch must be provided. This leads to the design of universally verifiable cascade mixnets.

C. Universally Verifiable (UV) Cascade Mixnet

In a universally verifiable (UV) cascade mixnet, even if all the n stages are compromised, they cannot produce an incorrect output batch, hence, always satisfying the three criteria of correctness $(C1 \cap C2 \cap C3)$. The main idea behind the design of a UV mixnet is that each stage must prove that each quantity in its output batch corresponds to an unique quantity in its input batch, without disclosing the relationship. Such a ZK proof provides correctness of the output batch at each stage, but requires up to $O(l)$ exponentiations in the computations at each stage, l being

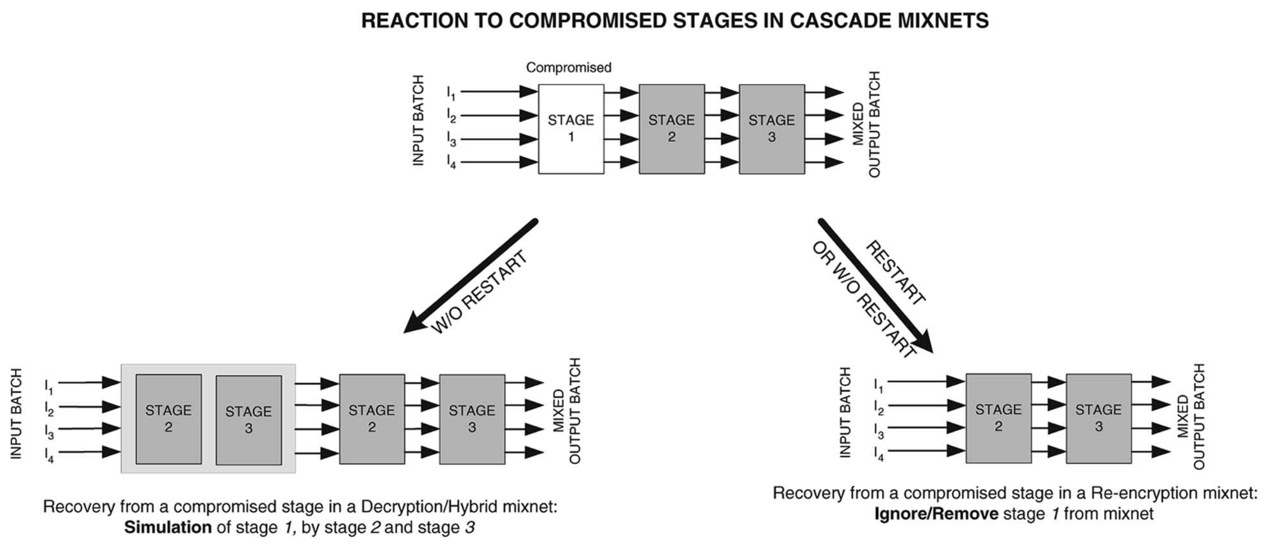

REACTION TO COMPROMISED STAGES IN CASCADE MIXNETS
Compromised
INPUT BATCH
I1 I2 I3 I4
STAGE 1
STAGE 2
STAGE 3
MIXED OUTPUT BATCH
W/O RESTART
RESTART OR W/O RESTART
Recovery from a compromised stage in a Decryption/Hybrid mixnet: Simulation of stage 1, by stage 2 and stage 3
Recovery from a compromised stage in a Re-encryption mixnet: Ignore/Remove stage 1 from mixnet

Fig. 13. ***Reaction to compromised stages in stage verifiable cascade mixnets. These mechanisms are also generally applicable to other verifiable cascade mixnets.***

the input batch size. The UV mixnet protocol is otherwise similar to the SV mixnets. To address reaction to compromised and faulty stages, techniques of SV mixnets, as shown in Fig. 13, may be used. The approaches to stage proof generation and verification in the UV mixnet protocol are shown in Fig. 14.

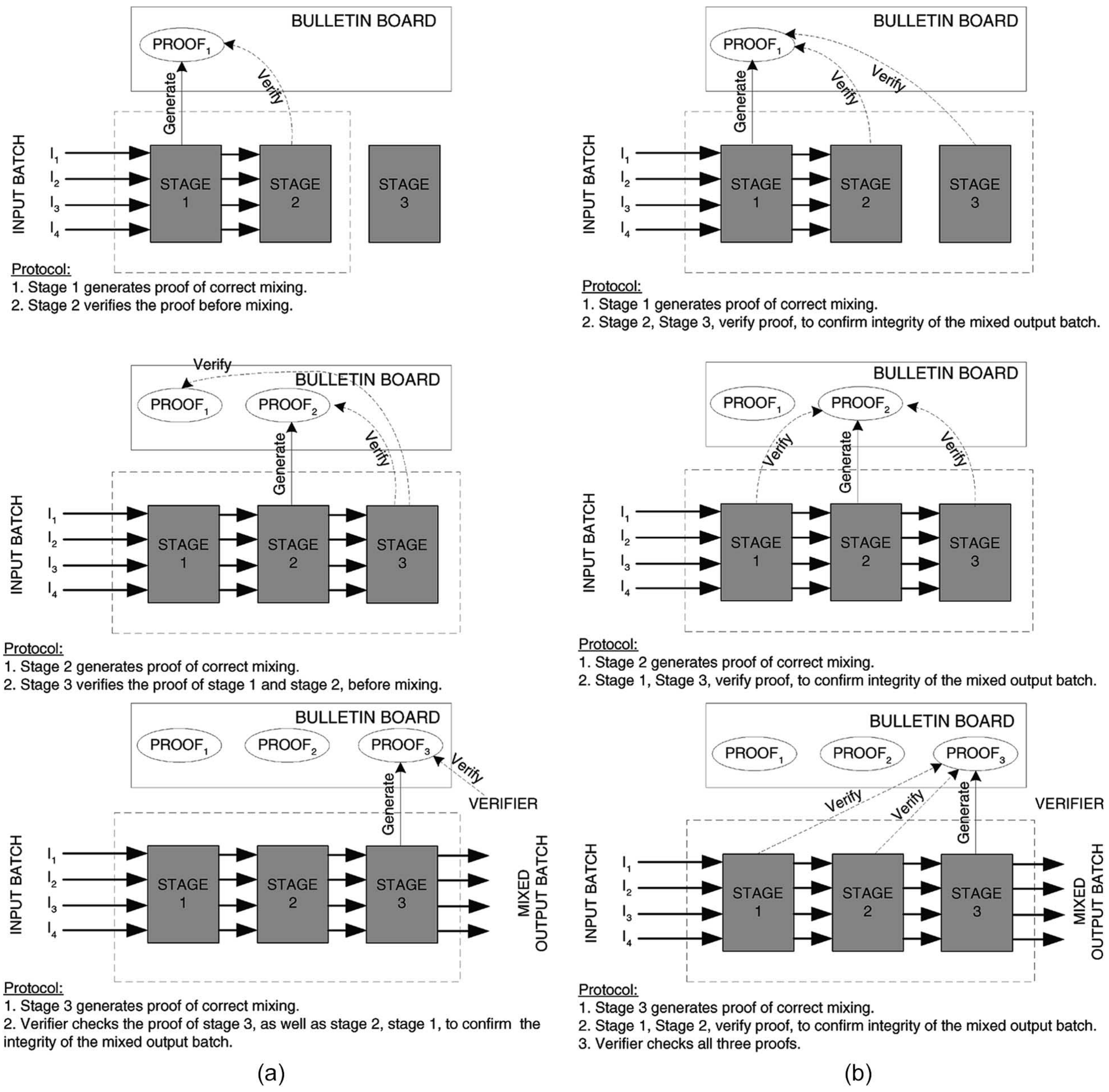
Universal verifiability in the mixnet comes from the fact that the verification of the proofs can be additionally performed by any external entity. This entity, which we refer to as the verifier, can guarantee correctness of the mixnet output even if all stages have been compromised, since any incorrect proofs can be detected. Note that because of the computationally expensive proofs needed in UV mixnets, the efficiency of the mixnet protocol is heavily dependent on these proofs. The UV cascade mixnets that have been developed are based on the ElGamal cryptosystem and are mostly of the reencryption type. A concise presentation of these mixnets is given as follows.

1) Nonrobust UV Cascade Mixnets: After the ElGamal-based reencryption and decryption mixnets introduced in [33] were shown in [42] as lacking robustness against compromised and faulty stages, the UV mixnets were introduced in [40]. Though the UV mixnets in [40] are made (partially) robust against attacks, using ZK proofs to detect any compromised stage, in [51], it was shown that the anonymity of the mixnets in [40] are breakable. Moreover, since the mixnet protocols in [40] do not include any mechanism for recovery from compromised and faulty stages, they are not fully robust to attacks and failures. Reencryption mixnets in [39] and [52] and the decryption mixnet in [53] overcome the robustness weaknesses.

2) Robust UV Cascade Mixnets: The mixnet in [52] is similar to the reencryption mixnet in [39] but with relatively more efficient proofs. For robustness, as in the SV mixnets, after mixing is performed, the group of n stages jointly verify that each stage has performed a correct operation. The stages generate a joint proof [52] that can be verified later by any entity. Hence, even if all n stages are compromised, it can be detected by verification of this joint proof. As in Fig. 13, if detection of compromised stages occurs in the reencryption mixnet, then the mixing protocol is restarted without the compromised stages.

In [53]–[55], mixnet protocols are constructed using permutation networks [56] that enable the use of efficient proofs. In contrast to [52], the verification of the proofs is performed during mixing. In [53], addition to a reencryption mixnet, a decryption mixnet protocol, is also proposed, where each stage verifies the proofs of all the previous stages before proceeding with the mixing of its input batch. This is illustrated in Fig. 14(a). Hence, only one stage needs to be involved in the verification of the proofs at each step. However, when stage j detects a compromised stage a, threshold t stages are required to simulate the stages from the compromised stage a until stage $j-1$. On the other hand, in the reencryption mixnet in [53], each stage mixes and produces a proof, which is then verified by all the remaining stages, as illustrated in Fig. 14(b). The advantage with this approach is that unlike in the SV reencryption mixnets [44], [46], a detected compromised stage can be ignored, and a restart of the reencryption mixnet protocol is *not* necessary. The input batch of the compromised stage $j-1$ can be transferred directly to the next stage j for further mixing.

However, note that for a fair comparison between the ElGamal-based UV reencryption mixnet and the ElGamal-based UV decryption mixnet, the decryption phase involved in the reencryption mixnet must be taken into account. This phase also requires ZK proofs from each stage, to prove exponentiation with the share of the private key. Thus, as observed in [53], the decryption mixnet provides relatively lower latency compared to the reencryption mixnet. But in the presence of a compromised stage, the latency of the decryption mixnet degrades significantly compared to a reencryption mixnet, due to the intensive recovery mechanisms involved. In [57], this weakness has been addressed, using a trusted verifier to verify the proof of each stage during mixing in the decryption mixnet. The trusted verifier receives the mixed output batch from each stage, verifies proof of the stage, and then forwards the batch to the next stage. On detection of a compromised stage j, the trusted verifier simulates the stage j. However, any compromise in the single trusted verifier leads to failure of the mixnet in [57]. It is also worth noting here



Protocol:
1. Stage 1 generates proof of correct mixing.
2. Stage 2 verifies the proof before mixing.

Protocol:
1. Stage 1 generates proof of correct mixing.
2. Stage 2, Stage 3, verify proof, to confirm integrity of the mixed output batch.

Protocol:
1. Stage 2 generates proof of correct mixing.
2. Stage 3 verifies the proof of stage 1 and stage 2, before mixing.

Protocol:
1. Stage 2 generates proof of correct mixing.
2. Stage 1, Stage 3, verify proof, to confirm integrity of the mixed output batch.

Protocol:
1. Stage 3 generates proof of correct mixing.
2. Verifier checks the proof of stage 3, as well as stage 2, stage 1, to confirm the integrity of the mixed output batch.

Protocol:
1. Stage 3 generates proof of correct mixing.
2. Stage 1, Stage 2, verify proof, to confirm integrity of the mixed output batch.
3. Verifier checks all three proofs.

Fig. 14. ***Proof generation and verification during mixing, in UV cascade mixnets. (a) Each stage verifies proof of all previous stages. (b) Proof generated by a stage is verified by all remaining stages in mixnet.***

that despite their robustness, some weaknesses in the integrity of UV mixnets presented in this section (including the following) have been identified in [58] along with countermeasures.

3) Efficiency of UV Cascade Mixnets: The reencryption mixnet protocols in [53]–[55] are not scalable in the batch size l, with the verifiability mechanism computational complexity being $O(nl \log_2 l)$ for a n stage mixnet. On the other hand, the mixnet protocols in [52] are limited by the inefficient ZK proofs, with verifiability mechanism computational complexity $O(nlk)$, where k is the number of repetitions of the ZK proof [52]. Computationally efficient ZK proofs for reencryption mixnet protocols have been developed in [59]–[61], and an computationally efficient ZK proof has been provided for ElGamal-based decryption mixnet in [57]. These proofs are linear in the size of the input batch, i.e., the verifiability mechanism with these proofs can have reduced complexity of $O(nl)$. Further, in [57], it is claimed that the overall computational complexity of the mixnet protocol (including decryption of mixnet output) based on [60] is lower compared to [59] and [61].

As efficient as it is, in the absence of any compromised stages in the mixnet, the universal verifiability mechanism taxes the cascade mixnet efficiency and degrades performance. While in the case of stage verifiable mixnets, relatively efficient performance can be achieved under such a scenario. Yet, in UV mixnets, the presence of compromised stages does not require a restart of the protocol which is not necessarily true for all SV mixnets. Moreover, UV mixnets allow for external verification of the mixing, unlike SV mixnets. There is another category of verifiability mechanisms in cascade mixnets that attempts to achieve a balance between these two approaches, called conditionally universal verifiability mechanisms.

D. Conditionally Universally Verifiable (CUV) Cascade Mixnet

The verifiability techniques employed in CUV cascade mixnets provide certain probabilistic guarantees for the correctness of the mixnet output batch that may not always satisfy correctness of the entire output batch of the mixnet. Hence, as indicated in Fig. 9, the CUV mixnets may satisfy from one to all the three criteria of correctness, i.e., the region $C1 \cup C2 \cup C3$. Similar to a UV mixnet, any entity can verify the mixnet output batch, but an anomaly is detected only with a certain probability. We now describe the conditionally universal verifiability techniques that have been proposed in the literature.

1) Random Partial Checking (RPC): This technique was proposed in [2] and does *not* require any ZK proofs. It is very generic and can be used in any type of cascade mixnet. As in SV mixnets, the technique can be used in a subprotocol after mixing has been performed on the input batch to detect any compromised stage. The main idea is that the stages are made to directly reveal the correspondences between a random selection from their input batch or output batch and also the cryptographic quantity involved in the transformation of the selections.

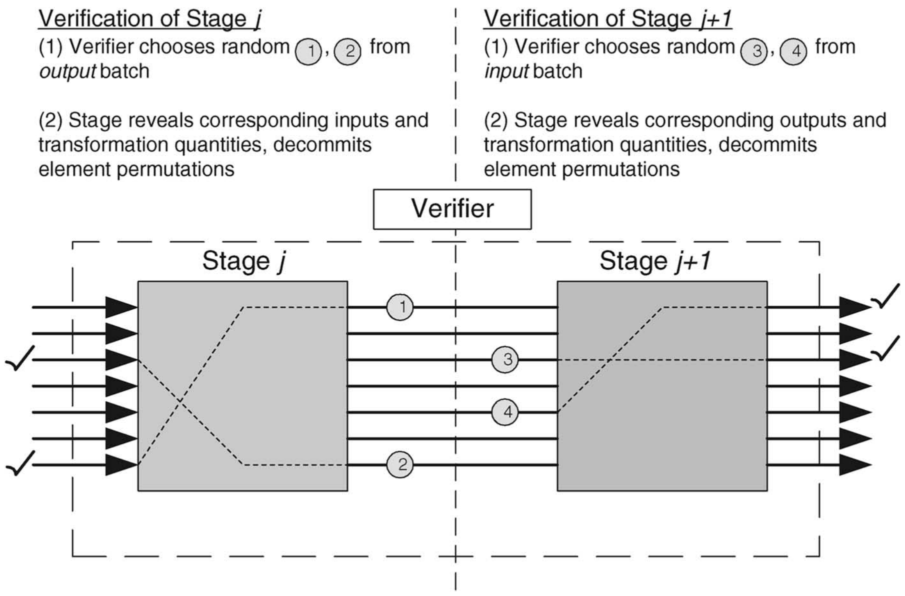

Verification of Stage j
(1) Verifier chooses random 1, 2 from output batch
(2) Stage reveals corresponding inputs and transformation quantities, decommits element permutations
Verification of Stage j+1
(1) Verifier chooses random 3, 4 from input batch
(2) Stage reveals corresponding outputs and transformation quantities, decommits element permutations
Verifier
Stage j
Stage j+1
1
2
3
4

Fig. 15. ***RPC: Stage j and stage $j+1$ are paired together as shown, such that output batch of stage j is input batch of stage $j+1$. Verifier selects a random set of outputs of stage j and makes it reveal corresponding inputs. Verifier then selects a random set of outputs of stage j (but different from already selected outputs) and makes stage $j+1$ reveal its corresponding outputs.***

A graphical illustration of random partial checking is shown in Fig. 15. For the purpose of revealing the secrets, the stages are paired together (assuming that n is even), such that the output batch of the first stage j is the input batch of second stage $j+1$ of each pair. Prior to mixing, stage j commits to a permutation $\pi_j = (\pi_{1,j}, \pi_{2,j}, \ldots, \pi_{l,j})$, and stage $j+1$ commits to $\pi_{j+1}^{-1} = (\pi_{1,j+1}^{-1}, \pi_{2,j+1}^{-1}, \ldots, \pi_{l,j+1}^{-1})$. Note that π_j, π_{j+1} are the permutations used in the mixing operation by stage j and stage $j+1$, respectively. During the subprotocol, a random set of outputs of the first stage j is selected as RS_j. The inputs corresponding to the selected outputs, and also the random exponents or keys used for their transformations, are revealed by the stage j. The proof is provided by stage j in the form of a decommitment of the element permutations of the revealed inputs $\{\pi_{i,j} : i \in RS_j\}$. Next, a random set of inputs of the second stage $j+1$ is selected as RS_{j+1}. However, these selected inputs must be from the complementary set of the revealed outputs in RS_j of the stage j. This is crucial so that an output from stage $j+1$ cannot be traced back to the input of stage j, thereby helping to preserve anonymity. Stage $j+1$ then reveals the outputs corresponding to the selected inputs. The process of revealing quantities in the transformation and the decommitment of $\{\pi_{i,j+1}^{-1} : i \in RS_{j+1}\}$ is repeated by stage $j+1$. This type of verification is repeated for the remaining $(n-2)/2$ pairs of stages in the cascade mixnet.

We note that the RPC technique sacrifices the anonymity that an input otherwise gains from batch size. Randomly correlating quantities prevents a compromised stage from predetermining which inputs to corrupt. However, at the same time, random selection of the correspondences may not detect a compromised stage, since the selection may not include the corrupted inputs or the added inputs. Therefore, the RPC technique provides only a probabilistic guarantee of the correctness of the mixnet output batch. Yet, it is quite efficient in computation, since *no ZK proofs* are involved. We will now describe an approach that makes minimal use of ZK proofs but offers a stronger guarantee of correctness.

2) Optimistic Verification: This technique, proposed in [62], is applicable to a reencryption cascade mixnet. After mixing is performed by all stages, a set of random quantities is selected from the output batch of each stage. Next, each stage provides two computationally efficient ZK proofs [44]: one proves that the product of the input batch of the stage is the same as the output batch of the stage, and the other proves that the selected quantities from output batch of the stage are from its input batch. Hence, the verification of the first proof detects any addition/deletion of inputs, while the verification of the second proof detects if any of the selected quantities have been corrupted within the stage. Note that unlike RPC technique, in optimistic verification the anonymity provided by the batch size is not lost, since the proof does not disclose the correspondences at any stage. However, as in RPC, the quantities selected may not be the corrupted quantities, hence, not being detected in the mixnet.

Consequently, a related but improved approach has been proposed in [63] where in addition to the optimistic verifiability mechanism, for the purpose of detection of corrupted inputs, the protocol uses cryptographic checksums (unkeyed hashes) [28]. Further, the approach makes use of a technique called double encryption to preserve the anonymity of corrupt inputs as seen in the following. Each input in the sender input batch contains three ElGamal encryptions

$$E_K(F, r) \| E_K(M, r') \| E_K(\mathcal{H}(F, M), r'') \tag{29}$$

where $F = g^R$, $M = mK^R$ are the ElGamal encryption components of the message. $\mathcal{H} : \{0,1\}^* \rightarrow G$ is a one-way hash function. After reencryption of the input batch, each stage j generates an efficient proof as in [62] to prove that the products of the output and input batches are the same.

Nevertheless, a compromised stage can make use of the homomorphic property of ElGamal encryption to corrupt the input batch and still generate a valid proof (described in Appendix A.3). The checksum detects such attacks as follows. After mixing, the stages together decrypt the mixnet output batch to reveal $(F, M, \mathcal{H}(F, M))$, corresponding to each input. Then, by verifying the checksum, any corrupt input can be detected. Because of the double encryption in (29), the anonymity of a corrupt input is protected, since the message m still remains encrypted and can be mixed again in a separate input batch.

Note that a cascade mixnet employing the techniques in [2] and [62] may not always satisfy all three criteria of correctness. However, they are the most computationally efficient of the verifiability mechanisms when there are no compromised stages, hence, incurring low latency in mixing. Under the condition that the inputs satisfy special encryption requirements, the reencryption cascade mixnet using the technique in [63] can satisfy *all* three criteria, with a much lesser number of computations than a UV cascade mixnet. Yet, upon detection of a corrupted input, unlike a UV mixnet where a restart can be avoided, the technique in [63] as well as in [2] and [62] require a restart of the mixnet protocol, which makes them relatively less efficient. These constraints justify the classification of these approaches as conditionally universally verifiable cascade mixnets.

Table 3 compares some of the most recent verifiable cascade mixnets in terms of their computational efficiency.

VI. PARALLELIZING CASCADE MIXNETS

While efficiency and low latency in verifiability mechanisms can be achieved, the basic cascade mixnet protocol

Table 3 Comparison of verifiable cascade mixnets. Approximate costs for sender, each stage, and receiver in n-stage mixnet in terms of exponentiations and multiplications, to verifiably mix l inputs. α—Security parameter in [62]

Scheme	Sender	Mixing	Verification	Receiver
SV [47]	2 exp.	$2l$ mult.	n exp.	$(2+4n)l$ exp.
UV [60]	2 exp.	$2l$ mult.	$18l(2n-1)$ exp.	$(2+4n)l$ exp.
UV [59]	2 exp.	$2l$ mult.	$8l(2n-1)$ exp.	$(2+4n)l$ exp.
CUV [2]	2 exp.	$2l$ mult.	$\frac{l(2n-1)}{2}$ exp.	$(2+4n)l$ exp.
CUV [62]	2 exp.	$2l$ mult.	$4n\alpha$ exp.	$(2+4n)l$ exp.
CUV [63]	6 exp.	$6l$ mult.	$6nl$ exp. + $2nl$ mult.	$(2+4n)l$ exp.

is inherently inefficient in its operation. In a decryption cascade mixnet, the sequential order of mixing results in inefficient use of the stages and in delay. This is mainly due to the fact that when the first batch of inputs arrives at stage 1, the remaining $n-1$ stages wait until they receive this input batch. The mixing process is pipelined only after stage n receives the input batch. However, in the case of a reencryption cascade mixnet, the sequential ordered mixing is not necessary, as was initially observed in [53]. In [64], this observation is utilized in the design of a reencryption mixnet, where the stages perform mixing in parallel. Such a mixnet achieves gain in latency due to the parallelizing technique explained as follows.

Each of the n stages is assigned a random subset of the input batch, i.e., each subset contains l/n inputs. Fig. 16, where $n=3$, threshold $t=3$, and $l=9$, provides a graphical description of the parallel mixing. The following is the protocol executed by the stages in the mixnet.

1) Mixing: Each stage mixes the assigned subset of size l/n. This operation occurs in parallel in all the n stages.
2) Rotation: Next, the stages perform $t-1$ rounds of rotations, where threshold $t \leq n$. Each rotation is a modulo operation $(\text{mod } n)$, where stage $j-1$ transmits its mixed output batch to stage j, while stage j transfers its mixed output batch to stage $j+1$, and so on. Stage n transfers its output batch to stage 1. Each stage j on receiving the output from stage $j-1$ $(\text{mod } n)$ mixes and transfers to next stage $j+1$ $(\text{mod } n)$ during the next rotation. Note that all the stages are processing in parallel for $t-1$ rounds.
3) Random Exchange: At the end of the $t-1$ rounds, each stage retains a random fraction $(l/n)/n$ of its output and sends equal random portions of the remaining outputs to each of the $n-1$ stages. Hence, each stage j receives l/n^2 inputs from each of the remaining stages, to obtain a total of $nl/n^2 = l/n$ inputs.
4) Next, steps 1 and 2 are repeated, where after mixing, another $t-1$ rounds of rotation are performed. The resulting output from the n stages is the final mixnet output.

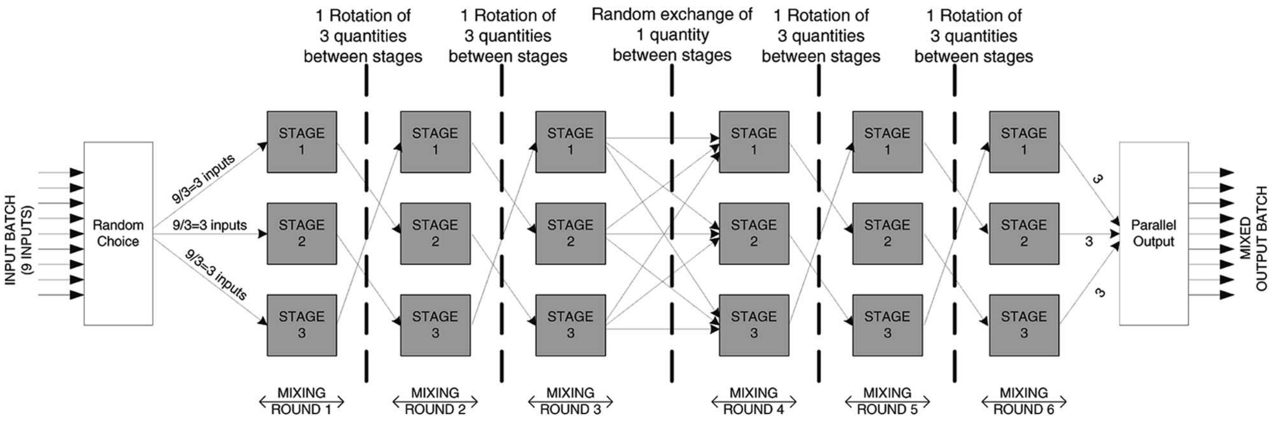


Fig. 16. ***Parallel mixnet containing three stages, with threshold $t=3$ and batch size $l=9$. Each stage receives $l/3=3$ inputs, from input batch. After initial mixing, $t-1=2$ rounds of rotation and mixing are performed by the stages. Next, stage 1 chooses randomly two quantities from its output batch and sends one each to the other two stages. Process is repeated in parallel by stages 2 and 3. After mixing, another $t-1=2$ rounds of rotation and mixing are performed. Totally, $2t=6$ rounds of parallel mixing are performed by three stages.***

The parallel mixnet in [64] is inherently robust against anonymity attacks. The rotation step provides robustness against tracing of inputs. For example, in Fig. 16, even if stage 1 was compromised and enabled an adversary to trace an input through it, the rotation with stage 2 would require the adversary to also compromise stage 2 to complete the tracing of the input. Now, if stage 2 as well as stage 3 are also compromised, then the mixnet is compromised, hence, indicating that robustness of the mixnet is limited to $t-1=2$.

Next, when the input batch of size l is divided into l/n, the anonymity of an input is decreased. But the random exchange step ensures that the anonymity is increased back to the batch size l. As an example, in Fig. 16, a passive adversary can observe a target input to be in the input batch of stage 1 and guess the input to be one of the three quantities passed to stages 2 and 3 during the rotation step. However, stage 3 then performs random exchange of two of its outputs with stage 1 and stage 2, while retaining one of the three outputs. Hence, after random exchange the adversary will have to guess the target input to be in one of the three stages, i.e., among the entire batch of nine inputs. Note that the random exchange step can be considered as an interstage random permutation of the mixed inputs.

The integrity and verifiability of the parallel mixnet is dependent on the verifiability mechanism used by the mixnet protocol. Note that the parallel mixnet requires a total of $2(t-1)+2=2t$, rounds of mixing. Then, for universal verifiability, the generation of proofs such as [59], [60], with complexity linear in the number of inputs, is required in each round. Since the mixing is performed on l/n inputs by each stage, the proofs take up to $2tl/n$ exponentiations for $2t$ rounds of mixing at each stage. To achieve robustness of $n-1$ as in a basic reencryption cascade mixnet, threshold $t=n$; hence, this would yield the complexity as $O(l)$, with a constant factor more than the reencryption cascade mixnet protocol. Therefore, by parallelizing the cascade mixnet, the computational efficiency remains at $O(nl)$ for the mixnet, but the latency in mixing is reduced, since all stages are processing in parallel. However, observe that the reaction to a compromised stage in this mixnet, necessitates a restart of the mixnet protocol. Hence, detection of compromised stages can degrade the latency.

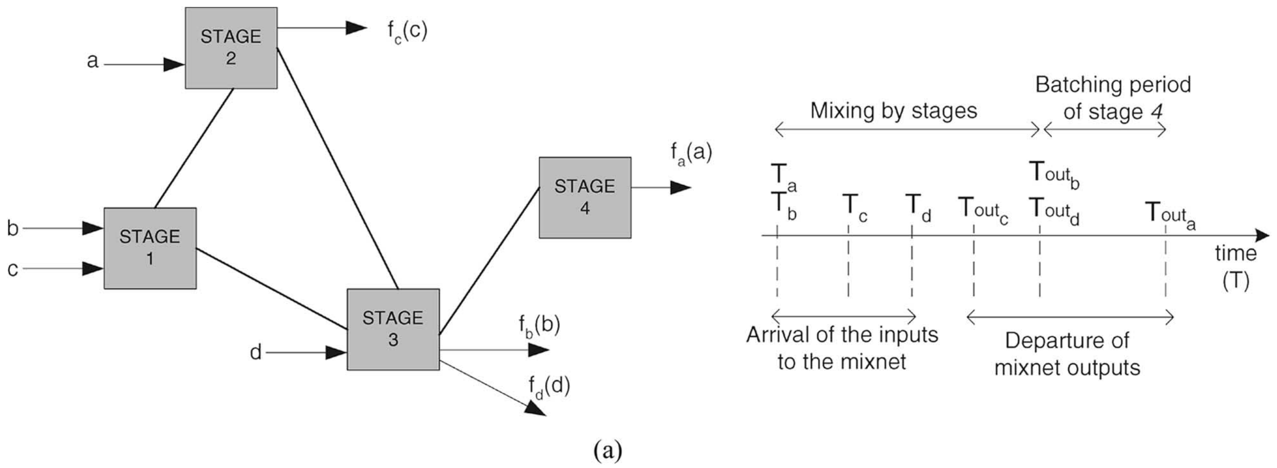
VII. FREE-ROUTING TOPOLOGY FOR ANONYMITY

As seen from the previous sections, a cascade mixnet requires the stages to interact and, hence, be dependent on each other for operation of the mixnet. Moreover, with the use of the public bulletin board and other robustness mechanisms, a centralized approach to mixnet implementation is desirable.

Network applications such as anonymous e-mail and anonymous web browsing need a distributed implementation for their operation over public networks. In addition, they may require low latency communications. A single cascade mixnet is unsuitable for such application requirements, hence, leading to the design of free-routing topology mixnets.

A. Asynchronous Batching in Free-Route Mixnets

Fig. 17 illustrates a free-route mixnet topology that consists of interconnected, but not necessarily dependent, stages. Note that a stage in a free-route mixnet can itself be a cascade mixnet. The free-route mixnet protocol is similar to the cascade mixnet in the mixing operation on the onions at the stages. However, the following important



(a)

Fig. 17. *Asynchronous batching in free-route mixnet with batch size threshold 2. Transformed input i at output of mixnet is denoted as $f_i(i)$, $i=a,b,c,d$.*

differences exist in the operating conditions of a free-route mixnet.

1) Any stage may receive inputs (onions) from the senders at different times, as illustrated in Fig. 17.
2) Additionally, the inputs to a stage may come from more than one connected stage. For example, stage 3 in the free-route mixnet of Fig. 17 receives inputs from stages 1 and 2.
3) Any stage can forward an output directly to an addressed receiver, which is again seen in the mixnet of Fig. 17, where stages 2–4 forward some outputs directly to a receiver.
4) Considering a fixed batch size l, each stage may wait for at least l inputs to arrive or may simply wait for a certain fixed amount of time before forwarding the inputs received to the next stage. In order to address latency constraints, there may be a batch threshold based on *size* (number of inputs), or time (batching period), or a combination of both, as described in [23]. For example, in Fig. 17, all stages have a batch size threshold of two inputs. Stage 4 receives only one input, and since it does not receive any more inputs, after a certain batching time period it forwards the transformed input to the addressed receiver. Note that in a cascade mixnet, a batch size threshold can also be applied, but only to the first stage of the mixnet. Therefore, the delay from waiting for a threshold number of inputs is incurred only once in cascade mixnets.

The previous four differences lead to asynchronous batching in a free-route mixnet. As shown in Fig. 6, recall that in cascade mixnets synchronous batching was used, where the mixnet input batches exited the mixnet in the same temporal order. In the case of asynchronous batching, such a temporal ordering may not exist, and inputs entering the mixnet at the same time may leave the mixnet at different times.

Fig. 17 illustrates asynchronous batching, where inputs arrive at different times. Though input c arrives after inputs a, b, it leaves the mixnet before them. This is because stage 1 waits for input c to arrive before forwarding in parallel the transformed b to stage 3 and transformed c to stage 2; while, stage 2, having received input a, will be waiting for another input. Upon receiving the transformed c, it is able to mix and forward in parallel the transformed input a to stage 3 and $f_c(c)$ to its destination. Hence, input c exits the mixnet before any of the other three inputs. Next, stage 3, which has already received the transformed b, receives the transformed a almost simultaneously with input d. Therefore, all three are forwarded in parallel in a batch, with the transformed a going to stage 4, and $f_b(b)$, $f_d(d)$ to their respective destinations. Exit times of all inputs are indicated. As mentioned earlier, $f_a(a)$ will only be sent after stage 4 waits for a batch period expecting more inputs.

B. Free-Route Mixnet Failure and Robustness Measures

Unlike the cascade mixnet topology, a number of anonymous paths with different number of stages are available in free-route mixnets. Because of the availability of a number of paths, a sender can choose an anonymous path, and the choice is reflected in the stage addresses included in the sender onion. A sender path can also contain loops, i.e., a stage may be traversed more than once in a path. For example, in the free-route mixnet in Fig. 17, a possible anonymous path can be sender-stage 1-stage 2-stage 3-stage 1-receiver. The multiple paths in the free-route mixnet inherently provide robustness against faulty stages (higher fault-tolerance). For example, in Fig. 17, if stage 2 fails, then to reach stage 4, the sender of input a can use the path stage 1-stage 3. However, the sender is required to reinitiate communication, including constructing and transmitting a new onion. This can be a disadvantage in certain mixnet applications. Techniques to reduce retransmissions include using diverse paths between sender and receiver, preassigning backup stages to support faulty stages, or enabling a stage to skip succeeding faulty stages in a path [85]. A concise presentation of the various application specific free-route mixnet designs that have been proposed is given as follows.

C. Remailer Applications

Free-route mixnets providing two-way communication have been proposed for remailer applications [6]–[8]. The remailers provide anonymous e-mail service. The security of the basic remailer using mixing, called Type-I remailer [5], was improved in the next generation of remailers (Type-II), such as the *Mixmaster* [6], [65] and *Babel* [7]. Recently, *Mixminion*, which belongs to the third generation of remailers (Type-III), was developed in [8], [66], addressing certain attacks on Type-II remailers. The basic mixnet protocol in all the remailers is identical to the decryption/hybrid mixnets given in Section III, with the RPI included by the sender, if a reply is expected from the receiver.

D. Low-Latency Applications

Free-route mixnets designed for remailers can incur delays that are not acceptable for real-time applications such as anonymous web browsing. This is because the remailers operate using a store-and-forward mechanism, which is not suited for the HTTP traffic involved in web browsing. In these applications, two-way communication is often bursty in nature and also delay-sensitive. Free-route mixnets providing two-way, real-time communication have been proposed in [11], [13], [67], and [68]. These mixnet protocols generally contain the following distinct phases.

1) *Anonymous path establishment*: Each sender generates and transmits an onion to set up a sequence of stages. As illustrated in Section III, the onion

contains the addresses as well as other necessary control information for each stage. This control information includes a forward symmetric key, and a reverse symmetric key, for each stage along the anonymous path.

2) *Anonymous communication*: Once the path is established, data communication takes place. The sender encrypts data using the forward symmetric keys that were included in the path establishment onion. Each stage decrypts a layer of encryption using the corresponding forward symmetric key obtained during path establishment. The data going from the receiver to the sender is encrypted using a reverse symmetric key at each stage.

There are additional mechanisms in the previous protocol phases that are used to achieve robustness against passive and active attacks (see Section VII-F). Such free-route mixnets have been proposed and practically implemented in *onion routing* [11], [14], [68] for two-way, real-time anonymous traffic. The mixnet in [11] has been improved, in terms of implementation, in the second generation of onion routing [68]. Unlike [11], the anonymous path establishment in [68] is initiated but not finalized by the sender.

E. Inapplicability of Reencryption Mixing, Authentication, and Verification in Free-Route Mixnets

In the previous sections, we saw that reencryption cascade mixnets outperform their decryption counterparts in terms of security and performance. However, we also mentioned in Section III-D, that reencryption mixnet design does not support free-routing topologies, since it cannot include the address of the stages. Hence, the advantages and properties of reencryption mixnets cannot be extended to free-routing topologies. The approach that is considered for the design of free-route mixnets is to have independent, interconnected, decryption/hybrid mixing stages, deployed on a public network.

In contrast to the cascade mixnet, a free-route mixnet cannot employ robustness measures such as sender authentication and verifiability. These mechanisms contradict with the purpose of the free-route mixnets. Since all stages may receive sender inputs, authentication would require a centralized approach to enable stages to verify the validity of any sender input. This restricts a distributed implementation and may also incur additional delays. Moreover, sender authentication by stages can be misused by a compromised stage to add inputs that it can claim to be from valid senders [67]. Also, as noted earlier in Section IV-E, an authenticated public bulletin board has to be avoided in decryption/hybrid mixnets for robustness against replay of inputs. Therefore, authentication is substituted with other solutions as discussed in the next section. Note that authentication can still be used between stages in the free-route mixnet.

As seen previously, most of the verifiability mechanisms require stages to participate jointly in the mixnet protocol. This leads to latency concerns and also to coordination requirements between stages that are undesirable properties in free-route mixnets. Thus, except for sender verification and checksums, other verifiability mechanisms are avoided. Consequently, without sender authentication and verifiability being available, a free-route mixnet is less secure compared to a cascade mixnet, as observed in [22]. We now consider the different types of attacks and the heuristic robustness solutions used in free-route mixnets.

F. Passive Attacks on Anonymity in the Free-Routing Topology

Many of the proposed free-route mixnets are dependent on the assumption that constant traffic is distributed uniformly in all paths of the mixnet, i.e., there are two or more sender communications in all the paths of the free-route mixnet. The assumption enables analysis of performance of the mixnet in terms of latency and analysis of the security of the mixnet under active attacks. But under variable traffic conditions at the stages, a sender's path may be traceable by a passive attack, since the stages in the path may not be used by other senders. Traffic analysis attacks, including attacks based on the variable batch size at the stages in a free-route mixnet, are described as follows. Detailed investigations into these traffic analysis attacks can be found in [20], and [69].

1) Misuse of the Decreasing Size of Onions: Consider the case, where the traffic in the mixnet is not uniformly distributed over the stages. By the inherent property of the layer encrypted onion, each sender's onion will reduce in size as it traverses the stages in the mixnet. This is illustrated in Fig. 18(a). As seen, stage 2 receives a single onion which it transfers and forwards to stage 1 after a batching period. Stage 1 receives the single onion of a particular size from stage 2, and also two sender onions of a different size. Then, by using this onion size information, it may be possible for an adversary to trace a specific target onion from input of stage 2 to the output of stage 4. As shown in Fig. 18(b), such an analysis attack can be partially avoided using padding [6], [11] at each stage to ensure a uniform size for all the onions exiting the stage. The padding can be a redundant string that is known only to the stages of the mixnet. Note that the overhead associated with padding results in decreased throughput of the mixnet.

2) Misuse of the Variable Batch Size: It can be noted that in Fig. 18(a) an adversary would still be able to partially trace a target input from stage 2 to stage 1. Such attacks can be dealt with by using dummy traffic [6] generated at the stages. The dummy onions must appear similar to the actual sender onions. In Fig. 19(a), stage 2 generates a

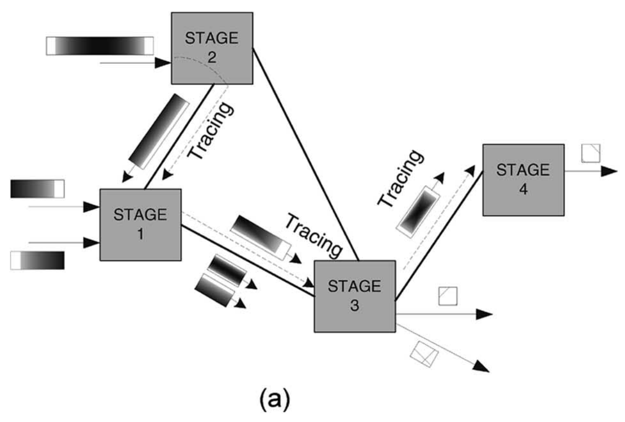


Fig. 18. *(a) Nonrobust free-route mixnet where a passive adversary can trace an input through mixnet. (b) Tracing is partially disabled with use of padding at stages. All onions in mixnet are padded to be of the same size.*

dummy onion that prevents tracing of the input from stage 2 to stage 1 by the adversary. The dummy onion is identified and discarded later by stage 3. A more robust mechanism can be obtained by using end-to-end dummy traffic, i.e., inactive senders keep sending dummy messages to receivers to maintain a fixed batch size at all the stages [9], [67]. Fig. 19(b) illustrates the benefit of end-to-end dummy traffic, where tracing through stage 4 is not possible. The disadvantage with end-to-end dummy traffic in a mixnet is the overhead generated for the senders/receivers having nothing to communicate, as well as overhead for the network. The tradeoff between the performance and the security properties with the use of dummy traffic still requires further investigation, as indicated in [23], [27], and [70].

3) Misuse of Deterministic Delay of a Path: The various anonymous paths in a free-route mixnet may have different fixed delays associated with them. Based on: 1) time of arrival of a target input; 2) the outputs exiting the mixnet at subsequent times; and 3) the known path delays, the adversary can conjecture the path of a target sender's input. It is also possible for a collusion of compromised stages in the mixnet to perform timing-based attacks [71].

To address these timing-based attacks, random delay of inputs can be used at the stages [6]. A stage essentially stores the inputs received in a pool [6], [72], and once a batch threshold (number of inputs and/or time) is reached, then a random selection of the inputs from the pool are mixed, along with (optionally) some dummy inputs, and sent in the output batch of the stage. The remaining inputs

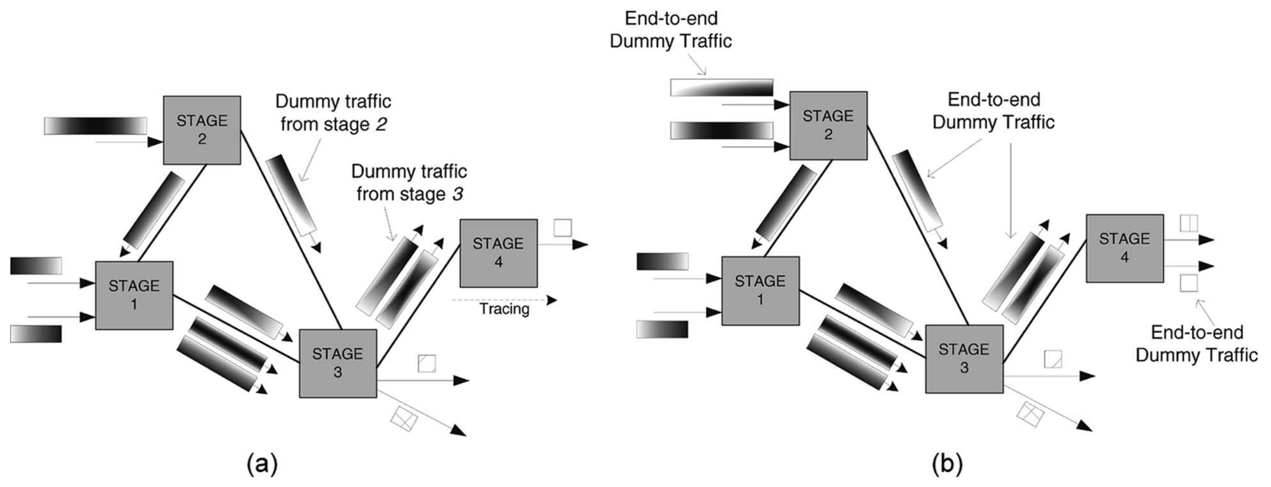


Fig. 19. *(a) Dummy onion generated at stage 2. Dummy onion is discarded by stage 3, which forwards transformed sender onions and dummy onion to stage 4 as output. Note that adversary is not able to trace sender onion from stage 2 to stage 1 but can still trace an onion at stage 4. (b) End-to-end dummy onion between a sender and a receiver. Stage 2 receives dummy onion from sender, and stage 4 will forward dummy to receiver. Hence, no tracing is possible.*

are kept in the pool for mixing at a later time. Therefore, the adversary may not be able to conjecture the delay incurred by a target input at each stage. However, the random delay mechanism may only slightly increase the difficulty of the adversary, since the average estimates of the delay at each stage, hence the paths containing them, can be computed by the adversary [23].

Another drawback of employing random delay at the stages is that it adds to latency in the communication process. An interesting variant has been proposed in [73], where the sender is able to choose the tradeoff between anonymity and latency. The technique employed is to make the sender include in the onion a time duration as well as an upper and lower time bound for *each* stage. For example, a stage j decrypts the sender onion and obtains a duration T_{d_j}, an upper time limit T_{u_j}, and a lower time limit T_{l_j}. If the current time is T, then stage j verifies if $T_{l_j} \leq T \leq T_{u_j}$, and if correct, then the forwarding onion is delayed by T_{d_j}. Hence, the sender decides the acceptable latency in its path as well as the possible anonymity gained from that latency. Observe that on the downside, the sender's onion may be dropped if any additional unintended (but harmless) delay occurs in the path.

It is important to note here that the use of random delay of inputs further contributes to asynchronous batching in free-route mixnets. An input entering a stage at time T_1 may exit the stage only after an input entering at time $T_2 \geq T_1$ has exited the stage.

Timing-based attacks can also be avoided to some extent, using end-to-end dummy traffic as in [9], since it would increase the uncertainty in the pool at each stage traversed by the targeted input. Another technique is to use dummy traffic only between stages in a segment of the anonymous path as in [74]. A related approach was proposed recently in [71], where random stages drop dummy onions traversing them, to address timing-based attacks by a collusion of compromised stages. The idea is, again, to create uncertainty for an adversary attempting to trace target onions.

G. Active Attacks on Anonymity in a Free-Routing Topology

In a free-route mixnet, since no authentication is used and since all stages can accept inputs, the adversary can manipulate the inputs entering *any* stage. This leads to a series of possible attacks and corresponding robustness mechanisms in the mixnet.

1) Manipulating Input Batch of Stage: In an active attack, the adversary can delay the target input and then wait for the remaining inputs to exit the stage j. The adversary can then send $l-1$ of *known* inputs along with the target input to stage j. It is also possible that the adversary may delay all but the target input to a stage j, so that it can then be traced through the stage, after the target input is forwarded when the batching time period concludes. These attacks are illustrated in Fig. 20.

To address such traffic manipulation attacks, time stamps can be used in the onions [7], [11]. Time stamps determine the freshness of an input. If the current time at the stage is a predetermined period beyond the time stamp in the onion, then the onion is dropped at the stage. However, time stamps require synchronization and also can lead to traffic analysis attacks based on time stamp granularity [8]. For instance, if the granularity is in the range of seconds, then it is likely that an input entering the

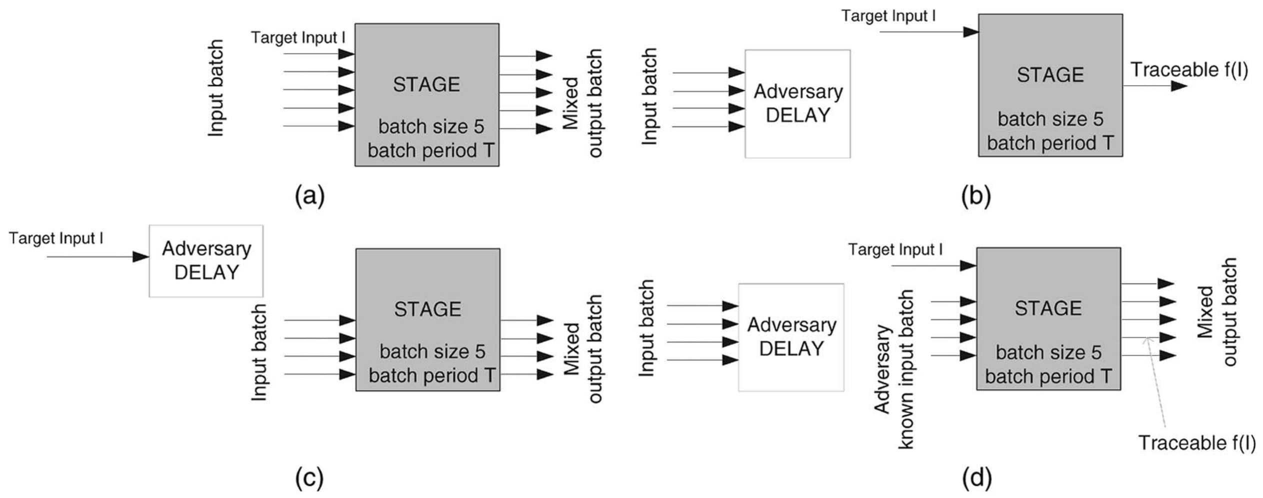


Fig. 20. ***(a) Passive attack requires adversary to guess input corresponding to one of the outputs in mixed batch. (b) Active attack where adversary delays all inputs except target input, which may then be flushed by stage after batching period T. (c) Active attack where attacker delays only target input. (d) After delaying target input, adversary floods the stage with four known inputs and delayed target input. Hence, adversary can trace target input even in mixed output batch.***

mixnet has a unique time stamp in the onion, which will make the input traceable through the stages on its path.

Another robustness mechanism uses link encryption between stages [6], [8], [13]. Such an approach makes it difficult for the adversary to recognize the known $l-1$ inputs exiting stage j. It can be observed that the use of link encryption between all stages of a decryption mixnet makes it equivalent to the reencryption mixnet, assuming that the sender cannot collude with any stage in the mixnet. A sender cannot trace its own input through the mixnet. However, a compromised stage $j+1$, which knows the pairwise key used for link encryption by stage j, can use the key at a later time to decrypt the inputs. To address this weakness, session keys are used for link encryption [8]. Session keys are used for only one input batch and then deleted. Hence, even if a stage $j+1$ is compromised, it cannot be used to obtain keys to decrypt old input batches received from other stages.

Also, the random delay mechanism that was used earlier for robustness against misuse of the path delays can be used against these traffic manipulation attacks. In order to flush the target input from the pool at the stage under attack, the adversary has to spend time and resources by continuing to send known inputs and delaying unknown inputs trying to enter the stage. The adversary must also be able to differentiate the stage transformed target input from the dummy inputs that might be added to the pool by the stage [107]. Self-addressed dummy inputs can also be used by stages to detect traffic manipulation attacks on the mixnet [75]. However, a compromised stage can still leverage random delay mechanism to deterministically delay a target input in the pool and enable its tracing in a subsequent batch. Such an attack can be detected by using commitment schemes for ensuring randomness in the batching at each stage [105].

2) Replay of Inputs: As seen in Section IV-E, in cascade mixnets an adversary may try to replay a target input in the same batch or a subsequent batch, in order to trace its path. To address such a replay attack in a free-route mixnet, each stage j in the path needs to store processed inputs and drop any detected replays. By storing time stamps and nonces used in the onions, detection of replay onions at the stages can take place. Alternatively, the stages can remember other quantities in the onion, such as the public key encryption of the symmetric key [7] or the hash of inputs [8]. Note that storing the hash of inputs prevents unwanted future access to information on the inputs received by a stage.

The period of time for storing the inputs may be determined by a public key rotation period [8]. This technique is similar to the use of session keys to provide forward secrecy [76]. Essentially, the public/private key of a stage is changed at the end of a rotation period, and the old public/private key is deleted. Hence, even when compromising a stage j, the adversary will not be able to decrypt old input batch replays beyond the fixed rotation period. Similarly, forward secrecy with session keys is achieved in [68]. The overhead associated with these techniques is mainly the key establishment and key distribution.

Replay attacks can also be addressed using interstage detours [7]. Each stage j selects a detour path to the next stages that are addressed in the onions of its input batch. The detour paths are randomly selected. As illustrated in

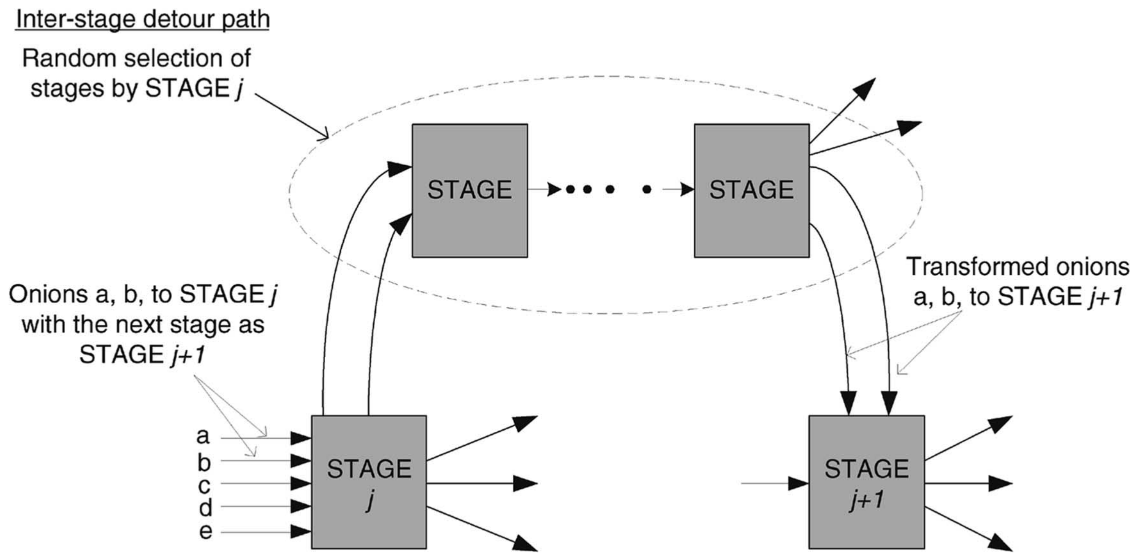

Inter-stage detour path
Random selection of stages by STAGE j
STAGE
STAGE
Onions a, b, to STAGE j with the next stage as STAGE j+1
Transformed onions a, b, to STAGE j+1
a
b
c
d
e
STAGE j
STAGE j+1

Fig. 21. ***Stage j receives a batch of onions. Two onions a, b have a stage $j+1$ as next stage in their path. Two onions are encrypted by stage j with public keys of randomly selected stages in detour path, with stage $j+1$ as receiver. If an onion, say a, is replayed by an adversary, then random selection of stages may be different, hence leading to different encryption by stage j.***

Table 4 Some of Robustness Techniques in Free-Route Mixnets: ↑-Increase; ↓-Decrease

Technique	Attack Addressed	Main Performance Property Affected
Traffic padding	Misuse of onion size	Throughput ↓, low latency
Dummy traffic	Misuse of variable batch size, misuse of path delay, manipulation of input batch	Throughput ↓, low latency
Random delay of inputs	Misuse of path delay, manipulation of input batch	Latency ↑, high throughput, throughput ↓(if dummies are used)
Time stamp in inputs	Manipulation of input batch, replay of inputs	Implementation
Link encryption between stages	Manipulation of input batch, replay on inputs	Implementation (if session key is used)
Session key	Manipulation of input batch, compromise of keys	Implementation
Public key rotation	Manipulation of input batch, compromise of keys	Implementation
Storing inputs at stages	Replay of inputs	–
Storing hash of inputs	Replay of inputs, compromise of input information	–
Inter-stage detours	Replay of inputs in forward and return paths	Latency ↑
Loose routing	Replay of inputs in forward and return paths	Latency ↑
Multiple RPI	Misuse of return path information by replay of RPI	Implementation

Fig. 21, after decrypting a layer from the onions a, b, stage j obtains the next stage $j+1$ address in both of them. Stage j then sends the two onions on a path containing a random selection of stages, with stage $j+1$ as the receiver. Before sending the onions on the detour path, stage j encrypts them with the public keys of the random stages in the detour path. Hence, if an onion is replayed by an adversary in any subsequent batch, the onion may appear different at the output of stage j, since a different detour path may be chosen by stage j. A similar approach called loose routing is proposed in [11], where additionally the length of the detour path is constrained by a parameter included in the onion. As pointed out in [11], this technique can also be used if the topology of the free-route mixnet is not known completely by the sender, thereby allowing each stage to independently determine the path to the next stage. However, both of the mechanisms in [7] and [11] add to the latency incurred by the sender onions, since the onions have to traverse the additional random stages selected at each stage in the original anonymous path of the sender. We observe that these mechanisms can be considered as an extension of link encryption with a session key between stage j and stage $j+1$.

3) *Misuse of Return Path Information:* While replay of inputs can possibly be detected or avoided in the forward path (sender to receiver), an active adversary can still utilize the RPI onion to trace the sender. The sender's RPI is included by a receiver in its reply. It is possible that a receiver may send the same reply to the sender messages. Hence, a stage j cannot distinguish a valid repeated reply from the receiver, from a replay by the adversary. The adversary can send l copies of the reply originally sent by the receiver, to stage j. Then, based on the path (included in the RPI of the reply) taken by l onions from stage j through the mixnet, the adversary may be able to successfully trace the sender receiving the l copies. Such an attack is possible in [7], since the same RPI can be used multiple times by the receiver, allowing the adversary to also use it for tracing. The attack is addressed in [8] by allowing only a single use of the RPI. With such an approach, however, the sender must be able to provide multiple RPI with different return paths to a communicating receiver.

Further, we note that the use of RPI enables tagging attacks which can be employed to trace the sender. Suitable defense mechanisms for such attacks are in [8]. Table 4 summarizes some of the main robustness techniques employed in free-route mixnets against passive and active attacks and their associated performance tradeoffs.

H. Attacks on Integrity in Free-Routing Topology

As seen in Section VII-E, verifiability mechanisms are not generally applicable to free-route mixnets. Hence, it is possible for an adversary to break the integrity of the inputs. Additionally, the adversary can launch an anonymity attack based on active corruption of inputs. This is the tagging attack [8], where a compromised stage j can corrupt an input and possibly identify it at a subsequent compromised stage $j+2$. Such an attack can be addressed partially by including checksums with each layer of the onion as in [24] (Section V-B-2) to verify the integrity of the received onion at each stage [6], [8].

However, the addition and deletion of inputs at a compromised stage can still occur in free-route mixnets. The assumption is that the sender will eventually be notified through mechanisms using receipts [1], [25] that its message has not been received. The sender will then reinitiate the anonymous communication. Consequently, it is useful to design a mechanism that can enable a sender to identify reliable paths in a free-route mixnet for anonymous communication. Such a mechanism based on *reputation of mixnets* is presented in the following section.

VIII. REPUTABILITY MECHANISMS IN MIXNETS

A reputability mechanism in mixnets provides a means for an uncompromised stage or an uncompromised cascade mixnet to improve its reputation. If the reputation is determined by the reliability, then a sender can construct a reliable anonymous path, by choosing stages or a cascade mixnet with high reputation. However, a reputability mechanism presents the danger of being misused by an adversary. A set of compromised stages or a compromised cascade mixnet can increase its reputation by performing correct mixing of inputs, while simultaneously attacking the uncompromised stages and cascade mixnets, so as to decrease their reputation. Senders may then possibly end up choosing the compromised stages with relatively high reputation in their paths. These anonymous communications can then be compromised by the adversary without detection. It is important that the design of a reputation mechanism address such a misuse by the adversary. We first describe a reputation mechanism proposed in [25] that can be used in free-route mixnets.

A. Reputation in Free-Route Mixnets

The mechanism proposed in [25], involves a set of raters that essentially includes the senders and the stages themselves. Also, a set of trusted/semi-trusted third parties called scorers are involved. The trusted scorers determine each stage's score (reputation) based on the ratings obtained by the stage. The stage's behavior during the mixing protocol determines its rating. Each stage in the sender's anonymous path is required to provide a signed receipt to the sender. Failure to follow protocol leads to the intervention by the scorers who then confirm the stage to be compromised/faulty and give the stage a negative rating. A successful transmission results in a positive rating for the stages involved. If the scorers are only semitrusted entities, then senders will be provided with a mechanism to maintain the scores of the stages on their own.

In order to address misuse of the reputability mechanism, every stage in the mixnet is required to obtain positive ratings from many senders to increase its score/ reputation effectively. Hence, an adversary will find it difficult to send its own messages through the compromised stages and thus increase their reputation while attacking other uncompromised stages.

B. Reputation in Cascade Mixnets

In a free-route mixnet it is possible that a stage may itself be a cascade mixnet. In a cascade mixnet, it is sufficient for the adversary to compromise a single stage to decrease the reputation of the entire mixnet. To address this type of attack, in [77] the proposed reputation mechanism reconstructs the cascade mixnets periodically, using a random choice of stages with high reputation. The remaining available stages are combined to form other cascade mixnets. There is no trusted third party, only the stages jointly participating in the reconstruction of the cascades. Hence, in such a reputation mechanism the adversary has to control several stages before it can create a high reputation cascade mixnet. However, the overhead generated by the periodic distributed reconstruction of the cascade mixnets requires further investigation.

While the previous reputability mechanisms can be integrated into mixnet protocols, both approaches assume that the adversary can decrease reputation of stages and cascade mixnets. In contrast, a mechanism that can be integrated into the cascade mixnet protocol to enable the mixnet to prove its reputation and also to detect any compromised stage is proposed in [78]. As long as the cascade mixnet does not contain threshold t or more compromised stages, the mechanism provides robustness against any misuse of reputation.

The main idea of the reputability mechanisms in [78] is that the stages can jointly show that a message comes from a valid input batch, without breaking the anonymity of its sender. This allows the cascade mixnet to assert that it is has not added or corrupted any message in its output batch. Hence, upon detection of a corrupted message at the receiver, the mixnet can indirectly prove that the message was corrupted before arriving or after leaving the mixnet. On the other hand, if a stage is compromised, then the mechanism does allow for its detection. The following are the reputability mechanisms that are proposed in [78], using threshold signature schemes [79], [80].

1) *Mechanism for a Decryption/Hybrid Cascade Mixnet*: The sender i interacts with the n stages to first obtain a threshold blind signature S on the message m (described in Appendix D). The sender then broadcasts the encryption of $(m||S||b)$ to a public bulletin board, where b indicates the current batch number, which is publicly known. Note that the blind signature is used to ensure that message m is not revealed to the stages, hence protecting sender privacy. From the mixnet output batch, anyone can verify if the signature S corresponding to each message m is valid, and if not, tracing is initiated. Tracing detects either corrupt inputs to the mixnet or a compromised stage. In the case of a corrupted input, anonymity of the communication from the sender to the receiver is breached. If a compromised stage is detected, then the mixnet reputation decreases.
2) *Mechanism for ElGamal Reencryption Cascade Mixnet*: In this mechanism, the ElGamal encrypted sender inputs containing $(m||b)$, broadcasted to the bulletin board, are jointly signed by

the n stages using a threshold signature scheme [78] as $(m||b)^d$, where d is the private key shared by the n stages. The stages then perform the reencryption of the message, as well as that of the signed form of the message. After the decryption of the mixed output batch is performed, anyone can verify based on the signature if an output is derived from one of the inputs in batch b. As before, any corrupted output leads to a tracing which in turn leads to either a corrupt input or a loss of mixnet reputation.

It can be observed that in both the mixnet protocols, if the threshold t or more stages are compromised, then the mixnet is compromised. Note that as in sender verifiable cascade mixnet, the sender here must verify if its message is present in the mixed output batch. Also, if the sender does not verify its input posted on the bulletin board, then its anonymity can be broken, since corruption of the input would lead to eventual trace back in the mixnet protocol. Hence, use of reputation in a mixnet may not provide robustness against attacks. In fact, the robustness of the mixnets in [78] is comparable to the sender verifiable cascade mixnets seen in Section V-A, where only detection of corrupted inputs is provided by a checksum.

IX. COMPARISON OF ANONYMITY SOLUTIONS AND MIXNETS

A. Solutions for Anonymity

Before providing a comparison of mixnets, we consider other types of solutions for anonymity that have been proposed, such as [81]–[84]. As shown in Fig. 22(a) and (b), in these channels the sender may have two or more connected peers, and as long as all of its connections are not tapped and all the peers are uncompromised, the sender cannot be traced from a communication [85]. Thus, sender anonymity [85], as well as untraceability, can be achieved. In contrast, the sender can be identified and traced to the input of the mixnet, as seen in Fig. 22(c). Untraceability is all that is achieved by the stages of the mixnet.

The anonymous channel proposed in [81] and [85] [Fig. 22(a)] is useful particularly for broadcast communications, and can provide unconditional anonymity to sender as well as receiver [85], [104]. The anonymous channels in [83] and [84] [Fig. 22(b)] are useful for low-latency communications. However, these anonymity solutions are suitable for peer-to-peer networks, requiring network nodes (senders and receivers) to participate, even if they have nothing to communicate. A single node can disrupt any anonymous communication traversing it. Moreover, assuming a powerful adversary that is capable of tapping all communication channels in the public network, the mixnet provides better anonymity compared to the solutions in [83] and [84]. In their current form, when compared to the mixnets, these peer-to-peer anonymous channels are not necessarily scalable, robust, or efficient for secure applications [86].

Fig. 24 summarizes the different approaches to anonymity and also presents the classification of mixnets based on verifiability. It should be noted that mixnets can indeed provide sender/receiver anonymity when combined with digital pseudonyms for the senders [1], [45], [85]. A comparison of the main classes of mixnets is provided as a summary, together with open problems in each type of mixnet.

B. Topology of Mixnet

In both the mixnet topologies, except for the first and last stages, an intermediate stage j would not know its

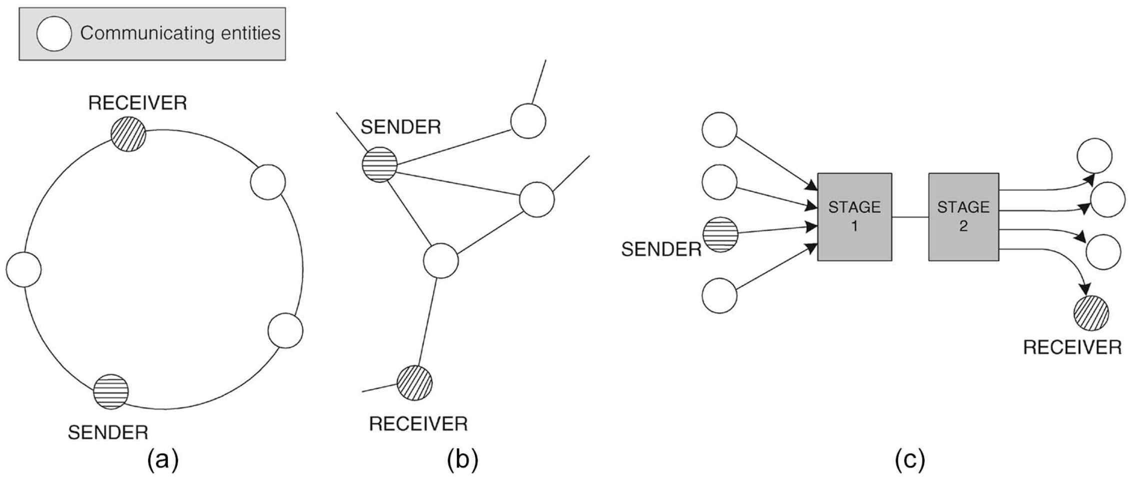


Fig. 22. *Solutions for anonymity. (a) Peer-to-peer-based solution capable of providing sender and receiver anonymity. (b) Peer-to-peer-based solution capable of providing sender anonymity. (c) Mixnet solution providing only untraceability.*

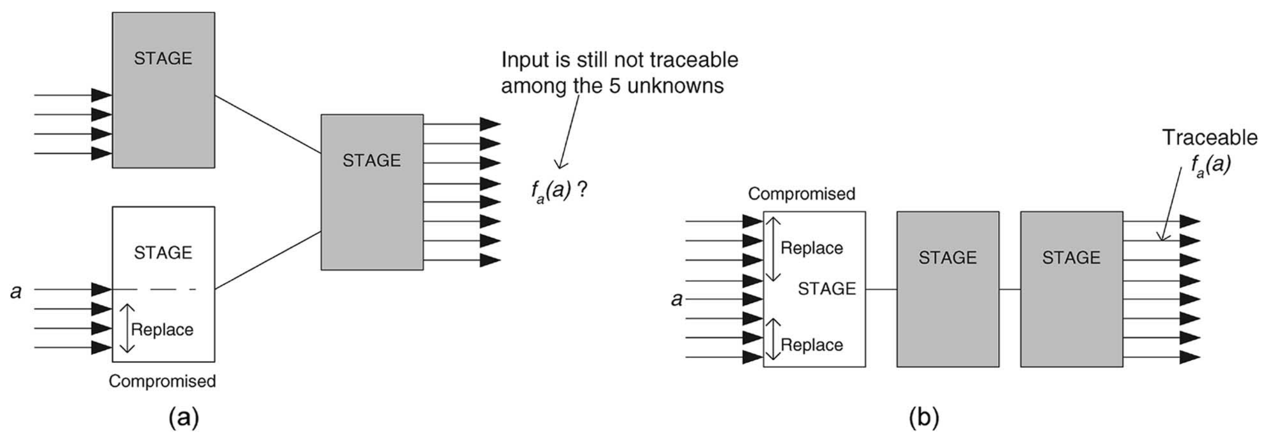


Fig. 23. ***Comparison of anonymity of two decryption/hybrid mixnet topologies for low-latency application, assuming that a single compromised stage replaces input batch to trace an input a. (a) In free-routing topology, anonymity of input is still protected by input batch from the other uncompromised stage. (b) In cascade topology, tracing of input*** *a* ***is possible. However, verifiability can address this attack at the expense of latency.***

location in the anonymous path. Each intermediate stage is only aware of its preceding and succeeding stages in the path. Note that under no active attack, the anonymity of a path in a mixnet is preserved as long as at least one stage in the path is uncompromised. In [22], it is observed that while the cascade topology provides overall better security properties compared to the free-routing topology in mixnets, under certain conditions, the free-routing topology can provide more robust anonymity.

1) Assume that a mixnet is to be designed for a low-latency application, so verifiability cannot be used in the cascade mixnet. As observed in [22], Fig. 23 illustrates a scenario where the anonymity of a target input *a* can be protected in the free-routing topology, while in the cascade topology it can be breached.

2) Also, recently, it was shown in [38] that a free-route mixnet with synchronous batching can provide better anonymity, fault-tolerance, and scalability properties, compared to cascade mixnets. The analysis in [38] is, however, limited to a fully connected free-routing topology, and it assumes that the input batch of the mixnet is large enough to provide uniform distribution of inputs across all the stages and that the anonymous paths in the mixnet are of equal length. Hence, input batches are received and processed synchronously by all the stages and will exit the mixnet in the same temporal order. An interesting observation is that the parallel mixnet proposed in [64] (Section VI) maps to a combination of two free-routing topologies from [38].

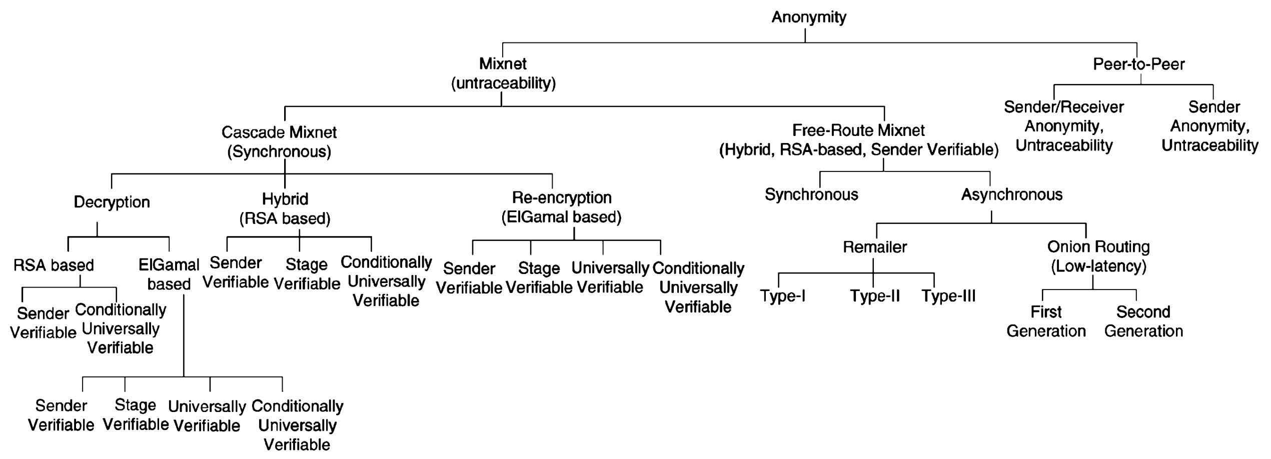

Anonymity
Mixnet (untraceability)
Peer-to-Peer
Sender/Receiver Anonymity, Untraceability
Sender Anonymity, Untraceability
Cascade Mixnet (Synchronous)
Free-Route Mixnet (Hybrid, RSA-based, Sender Verifiable)
Decryption
Hybrid (RSA based)
Re-encryption (ElGamal based)
Synchronous
Asynchronous
Remailer
Onion Routing (Low-latency)
RSA based
ElGamal based
Sender Verifiable
Stage Verifiable
Conditionally Universally Verifiable
Sender Verifiable
Stage Verifiable
Universally Verifiable
Conditionally Universally Verifiable
Type-I
Type-II
Type-III
First Generation
Second Generation
Sender Verifiable
Conditionally Universally Verifiable
Sender Verifiable
Stage Verifiable
Universally Verifiable
Conditionally Universally Verifiable

Fig. 24. ***Overall classification of anonymity and mixnets.***

Table 5 Comparison of Cascade Mixnets: D-Decryption; H-Hybrid; R-Reencryption; E-ElGamal; n-Number of Stages in Mixnet; t-Threshold of (t, n) secret sharing scheme used to share keys; $\surd$-Satisfied; C-Conditionally Satisfied; $\times$-Not Satisfied; Med-Medium; SV-Stage Verifiable; UV-Universally Verifiable; CUV-Conditionally Universally Verifiable

Mixnet	Type	Cryptosystem	Anonymity	Integrity	Verifiability	Robustness	Fault-tolerance	Latency	Efficiency	Scalability
[1]	D, H	RSA	C	$\times$	Sender	$\times$	$\times$	Med	Low	C
[9]	H	RSA	C	$\times$	Sender	$\times$	$\times$	Low	Low	C
[33]	D, R	E	C	$\times$	Sender	$\times$	$\times$	Med	Med	C
[40]	D, R	E	C	$\surd$	UV	$\times$	$\times$	High	Low	C
[39]	R	E	$t-1$	$\surd$	UV	$t-1$	$n-t$	High	Low	C
[52]	R	E	$t-1$	$\surd$	UV	$t-1$	$n-t$	High	Low	C
[44]	R	E	$t-1$	C	SV	$t-1$	$n-t$	C	C	C
[46]	R	E	$t-1$	C	SV	$t-1$	$n-t$	C	C	C
[53]	D, R	E	$t-1$	$\surd$	UV	$t-1$	$n-t$	High	Low	C
[54]	R	E	$t-1$	$\surd$	UV	$t-1$	$n-t$	High	Low	C
[47]	R	E	$\sqrt{n}-1$	C	SV	$\sqrt{n}-1$	$\sqrt{n}-1$	Med	Med	$\times$
[49]	H	E	$t-1$	C	SV	$t-1$	$n-t$	Med	Med	$\times$
[24]	H	RSA	C	C	SV	$t-1$	$n-t$	C	C	C
[59][a]	R	E	$t-1$	$\surd$	UV	$t-1$	$n-t$	Med	Med	C
[60][a]	R	E	$t-1$	$\surd$	UV	$t-1$	$n-t$	Med	Med	C
[57][b]	D	E	$t-1$	$\surd$	UV	C	C	High	C	C
[2][a]	D, H, R	E, RSA	C	C	CUV	$t-1$	$n-t$	C	C	C
[62]	R	E	C	C	CUV	$t-1$	$n-t$	C	C	C
[63]	R	E	C	C	CUV	$t-1$	$n-t$	C	C	C
[19]	R	E	$n-1$	$\surd$	UV	$n-1$	$n-1$	High	Low	$\times$
[64]	R	E	$t-1$	$\surd$	UV	$t-1$	$n-t$	Low	Med	$\surd$

[a] The proposed mechanisms can be employed in a robust cascade mixnet

[b] Need a trusted entity for robustness of the cascade mixnet

Further, we note that under a model weaker than the powerful adversary model, the free-routing topology requires more adversarial effort for controlling input and output points of the mixnet. An interesting direction of research, as suggested in [87], is to develop a mixnet design that can provide the advantages of the free-routing as well as the cascade topologies.

C. Cryptographic Transformation in the Mixnet

Table 5 provides an overall comparison of the various cascade mixnets that were presented in the previous sections. As can be observed, the ElGamal-based reencryption and decryption mixnets efficiently satisfy many of the security properties. Further, computationally efficient proofs have been developed in [59] and [60], for universally verifiable reencryption mixnets, and in [53] and [57] for universally verifiable decryption mixnets. However, due to the inability to accommodate for addresses of the stages, the ElGamal-based mixnets cannot extend their advantages to support free-routing topologies that require senders to include the addresses. Thus, extending the ElGamal-based mixnet design to free-routing topologies remains an open problem. A possible solution to this problem may be obtained by adapting the reencryption mixnet designs proposed in [19]. However, recent results have indicated that this solution can be challenging [106].

The RSA-based decryption/hybrid mixnets are readily applicable to free-routing topologies, since they can accommodate stage addresses. However, they need the sender to perform multiple encryptions. An open problem in the design of decryption/hybrid cascade mixnets is that they are not universally verifiable. A threshold number of compromised stages in RSA-based decryption/hybrid cascade mixnet can break the mixnet integrity without detection. Hence, in their current form, they have limited use in applications such as secure electronic voting.

X. CONCLUSION

In this tutorial paper, we have described mix networks under a common framework and have presented the tradeoffs that arise in its design. While the number of applications of mixnets has grown since its construction, we focus on presenting its use only in some secure applications. Electronic voting, which is a popular social application of cryptography, benefits from the anonymity as well as other security properties exhibited by the mixnets. Anonymity itself can be provided by other cryptographic constructions and network architectures. But, for anonymous communication in terms of untraceability from the receiver to the sender, the mixnet is still the most attractive solution when both security properties and scalability need to be satisfied. The peer-to-peer-based anonymity solutions can offer unconditional sender and receiver anonymity and are attractive in terms of protection from the consequences of the use/misuse of anonymity. However, in their current form, these solutions can fail to meet both security as well as performance requirements.

One of the most challenging future works in mixnets is its efficient and secure application to free-routing topologies. Currently, the decryption and hybrid type of mixnets are solutions one can provide for a distributed implementation, requiring multiple encryptions at the user end. A robust solution, based on the reencryption type of mixnet, can be more computationally efficient as well as secure. Although a step towards achieving such a solution has been taken in [19], the design still needs multiple encryptions by the sender and has reported weaknesses in its applicability.

An inherent limitation of mixnet design is that the maximum anonymity is limited only to a collusion of all stages of the mixnet. Design of a mechanism in mixnets that enables detection/prevention of such a collusion is an open problem. In [88], an approach that performs fragile mixing has been proposed, where stages are discouraged from breaking anonymity of specific inputs in the input batch. However, this approach assumes that the stages have a vested interest in preserving anonymity of other correspondences in the batch.

Another area of research in mixnets is related to its practical implementation. Recent developments have shown that the anonymous communication providers can be discouraged to provide service, due to the possible consequences related to misuse of anonymity. The reputation mechanism in [78] provides an interesting direction by enabling a reencryption cascade mixnet to prove its innocence, if any illegal messages are detected from it. A similar mechanism for a free-route mixnet can be attractive for remailers and anonymous web browsing providers.

In this paper, we have also provided a classification of mixnets, based on verifiability mechanisms, that plays a pivotal role in secure applications. The classification helps in understanding the effective role of anonymity and mixnets in widespread social applications, such as secure electronic voting schemes and the related performance and security tradeoffs. ■

APPENDIX

A. Public Key Cryptosystems

As in many applications of cryptography, public key cryptosystems [29] play a central role in the construction of mixnets. They are used by the senders and receivers to encrypt the messages and by the stages to transform inputs during the mixing operation. The public key cryptosystems that have been employed in mixnets are: 1) RSA [30] and 2) ElGamal [34].

1) RSA Cryptosystem: In the RSA public key cryptosystem [30], encryption of a message m is computed as

$$E_K(m, r) = (m \| r)^K \bmod N \tag{30}$$

where K is the public key, N is the RSA modulus [30], and r is a random string. The decryption of the message m is performed as

$$\begin{aligned} D_K(E_K(m, r)) &= \left((m \| r)^K\right)^d \bmod N \\ &= (m \| r)^{Kd} \bmod N \equiv (m \| r) \end{aligned} \tag{31}$$

where d is the private key used for decryption, and $Kd \equiv 1 \bmod \phi(N)$, where $\phi(.)$ is the Euler's *Phi* function [28]. Note that the random string is used to randomize the RSA encryption of a message m. If two inputs to a mixnet contain the same message m, it is important that their encryptions are different, in order to protect the privacy of the corresponding senders.

2) ElGamal Cryptosystem: In ElGamal cryptosystem [34], the randomized encryption algorithm is

$$E_K(m, r) = (g^r \| mK^r) \bmod p \tag{32}$$

where p is a large prime, $K = g^d$ is the public key, d is the private key, r is a random string, and g is a generator [34] of the group Z_p^*, i.e., $g^{p-1} \equiv 1 \bmod p$. In some implementations of ElGamal cryptosystems [42], g is chosen from a subgroup $G \subset Z_p^*$, such that the order of G is $|G| = q$, where q is a large prime, and $p = uq + 1$, with u being an integer. The decryption of the message m is performed as

$$D_K(E_K(m, r)) = \frac{mK^r}{(g^r)^d} \bmod p = \frac{mK^r}{K^r} \bmod p = m. \tag{33}$$

In the remaining sections, the modulo operation is assumed in all the cryptographic transformations.

3) Homomorphic and Reencryption Properties of Public Key Cryptosystems: An encryption algorithm E_K is said to be homomorphic, if given two encryptions, $E_K(m_1, r_1)$ and $E_K(m_2, r_2)$, one can obtain $E_K(m_1 \odot m_2, r_1 \star r_2)$ without decrypting m_1 and m_2 individually. The operations $\odot, \star$ can be modular addition or multiplication. For additive homomorphism, the operation $\odot$ is a modular addition, and for multiplicative homomorphism, $\odot$ is a modular multiplication. The RSA cryptosystem, as well as ElGamal cryptosystem, exhibit multiplicative homomorphism [28]. For example, given two ElGamal encryptions, $E_K(m_1, r_1) = (g^{r_1} \| m_1 K^{r_1})$, $E_K(m_2, r_2) = (g^{r_2} \| m_2 K^{r_2})$, by multiplying the two encryptions, we obtain

$$\begin{aligned} E_K(m_1, r_1) E_K(m_2, r_2) &= (g^{r_1} g^{r_2} \| m_1 m_2 K^{r_1} K^{r_2}) \\ &= (g^{r_1+r_2} \| m_1 m_2 K^{r_1+r_2}) \\ &= E_K(m_1 m_2, r_1 + r_2). \end{aligned} \tag{34}$$

The homomorphic property, while being useful for many applications [89], can be misused in the case of mixnets [48], [63]. For example, as illustrated in [63], an adversary can modify inputs from two senders, $\{E_K(m_1, r_1), E_K(m_2, r_2)\}$, to be $\{E_K(1, 1), E_K(m_1 m_2, r_1 + r_2)\}$. Note that the product of each pair is the same, hence such a modification may not be detected as seen in Section V-D2.

The ElGamal cryptosystem also has the reencryption property [54]. Given an ElGamal encryption of a message m with a public key K, it is possible to reencrypt as

$$\begin{aligned} E_K(m, r) &= (g^r g^{r'} \| m K^r K^{r'}) \\ &= (g^{r+r'} \| m K^{r+r'}) \\ &= E_K(m, r + r'). \end{aligned} \tag{35}$$

Notice that by reencrypting as shown, with a random string r', the appearance of the encryption of m is changed. This is an important property that is used in the mixing operation at the stages of a mixnet, as seen in Section III-C. However, reencryption can also be misused [42], since an adversary can successfully replay the reencryption of a target sender input and then possibly trace it at the mixnet output (see Section IV-E).

B. Bulletin Board

A publicly accessible bulletin board [33], [89] is a broadcast communication channel with memory. Each sender i broadcasts its encrypted messages to the bulletin board, which contains an authenticated section for sender i, with append-only write access. However, the contents of the section are publicly readable. Such a bulletin board can be implemented robustly using multiple servers. The authenticated bulletin board, as seen in Section IV-D, provides robustness against certain attacks on mixnets. An additional advantage is that the bulletin board provides verifiability of the message to the sender, since the sender can verify if the encrypted message has been received without any error. Apart from obtaining the input batches, the stages of the mixnet may also use the bulletin board to broadcast their mixed output batches and proofs, as seen in Section IV-D.

C. Secret Sharing

In a mixnet, anonymity can be achieved by letting a single stage perform mixing operation. However, a robust mixnet design will require multiple stages (see Section IV-B). In such a design, a cooperating group of stages, particularly in a cascade mixnet, may require secrets to be shared among the group members. A (t, n) threshold secret sharing scheme [35], [90]–[94], where the threshold $t \leq n$, can be used to share a secret S between n stages. To reconstruct the secret, at least t stages have to submit their shares, which are then combined. Consequently, the secret can be unconditionally protected up to a collusion of $t - 1$ compromised and $n - t$ faulty stages [35].

D. Blind Signatures

As seen in Section VIII-B, a sender may need to obtain the signature of the stages of a mixnet on a message, before encrypting it. Then, in order to protect the privacy of the sender, the message m must not be accessed by the stages. A blind signature is a cryptographic protocol that can be used to obtain the signature without revealing the message. For example, an RSA blind signature protocol [95] can be described as follows. As illustrated in Fig. 25, a sender i blinds its message m using a random string r, and the RSA public key K of the mixnet as mr^K. The stages of the mixnet sharing the RSA private key d [79], [80], [96] obtain the blinded message and then jointly sign it as $(mr^K)^d = m^d r^{Kd} \equiv m^d r$, and send $m^d r$ to the sender i. Since the sender i knows r, it can obtain m^d, which is the message m signed by the stages of the mixnet.

E. ZK Proofs

In order to avoid replay of inputs by an adversary [42], as seen in Section IV-E, a sender may be required by the mixnet to prove knowledge of the message encrypted [44]. Also, to address the integrity of the mixnet, the stages may be required to prove the validity of the mixing operation [40], as seen in Section V. This can be achieved using interactive and noninteractive proofs [32], [97]–[99] that

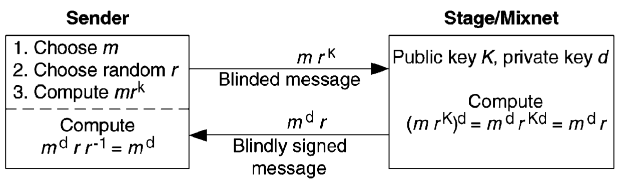


Fig. 25. *RSA blind signature protocol between sender and stage or mixnet containing stages sharing private key d.*

are cryptographic protocols implemented by an entity P (prover) to prove knowledge of a secret to an entity V (verifier). If a *proof* does not leak the secret then it has ZK [32]. Since ZK proofs require expensive public key operations including modular exponentiations, the efficiency of the mixnet using ZK proofs is dependent on them. A single noninteractive ZK proof may need up to k communication rounds, k being a security parameter, to attain high probability of error detection [99]. However, efficient noninteractive ZK proofs for mixnets in [59], [60] need only one round.

F. Cryptographic Hash Functions

Cryptographic hash functions [28] have a variety of applications in mixnets. In certain types of mixnets [8], [63] (Section V-D2), a practical collision-resistant hash function, such as SHA-1 [100], is used as a checksum to ensure the integrity of the sender inputs. A keyed hash function can be used as a MAC [28] to securely ensure integrity of the input at each stage of the mixnet (Section V-B2). A hash function is also used to compute symmetric keys [13] and to securely store inputs for avoiding replay of inputs [8], [76].

REFERENCES

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[2] M. Jakobsson, A. Juels, and R. Rivest, "Making mix nets robust for electronic voting by randomized partial checking," in *Proc. USENIX' 02*. New York: Springer-Verlag, 2002, pp. 339–353.

[3] R. Sampigethaya and R. Poovendran, *A framework and taxonomy for comparison of electronic voting schemes*, 2004, manuscript submitted for publication.

[4] ——, *Evaluating secure remote electronic voting systems*, 2004, manuscript submitted for publication.

[5] S. Parekh, *Prospects for remailers*. [Online]. Available: http://www.firstmonday.org/issues/issue2/remailers/

[6] L. Cottrell. (1994). *Mixmaster and remailer Attacks*. [Online]. Available: http://www.obscura.com/loki/remailer/remailer-essay.html

[7] C. Gulcu and G. Tsudik, "Mixing email with Babel," in *Proc. Internet Soc. Symp. Network and Distributed System Security*, 1996, pp. 2–16.

[8] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *Proc. 2003 IEEE Symp. Security Privacy*, 2003, pp. 2–15.

[9] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-Mixes: Untraceable communication with very small bandwidth overhead," in *Proc. GI/ITG Conf. Communication Distributed Systems, Informatik-Fachberichte*. New York: Springer-Verlag, 1991, pp. 451–463.

[10] A. Jerichow, J. Muller, A. Pfitzmann, B. Pfitzmann, and M. Waidner, "Real-time mixes: A bandwidth-efficient anonymity protocol," *IEEE J. Select. Areas Commun.*, vol. 16, no. 4, pp. 495–509, Apr. 1998.

[11] D. Goldschlag, M. Reed, and P. Syverson, "Hiding routing information," in *Proc. Information Hiding*, 1996, pp. 137–150.

[12] P. Syverson, D. Goldschlag, and M. Reed, "Anonymous connections and onion routing," in *Proc. IEEE Symp. Security Privacy*, 1997, pp. 44–54.

[13] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE J. Special Areas Commun.*, vol. 16, no. 4, pp. 482–494, Apr. 1998.

[14] *Onion routing*. [Online]. Available: http://www.onion-router.net/

[15] A. Young and M. Yung, *Malicious Cryptography: Exposing Cryptovirology*. New York: Wiley, 2004.

[16] H. Federrath, A. Jerichow, and A. Pfitzmann, "MIXes in mobile communication systems: Location management with privacy," in *Proc. Information Hiding*, 1996, pp. 121–135.

[17] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Protocols using anonymous connections: Mobile applications," in *Proc. Security Protocols*, 1997, pp. 13–23.

[18] L. Huang, K. Matsuura, H. Yamane, and K. Sezaki, "Enhancing wireless location privacy using silent period," in *Proc. IEEE Wireless Communications Networking Conf.*, 2005.

[19] P. Golle, M. Jakobsson, A. Juels, and P. Syverson, "Universal re-encryption for mixnets," in *Proc. RSA Conf. Cryptographers' Track*, 2004, pp. 163–178.

[20] J. Raymond, "Traffic analysis: Protocols, attacks, design issues and open problems," in *Proc. Designing Privacy Enhancing Technologies Workshop*, 2001, pp. 10–29.

[21] *Free haven project bibliography*. [Online]. Available: http://www.freehaven.net/anonbib/date.html

[22] O. Berthold, A. Pfitzmann, and R. Standtke, "The disadvantages of free mix routes and how to overcome them," in *Proc. Anonymity*, 2001, pp. 30–45.

[23] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *Proc. Information Hiding Workshop*, 2002, pp. 36–52.

[24] M. Jakobsson and A. Juels, "An optimally robust hybrid mix network," in *Proc. 20th Annu. ACM Symp. Principles Distributed Computing (PODC 2001)*, 2001, pp. 284–292.

[25] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar, "A reputation system to increase MIX-net reliability," in *Proc. Information Hiding Workshop*, 2001, pp. 126–141.

[26] C. Diaz and A. Serjantov, "Generalizing mixes," in *Proc. Workshop on Privacy Enhancing Technologies*, 2003, pp. 18–31.

[27] C. Diaz and B. Preneel, "Taxonomy of mixes and dummy traffic," in *Proc. 19th IFIP Int. Information Security Conf.*, 2004.

[28] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC , 1997.

[29] D. R. Stinson, *Cryptography: Theory and Practice*, 2nd ed. Boca Raton, FL: CRC, 2002.

[30] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, 1978.

[31] B. Möller, "Provably secure public-key encryption for length-preserving Chaumian mixes," in *Proc. CT-RSA*. New York: Springer-Verlag, 2003, pp. 244–262.

[32] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Computing*, vol. 18, pp. 186–208, 1989.

[33] C. Park, K. Itoh, and K. Kurosawa, "Efficient anonymous channel and all/nothing election scheme," in *Advances in Cryptology-Eurocrypt*. New York: Springer-Verlag, 1994, pp. 248–259.

[34] T. Elgamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[35] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[36] P. Golle and M. Jakobsson, "Reusable anonymous return channels," in *Proc. Workshop Privacy Electronic Soc.*, 2003, pp. 94–100.

[37] M. Jakobsson, "On quorum controlled asymmetric proxy re-encryption," in *Proc. PKC*, 1999, pp. 112–121.

[38] R. Dingledine, V. Shmatikov, and P. Syverson, "Synchronous batching: From cascades to free routes," in *Proc. Privacy Enhancing Technologies Workshop*, 2004.

[39] W. Ogata, K. Kurosawa, K. Sako, and K. Takatani, "Fault-tolerant anonymous channel," in *Proc. ICICS*, 1997, pp. 440–444.

[40] K. Sako and J. Killian, "Receipt-free mix-type voting scheme—A practical solution to the implementation of a voting booth," in *Advances Cryptology-Eurocrypt*, 1995, pp. 393–403.

[41] B. Pfitzmann and A. Pfitzmann, "How to break the direct rsa-implementation of mixes," in *Advances Cryptology-Eurocrypt*, 1989, pp. 373–381.

[42] B. Pfitzmann, "Breaking an efficient anonymous channel," in *Advances Cryptology-Eurocrypt*, 1994, pp. 332–340.

[43] C. Rackoff and D. Simon, "Cryptographic defense against traffic analysis," in *Proc. 25th Annu. ACM Symp. Theory of Computing*, 1993, pp. 672–681.

[44] M. Jakobsson, "A practical mix," in *Advances in Cryptology-Eurocrypt*, 1998, pp. 449–461.

[45] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[46] M. Jakobsson, "Flash mixing," in *Proc. ACM Symp. Principles of Distributed Computing (PODC)*, 1999, pp. 83–89.

[47] Y. Desmedt and K. Kurosawa, "How to break a practical mix and design a new one," in

Advances Cryptology-Eurocrypt, 2000, pp. 557–572.

[48] M. Mitomo and K. Kurosawa, "Attack for flash MIX," in *Advances in Cryptology-Asiacrypt*, 2000, pp. 192–204.

[49] M. Ohkubo and M. Abe, "A length-invariant hybrid mix," in *Advances Cryptology-Asiacrypt*, 2000, pp. 178–191.

[50] M. Abe and H. Imai, "Flaws in some robust optimistic mixnets," in *Proc. ACISP*, 2003, pp. 39–50.

[51] M. Michels and P. Horster, "Some remarks on a receipt-free and universally verifiable mix-type voting scheme," in *Advances Cryptology-Asiacrypt*, 1996, pp. 125–132.

[52] M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-centers," in *Advances Cryptology-Eurocrypt*, 1998, pp. 437–447.

[53] ——, "Mix-networks on permutation networks," in *Advances Cryptology-Asiacrypt*, 1999, pp. 258–273.

[54] M. Jakobsson and A. Juels, *Millimix: Mixing in small batches, DIMACS Techn. Rep. 99-33, 1999.*

[55] M. Abe and F. Hoshino, "Remarks on mix-networks based on permutation networks," in *Proc. PKC*, 2001, pp. 317–324.

[56] A. Waksman, "A permutation network," *J. ACM*, vol. 15, no. 1, pp. 159–163, 1968.

[57] J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako, "An implementation of a universally verifiable electronic voting scheme based on shuffling," in *Proc. Financial Cryptography (FC 2002)*, 2003, pp. 16–30.

[58] L. Nguyen and R. Safavi-Naini, "Breaking and mending resilient mixnets," in *Proc. Privacy Enhancing Technologies Workshop*, 2003, pp. 66–80.

[59] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proc. 8th ACM Conf. Computer Communications Security*, 2001, pp. 116–125.

[60] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *Advances Cryptology-Crypto*, 2001, pp. 368–387.

[61] J. Groth, "A verifiable secret shuffle of homomorphic encryptions," in *Proc. PKC*, 2003, pp. 145–160.

[62] D. Boneh and P. Golle, "Almost entirely correct mixing with applications to voting," in *Proc. 9th ACM Conf. Computer Communications Security*, 2002, pp. 68–77.

[63] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels, "Optimistic mixing for exit-polls," in *Advances Cryptology-Asiacrypt*, 2002, pp. 451–465.

[64] P. Golle and A. Juels, "Parallel mixing," in *Proc. 11th ACM Conf. Computer and Communications Security*, 2004, pp. 220–226.

[65] U. Moller, L. Cottrell, P. Palfrader, and L. Sassaman. (2003). *Mixmaster Protocol-Version 2 Draft*. [Online]. Available: http://www.abditum.com/mixmaster-spec.txt

[66] *Mixminion*. [Online]. Available: http://mixminion.net/

[67] O. Berthold, H. Federrath, and S. Kopsell, "Web MIXes: A system for anonymous and unobservable internet access," in *Proc. Designing Privacy Enhancing Technologies Workshop*, 2000, pp. 115–129.

[68] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second generation onion router," in *Proc. 13th USENIX Security Symp.*, 2004, pp. 303–320.

[69] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proc. Privacy Enhancing Technologies Workshop*, 2004.

[70] O. Berthold and H. Langos, "Dummy traffic against long term intersection attacks," in *Proc. Privacy Enhancing Technologies Workshop (PET 2002)*, 2003, pp. 110–128.

[71] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems," in *Proc. Financial Cryptography (FC 2004)*, 2004, pp. 251–265.

[72] A. Serjantov and R. E. Newman, "On the anonymity of timed pool mixes," in *Proc. Workshop Privacy Anonymity Issues in Networked Distributed Syst.*, 2003, pp. 427–434.

[73] D. Kesdogan, J. Egner, and R. Buschkes, "Stop-and-go mixes providing probabilistic security in an open system," in *Proc. 2nd Int. Workshop Information Hiding*, 1998, pp. 83–98.

[74] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in *Proc. Designing Privacy Enhancing Technologies Workshop*, 2000, pp. 96–114.

[75] G. Danezis and L. Sassaman, "Heartbeat traffic to counter (n-1) attacks: Red-green-black mixes," in *Proc. ACM Workshop Privacy Electronic Soc.*, 2003, pp. 89–93.

[76] G. Danezis, "Forward secure mixes," in *Proc. Nordic Workshop Secure IT Syst.*, 2002, pp. 195–207.

[77] R. Dingledine and P. Syverson, "Reliable MIX cascade networks through reputation," in *Proc. Financial Cryptography (FC 2002)*, 2002.

[78] P. Golle, "Reputable mix networks," in *Proc. Privacy Enhancing Technologies Workshop*, 2004.

[79] Y. Desmedt and Y. Frankel, "Shared generation of authenticators and signatures," in *Advances Cryptology-Crypto*, 1992, pp. 457–469.

[80] V. Shoup, "Practical threshold signatures," in *Advances Cryptology-Eurocrypt*, 2000, pp. 207–220.

[81] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *J. Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[82] P. Golle and A. Juels, "Dining cryptographers revisited," in *Advances Cryptology-Eurocrypt*, 2004, pp. 456–473.

[83] M. K. Reiter and A. D. Rubing, "Crowds: Anonymity for web transactions," *ACM Trans. Information Syst. Security*, vol. 1, no. 1, pp. 66–92, 1998.

[84] M. Rennhard and B. Plattner, "Introducing Morphmix: Peer-to-peer based anonymous internet usage with collusion detection," in *Proc. Workshop Privacy Electronic Soc.*, 2002, pp. 91–102.

[85] A. Pfitzmann and M. Waidner, "Networks without user observability—Design options," in *Advances in Cryptology-Eurocrypt*, 1985, pp. 245–253.

[86] R. Bohme, G. Danezis, C. Diaz, S. Kopsell, and A. Pfitzmann, "Mix cascades vs. peer-to-peer: Is one concept superior?" in *Proc. Privacy Enhancing Technologies Workshop*, 2004.

[87] G. Danezis, "Mix-networks with restricted routes," in *Proc. Privacy Enhancing Technologies Workshop*, 2003.

[88] M. Reiter and X. Wang, "Fragile mixing," in *Proc. 11th ACM Conf. Computer Communications Security*, 2004, pp. 227–235.

[89] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," in *Advances in Cryptology-Eurocrypt*, 1997, pp. 103–118.

[90] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneous broadcast," in *Proc. IEEE Fund. Comp. Sci.*, 1985, pp. 335–344.

[91] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. 28th IEEE Symp. Found. Computer Sci. (FOCS 87)*, 1987, pp. 427 437.

[92] T. Pedersen, "A threshold cryptosystem without a trusted party," in *Advances Cryptology-Eurocrypt*, 1991, p. 5 227 526.

[93] M. Stadler, "Publicly verifiable secret sharing," in *Advances in Cryptology-Eurocrypt*, 1996, pp. 190–199.

[94] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its applications to electronic voting," in *Advances Cryptology-Crypto*, 1999, pp. 148–164.

[95] D. Chaum, "Blind signature system," in *Advances Cryptology-Crypto*, 1984, pp. 153–153.

[96] Y. Desmedt, "Society and group oriented cryptography: A new concept," in *Advances Cryptology-Crypto*, 1987, pp. 120–127.

[97] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but the validity of the assertion, and a methodology of cryptographic protocol design," in *Proc. 27th IEEE Symp. Found. Computer Sci.*, 1986, pp. 174–187.

[98] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances Cryptology-Crypto*, 1987, pp. 186–194.

[99] M. Blum, A. D. Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge," *SIAM J. Computing*, vol. 20, no. 6, pp. 1084–1118, 1991.

[100] "FIPS 180-1, Secure Hash Standard," Apr. 1995: U.S. Dept. Commerce/N.I.S.T., Nat. Tech. Information Service, Springfield, VA.

[101] A. Pfitzmann and M. Hansen, *Anonymity, unlinkability, unobservability, pseudonymity, and identity management a consolidated proposal for terminology*, ver. v0.25, Dec. 2005.

[102] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proc. Workshop Privacy Enhancing Technologies (PET)*, Apr. 2002, pp. 41–53.

[103] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proc. Workshop Privacy Enhancing Technologies*, Apr. 2002, pp. 54–68.

[104] M. Waidner, "Unconditional sender and recipient untraceability in spite of active attacks," in *Advances Cryptology—Eurocrypt*, vol. 434, 1990, pp. 302–319.

[105] E. Franz, A. Graubner, A. Jerichow, and A. Pfitzmann, "Comparison of commitment schemes used in mix-mediated anonymous communication for preventing pool-mode attack," in *Proc. 3rd Australasian Conf. Information Security Privacy (ACISP'98)*, vol. 1438, 1990, pp. 111–122.

[106] G. Danezis, "Breaking four mix-related schemes based on universal re-encryption," in *Proc. Information Security Conf.*, Sep. 2006.

[107] C. Díaz, "Anonymity and privacy in electronic services," Ph.D. dissertation, Katholieke Universiteit Leuven, Leuven, Belgium, Dec. 2005.

ABOUT THE AUTHORS

Krishna Sampigethaya (Student Member, IEEE) received the M.S.E.E. degree in communications and networking, and he is currently working toward the Ph.D. degree from the Department of Electrical Engineering, University of Washington, Seattle.

He is a Graduate Research Assistant in the Network Security Lab, University of Washington, and is conducting research on protocols related to secure electronic voting and on location privacy in vehicular networks. He has conducted research in the Math and Computing Division, Boeing Phantom Works, Seattle.

Radha Poovendran (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Maryland, College Park, in 1999.

He is an Associate Professor and Founding Director of the Network Security Lab (NSL), Electrical Engineering Department, University of Washington, Seattle. His research interests include the areas of applied cryptography for multiuser environment, wireless networking, and applications of information theory to security. He is a co-editor of *Secure Localization and Time Synchronization in Wireless ad hoc and Sensor Networks* (Springer-Verlag).

Dr. Poovendran is a recipient of an NSA Rising Star Award and Faculty Early Career Awards including NSF CAREER (2001), ARO YIP (2002), ONR YIP (2004), and the PECASE (2005), for his research contributions to multiuser security.