Computer Communications 33 (2010) 420–431

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Survey on anonymous communications in computer networks

Jian Ren [a,*], Jie Wu [b]

[a] Michigan State University, East Lansing, MI 48824, USA
[b] Temple University, Philadelphia, PA 19122, USA

ARTICLE INFO

Article history:
Available online 20 November 2009

Keywords:
Communication anonymity
Source anonymous message authentication
Sender anonymity
Recipient anonymity
Relationship anonymity

ABSTRACT

Anonymous communications aim to preserve communications privacy within the shared public network environment. It can provide security well beyond content privacy and integrity. The scientific studies of anonymous communications are largely originated from Chaum's two seminal approaches: mixnet and DC-net. In this paper, we present an overview of the research in this field. We start with the basic definitions of anonymous communications. We then describe the cryptographic primitives, the network protocols, and some of the representative anonymous communication systems. We also describe verifiable mixnets and their applications to electronic voting. Finally, we briefly cite some other anonymous systems.

1. Introduction to anonymous communications

The rapid growth of Internet applications has made communication privacy an increasingly important security requirement. While end-to-end encryption can protect the data content of communications from adversarial access, it does not conceal all the relevant information that two users are communicating. Adversaries can still learn significant information about the traffic carried on the network and the physical world entities, such as the network of the sender and receiver or the network addresses of its end-to-end source and destination. The exposure of network addresses may result in a number of severe consequences. Adversaries can easily overhear all the messages and perform *traffic analysis*. Even if communication content is encrypted, routing information is still sent in the clear because routers need to know packets' destinations in order to route them in the right direction. Traffic analysis can also be done by watching particular data moving through a network, by matching the amount of data, or by examining coincidences, such as connections opening and closing at about the same time.

In a tactical military communication network, an abrupt change in the traffic pattern may indicate some forthcoming activities. This can be extremely dangerous in that adversaries can easily identify critical network nodes and then launch targeted attacks on them. This makes source privacy an essential security requirement for government and military communications.

In addition, people seeking sensitive information have a strong desire to remain anonymous so as to avoid being stigmatized or even to avoid physical or social detriment by suppressors. The freedom of the information exchange is another important issue that has received increasing attention in the last years. Some organizations, such as governments or private companies, may regard a discussion topic or a report as inconvenient or even harmful when exposed. They may thus try to censor the exchange of undesired information by either suppressing the resource providers, or if the information is protected by anonymity services, taking control of strategic regions of the network, such as gateways and proxies, and then filtering communication.

The research on privacy preserving communications was initiated by Chaum in his seminal work "untraceable electronic mail, return address, and digital pseudonyms" published in 1981 [1]. Since then, research in anonymous communications has been extended to many areas. The existing anonymous communications systems can largely be divided into four categories: cryptosystem-based schemes, routing-based schemes, broadcasting-based systems, and peer-to-peer communication systems.

The rest of this paper is organized as follows: In Section 2, we introduce some basic definitions of anonymous communications. We present some cryptographic primitives that provide source protections in Section 3. In Section 4, we review mixnet-based anonymous communication schemes. We describe DC-net-based systems in Section 5. In Section 6, we outline routing-based anonymous communication systems. We review peer-to-peer based systems in Section 7. In Section 8, we present a brief outline of verifiable mixnets and their applications to electronic voting. We conclude in Section 9 with a brief citing of some other systems.

2. Terminology

In this section, we will introduce some definitions of anonymous communications. A comprehensive reference on the

* Corresponding author. Tel.: +1 517 353 4379.
E-mail addresses: renjian@egr.msu.edu (J. Ren), jiewu@temple.edu (J. Wu).

0140-3664/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2009.11.009

definitions has been defined by Pfitzmann and Hansen, which has been updated multiple times. The most recent version was developed in 2008 [2].

2.1. Anonymity

To enable anonymity of a subject, there has to be an appropriate set of subjects with potentially the same attributes. Anonymity of a subject is defined as *the state of not being identifiable within the set of subjects, which is called the anonymity set*.

The *anonymity set* is the set of all possible subjects. It is also called the *ambiguity set* [3]. With respect to acting entities, such as the senders, the anonymity set consists of the subjects who might cause an action. With respect to acted entities, such as recipients, the anonymity set consists of the subjects who might be acted upon. In this way, both the sender and the recipient may be anonymous only with their respective anonymity set. The anonymity sets for the sender and the recipient may be disjoint, the same, or they may overlap. The anonymity set may also vary over time.

As a security requirement for anonymity, the probability that a verifier can successfully determine the real source is exactly $1/n$, where n is the number of members in the anonymity set.

2.2. Unlinkability

Unlinkability ensures that a user may make multiple uses of resources or services without others being able to link these uses together. The requirements for unlinkability are intended to protect the user's identity against the use of profiling of the operations. Hiding the relationship between different invocations of a service or access of a resource will prevent this kind of information gathering.

As a result, a requirement for unlinkability could imply that the subject and user identity of an operation must be protected so that two or more items, or operations of interest to an attacker, cannot be distinguished or related. Otherwise, this information might be used to link operations together.

Anonymity may be defined as unlinkability of an item of interest (IOI) and any identifier of a subject. In other words, anonymity of an IOI means that it is not linkable to any particular identity (ID), and the anonymity of an ID is defined as not being linkable to any IOI. With this definition, sender anonymity means that a particular message is not linkable to any sender, and no message is linkable to a particular sender. The same description can be applied to recipient anonymity.

Unlinkability differs from pseudonymity in that, although in pseudonymity the user is also not known, relations between different actions can be provided.

2.3. Unobservability

Unobservability is the state of items of interest (IOIs) being indistinguishable from any IOI (of the same type) at all. This means that messages are not discernible from random noise. Similar to anonymity sets, we have *unobservability sets* of subjects with respect to unobservability. Likewise, sender unobservability means that it is not noticeable whether any sender within the unobservability set sends. Recipient unobservability means that it is not noticeable whether any recipient within the unobservability set receives. In addition, we can also define relationship unobservability as the state that is not noticeable whether anything is sent out of a set of could-be senders to a set of could-be recipients. In other words, it is not noticeable whether within the relationship unobservability set of all possible sender-recipient-pairs, a message is exchanged in any relationship.

2.4. Pseudonymity

Pseudonyms are identifiers of subjects, unlike the subject's real names, which are fixed identifiers. Pseudonyms are generally dynamic identifiers, or names of the subjects that are hard to be linked to the real identities without the shared secret keys. In other words, a pseudonym is an identifier of a subject other than one of the subject's real names.

We can generalize pseudonyms to be identifiers of sets of subjects, but we do not need this in our setting.

The subject, which the pseudonym refers to, is the holder of the pseudonym. A subject is pseudonymous if a pseudonym is used as an identifier instead of one of its real names. Pseudonymity is the use of pseudonyms as identifiers.

Sender pseudonymity is defined as the sender being pseudonymous; recipient pseudonymity is defined as the recipient being pseudonymous. We assume that each pseudonym refers to exactly one specific holder, invariant over time, not being transferred to other subjects. Specific kinds of pseudonyms may extend this setting: a group pseudonym refers to a set of holders; a transferable pseudonym can be transferred from one holder to another subject, becoming its holder. Such a group pseudonym may induce an anonymity set: Using the information provided by the pseudonym only, an attacker cannot decide whether an action was performed by a specific subject within the set.

3. Cryptographic primitives for anonymous communications

In this section, we will outline some cryptographic primitives that can be used to provide source privacy.

3.1. Group signature

The concept of group signature was first introduced by Chaum and van Heyst in 1991 [4]. A group signature allows any member of a group to digitally sign a document anonymously without being individually identified. The group signature of a document can be verified by any verifier. As an application, a group signature scheme can be used by an employee of a large company to sign electronic documents so that a verifier can be assured that the message was originated from the company, however, without leaking the particular employee who signed it. Another application is for keycard to access restricted areas where it is inappropriate to track individual employees' movements, but necessary to secure areas to only employees in the group.

Each group signature requires a group manager. The group manager is responsible for adding and revealing of the original signer in the event of disputes. In some systems the responsibilities of adding members and revoking signature anonymity are separated and given to a membership manager and revocation manager respectively. A group signature scheme should follow these basic requirements:

- *Soundness and completeness*: Valid signatures generated by group members always verify correctly, and invalid signatures always fail verification.
- *Unforgeability*: Only members of the group can create valid group signatures.
- *Signer ambiguity*: Given a message and its signature, the identity of the individual signer cannot be determined without the revocation manager's secret key.
- *Unlinkability*: Given two messages and their signatures, we cannot tell if the signatures were from the same signer or not.
- *No framing*: Even if all other group members (and the managers) collude, they cannot forge a signature for a non-participating group member.

- *Unforgeable tracing verification*: The revocation manager cannot falsely accuse a signer of creating a signature he did not create.

Unfortunately, in the original group signature scheme [4], each time a member of the group signs a document, a new key pair has to be generated for the signer. The generation of new key pairs causes the length of both the group members' secret keys and the designated authority's auxiliary information to grow. This tends to cause the scheme to become unwieldy when used by a group to sign numerous messages or when used for an extended period of time. Some improvements have been made in the efficiency of this scheme [5–8]. However, it is hard to give an example of a group signature without going very deep into technical discussions.

3.2. Ring signature

The concept of ring signature was invented by Rivest et al. in 2001 [3]. In cryptography, a ring signature is a type of digital signature that can be performed by any member of a group of users. In a ring signature, instead of revealing the actual identity of the message signer, it specifies a set of possible signers. The verifier can be convinced that the signature was indeed generated by one of the ring members, however, he is unable to tell which member actually produced the signature.

The name *ring signature* comes from the ring-like structure of the signature algorithm. The idea behind ring signature schemes is similar to that of group signatures [4,7,8], but with some variations. First of all, unlike a group signature, a ring signature scheme does not require a group manager to administrate the set of ring members. The actual message signer has the freedom to select all the ring members and sign whatever messages he likes. Second, in a group signature scheme, the group manager can recover the real identity of the actual message signer. This makes a group signature only look indistinguishable to the verifier but not to the group manager. The group manager can even revoke the anonymity of misbehaving signers. However, for a ring signature, since there is no group manager, nobody can revoke the anonymity of the ring signature.

Ring signatures can provide unconditional signer-ambiguity from any other members in the anonymous set. Ring signatures are provably secure in the random oracle model. Since the introduction of the ring signature, several ring signature schemes have been proposed to increase its efficiency, including discrete logarithm based ring signatures [9–11], bilinear pairing-based ring signatures [12–15] and identity-based ring signature schemes [14,16]. Some variations of ring signature schemes are also presented in literature [15,17–20].

4. Mixnet-based schemes

In the past two decades, originated largely from Chaum's mixnet [1] and DC-net [21], a number of anonymous communication protocols have been developed. They can largely be divided into mixnet-based systems and DC-net-based systems.

All secure mix systems are based on Chaum's seminal work as a means of sender anonymity. The mixnet family protocols use a set of "mix" servers that mix the received packets to make the communication paths (including those of the sender and the recipient) ambiguous (See Fig. 1). They rely on the statistical properties of background traffic, also referred to as *cover traffic*, to achieve the desired anonymity. The security of mixnet is based on the trust relationship of the mixers, and cannot provide unconditional anonymity.

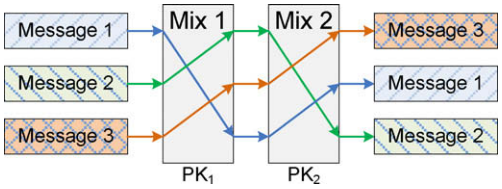

Fig. 1. Illustration of Chaumian mixnet.

4.1. Chaum's mix network

Chaum's research has been applied to many areas as a central tool for many applications such as message-based email and flow-based low latency communications. Comparing to information-theoretically secure sender anonymity schemes, such as DC-net (see Section 5), the most important advantage of mixnet is its smaller communication overhead.

For anonymous email applications, Chaum's mixnet can be viewed as a public-key cryptographic primitive that takes as input a number of ciphertexts, encrypted using public keys of the relay servers, called *mixes*, decrypts and shuffles them and finally outputs a random permutation of plaintexts. To ensure anonymity, multiple mixes with secret permutations shared between them are proposed so that even if one or several of the mixes are corrupted, the correspondence between the items in its input and those in its output is still hidden. As a result, this mix system can provide source and destination address privacy for the communicating parties.

For multiple cascade, or series of mixes, the structure of the mixes is analogous to message transmission with multiple envelopes constructed by a sender, with the address of the first mix to be on the outermost envelopes, and the last mix's address on the innermost envelope. These envelopes are implemented through encryption using public keys of the mixes. The idea of cascaded envelopes is called onion in onion routing [22]. In this case, it suffices that one of the mixes is trustworthy.

Upon receiving the message, the first mix peels off its envelope, or layer of the onion, to get the second envelope through decryption using its private key. The second envelope will provide the address of the second mix that it can forward the message to. The message will be encrypted using the public key of the second mix that only the second mix can decrypt. This process will be repeated until the last mix decrypts the message and removes the final envelope. Then, the recipient's destination address and the real message will be exposed, and the message can be delivered by the last mix.

In [23], Pfitzmann and Pfitzmann have proved that Chaum's scheme [1] may not provide necessary unlinkability. In Chaum's scheme, the encryption function was assumed to be deterministic. As a result, everybody can take an output message, encrypt it again and check with the input messages they obtain. In this way, the mix can be reversed. To prevent this re-encryption and possible size matching of the incoming flow and output flow, all packets will be made the same size through random string padding. The output messages from the mix will be indistinguishable to adversaries, and therefore we can prevent traffic analysis of network transmissions. This will also ensure that no item is processed more than once. By discarding the repeated input messages, replay attacks can also be prevented. Otherwise, an attacker can repeat the input message and observe which output message is repeated. In this way, the relation of input messages and output messages can be discovered.

Chaum also proposed to use digital pseudonyms to provide untraceability for return addresses. However, it has been analyzed in [23] that in case RSA [24] is used for public-key encryption, the

resulting mixes can be broken by an active attack that is feasible in a typical mix-environment.

Chaum's mix was extended in Babel [25]. Unfortunately, Babel's replies for forward messages are only indistinguishable to passive observers. Although Pfitzmann–Pfitzmann [23] showed an attack against the RSA implementation of Chaum's mix scheme, the concept itself was not broken, but was refined. In fact, if the ElGamal public-key cryptosystem [26] is used in the mixnet design, this attack is no longer feasible [27] since the ElGamal public-key scheme can provide semantic security. Almost all mixnets proposed from then on are based on the ElGamal public-key scheme.

Intensive research has been done to realize a robust mixnet that withstands the malicious behavior of servers and users. Much of the research focuses on publicly verifiable mixnets [28,29]. A privately convincing mixnet was introduced in [30], where only the mix servers can convince themselves of correctness. In exchange for this limitation, this kind of mixnet can provide better efficiency. Unfortunately, this particular construction can lose anonymity in the presence of a malicious user even when all servers are honest [31,33]. However, the principle has been adopted by several of the latest schemes [32].

Another interesting attempt is described in [34], where the processing is very fast as long as the mix servers are honest. If a malfunctioning server is detected, the input data is processed by a full-fledged robust mixnet. However, a malicious user colluding with the first mix-server can break anonymity at the risk of having the server accused [33,35].

Several mixnets have also been designed based on zero-knowledge proofs and stronger security assumptions to guarantee delivery or to detect and exclude misbehaving participants. These schemes include flash mixes [36], hybrid mixes [37,38], and provable shuffles [39,40]. Recently, the concept of anonymity under chosen ciphertext attacks (CCA-anonymity) for mixnets was proposed by Abe and Imai in [33]. CCA-anonymity allows an adversary to use multiple mixing rounds to mount a chosen ciphertext attack on the mixnet. A scheme that can achieve CCA-anonymity was proposed in [41].

4.2. Anonymous remailer

Cypherpunk remailers [42,43] are the first widespread public implementation of mixnet. They are also called *Type I remailers*. Type I remailers attempt to limit the feasibility of traffic analysis by providing an anonymous store and forward architecture. Type I remailers forward messages between several systems before reaching their destinations with identities stripped at each link. In addition, to prevent replay attacks, each Type I remailer keeps a log of sent messages, but they never create a database of identities. Later on, Cottrell implemented the Mixmaster systems [44,45], or *Type II remailers*. Type II remailers assume that every network connection is being monitored. In order to protect emails from network traffic monitoring, Type II remailers include message padding, message pools, and other mix features. Type II remailers are much more resistant to traffic analysis, unreliable nodes, and other attacks than Type I remailers.

However, Type II remailers still implement reusable reply blocks, such as those in the Cypherpunk remailer and in Babel. They pose a security risk – by their very nature they let people send multiple messages through them. An attacker can use this property to trace the recipient's path: if two incoming batches both include a message to the same reply block, then the next hop must be in the intersection of both outgoing batches.

Mixminion proposed in [46] is called a *Type III remailer*. It uses a free-route mixnet just like Mixmaster [45] and Babel [25] to provide strong anonymity, and to prevent eavesdroppers and other attackers from linking the senders and the recipients. Every E-mail passes through several mixes, and no single mix can see any more of the path besides the immediately adjacent mixes; therefore, no mix can link the message senders with the recipients.

Messages in Mixminion are composed of a header section and a payload. Each header is split into a main header and a secondary header, encrypted under the public keys of intermediary mixes. Miximinion allows messages to be sent from the message sender to the recipient in one of three ways: forward, direct reply, and anonymized reply. In the message forward, only the sender will be anonymous. In the direct reply, only the recipient will be anonymous. In the anonymized reply, both the sender and the recipient will be anonymous.

The Mixminion breaks each message into uniformly-sized packets, pads the packets to a uniform size, and chooses a path through the mix network for each packet. The Mixminion does not implement reusable reply blocks. Instead, Mixminion provides only single-use reply blocks (SURBs) to allow anonymous recipients. The Mixminion protocol makes reply messages indistinguishable from forward messages even for the mix nodes. Thus forward and reply messages can share the same anonymity set. However, by making the forward messages and replies indistinguishable, it also makes it more difficult to prevent a *tagging attack*, which is an active attack in which a message is modified by altering part of it so that an attacker can recognize the tag and trace the message later on in the path. This is because the author of the SURB cannot predict the message that will be attached to it, and a hash of the entire message cannot be included in the SURB. Mixminion uses cryptographic checksums to protect the headers to ensure that the address information contained in the headers is destroyed if the payload is modified by an adversary.

Unlike Type I and Type II remailers that use SMTP [47] as their underlying transport mechanism, Mixminion clients and nodes communicate using a forward secure encrypted channel based on TLS [48]. TLS allows the establishment of an ephemeral shared secret session key and an encrypted tunnel using Diffie–Hellman key exchange [49]. To ensure that the receiving end is the one intended by the sender of the anonymous message, the receiving node needs to sign the ephemeral key. As soon as the session key has been established, the parties destroy their Diffile–Hellman keys and begin sending messages through the tunnel. The parties will perform a standard key update operation to generate a fresh session key and delete the old key material after each message. Key updates do not require any public-key encryption, which can be relatively fast.

Link encryption can provide forward secrecy in that, once the ephemeral link keys have been deleted, not even the nodes that exchange messages can decrypt or recognize the messages intercepted on the links. Additionally, link encryption makes active and passive attacks on the network links more difficult. Since each message specifies the identity of its successor mix in the path, it is difficult for an attacker to mount a man-in-the-middle attack to modify messages, inject messages to a node, or delete messages. A TLS tunnel can also be used to complicate message delaying attacks.

However, the link encryption offers only limited protection against traffic analysis. Though the encrypted links between honest nodes prevent an adversary from recognizing even his own messages, without padding, the adversary can still measure the traffic that is being transmitted.

The implementation of Mixminion also brings some practical issues [50]. Since the message transmission in Mixminion is unreliable, it is important to implement mechanisms for retransmission and forward error correction (FEC). Such schemes are not trivial to implement; in addition, these schemes may lead to traffic analysis. The security of this issue requires all clients to have full knowledge of the network remailer, which has been proven to be a scalability problem, and a distributed directory service has to be specified to distribute this information.

4.3. Onion routing

Onion routing was developed by Reed, Syverson, and Goldschlag [22,51,52]. It is a distributed overlay network designed to anonymize TCP-based communications over a computer network according to the principle of Chaum's mix cascades [1]. Since onion routing provides an anonymous socket connection through the mixes, it can be easily used by many applications like Web browsing, secure shell, and instant messaging.

The onion data structure, or simply onion, is composed of layer upon layer of encryption wrapped around payload, see Fig. 2. Each onion router is a store-and-forward device that accepts fixed-length messages, performs cryptographic operations on the messages, and then forwards the messages to the next node in the routing path, called *onion router* or OR. When an onion router receives a message, it knows the immediate predecessor of this message and to whom it should be passed. Each onion router removes a layer of encryption using its own private key, which will also uncover routing instructions of the next onion router. This process will be repeated until the message is being delivered to the final onion router.

Although onion routing may be used for anonymous communications, it differs from anonymous remailers [25,44] in that onion routing provides real-time and bidirectional communications, without requiring that the network nodes know the message source identity or location. Onion routing provides application independent anonymous connections, such as anonymous mail, as well as other applications. In onion routing, the clients choose a path through the network and build a circuit, in which each network onion router knows its predecessor and successor, but no information of other nodes in the circuit. In this way, the intermediary nodes will have no knowledge of the origin, destination, and content of the message.

Although Chaum's mixnet could store messages for an indefinite amount of time while waiting to receive an adequate number of messages to mix together, the onion routing is designed to pass information in real time, which limits mixing and potentially weakens the protection.

Note that at each hop, the onion shrinks as a layer is peeled off. To prevent route information inferring from the monotonically diminishing size, a random bit string in the size of the peeled off layer is appended at the end of the payload before forwarding. Only the last onion router knows how much of the payload he receives is padding and how much he receives is the message. In fact, even a constant size onion might be traced unless all onions have the same size. To solve this problem, in onion routing, the size of the onion is fixed. This requires the initiator proxy to pad the central payload according to the size of the onion. When the onion arrives at the recipient's proxy, the size of the onion and the padding will remain the same.

Onion routing's overhead is relatively small. Connection setup overhead is typically much less than one second and appears to be no more noticeable than other delays [52], such as normal Web connection setup on the Internet. The computationally expensive public-key cryptographic operation is required for the onion routers only during the connection setup phase.

Onion routing provides a strong degree of unlinkability so that an eavesdropper cannot easily determine the sender and receiver of a given message. Onion routing's network of core onion routers is distributed, and under the control of multiple administrative domains, so no single onion router can bring down the network or compromise a user's privacy. However, onion routing does not provide perfect sender or receiver anonymity against all possible eavesdroppers. In fact, it is possible for a local eavesdropper to observe the activities of an individual node for message transmission and receiving. It does not provide any absolute guarantee of privacy; rather, it provides a continuum in which the degree of privacy is generally a function of the number of participating routers versus the number of compromised or malicious routers.

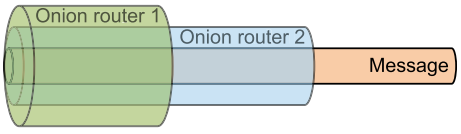

Fig. 2. Onion routing.

4.4. Tor: the second-generation onion router

Tor was developed in 2004 by the onion router project team as the second-generation of the onion router [53,54]. It was developed to address the limitations of onion routing. Tor provides perfect forward security so that users can connect to Internet sites without revealing their logical or physical locations to those sites or to observers. It also provides location-hidden services via rendezvous points, so that servers can support authorized users without giving an effective vector for physical or online attackers. Tor also provides congestion control, directory servers, integrity checking and configurable exit policies. Tor provides these protections even when a portion of its infrastructure is compromised without requiring any special privileges or kernel modifications.

One of the major vulnerabilities for a hidden service in Tor is the server's selection of the first and last node in the communication path. To a first approximation, if an adversary can watch the edges of a Tor circuit, then he can confirm who is communicating.

Tor differs from other deployed systems for traffic analysis resistance in security and flexibility. Mixnets, such as Babel [25], Mixmaster [45] and Mixminion [46], provide anonymity at the expense of introducing comparatively large and variable latencies, making them unsuitable for applications such as Web browsers. Tor was designed to provide communication anonymity for low-latency and interactive network traffic by frustrating attackers from linking communication partners, or from linking multiple communications to or from a single user. Tor can handle a variety of bidirectional protocols.

Tor network is an overlay network. Tor uses a traditional network architecture through a list of volunteer servers downloaded from a directory service. Each onion router runs as a normal user-level process without any special privileges. Each onion router also maintains a TLS [48] connection to every other onion router.

The simplest low-latency designs are trusted single-hop proxies such as the Anonymizer [55] and Web MIXes (known as *cascades*) [56]. They are easy to implement, but their designs require all users to trust the single anonymizer proxy. Though concentration of traffic to this single proxy expands the anonymity set, this approach is vulnerable to traffic analysis if the adversary can observe all traffic entering and leaving the proxy [57]. Tor is a distributed-trust, circuit-based anonymizing system. As a distributed-trust anonymizing system, Tor needs to prevent attackers from adding too many servers and thus compromising user paths. Tor relies on a small set of well-known directory servers, which are run by independent parties, to decide which nodes can join.

Establishing circuits is computationally expensive and typically requires public-key cryptography. However, because a circuit crosses several servers, and each server only knows the adjacent servers in the circuit, no single server can link a user to his communication partners.

Circuit-based designs must choose which protocol layer to anonymize. Making this protocol-layer decision requires a compromise between flexibility and anonymity. Application-level anonymizers can accept application-level protocols and relay the

application requests themselves. However, a system that understands the application can strip identifying information from requests, limit the number of requests that leave the network, and even possibly minimize the number of connections. On the other hand, an IP-level anonymizer can handle nearly any protocol, even ones unforeseen by its designers. The problem is that these systems are more complex and less portable, since they require kernel-level modifications to some operating systems. Tor is designed to provide TCP-level anonymity which presents a middle approach. It is application neutral by treating application connections as data streams rather than raw TCP packets, meanwhile avoiding the inefficiencies of tunneling TCP over TCP.

Onion routers communicate with one another, and with users' onion proxies, via TLS connections. A user's onion proxy uses an iterative mechanism to construct circuits, negotiating a symmetric key with each onion router on the circuit. The bi-directional channel is used at each stage to perform an authenticated Diffie–Hellman key exchange. TLS conceals the data in the connection with perfect forward secrecy, and prevents an attacker from modifying data on the wire or impersonating an onion router [58].

Each onion router maintains two keys: a long-term identity key and a short-term onion key. The identity key is used to sign TLS certificates, router descriptors, and directories, while the onion key is used to decrypt requests from users to set up a circuit and negotiate ephemeral keys. The TLS protocol also establishes a short-term link key when communicating between onion routers. Short-term keys are rotated periodically and independently to limit the impact of key compromise.

One of the notable differences between Tor and previous research is that Tor does not claim to offer security against even passive global observers. Tor opts to provide security services through being highly usable and cheap to operate [53]. However, vulnerability to passive adversaries has made Tor fragile against possible routing and connection recovery [59,60].

Tor also proposed to use location-hidden services via rendezvous points. In Tor, clients negotiate rendezvous points to connect with hidden servers. Reply onions are no longer required. One of the major vulnerabilities for a hidden service in Tor is the server's selection of the first and last node in the communication path [61]. An attack against this early architecture was demonstrated by Øverlier and Syverson [61]. If an adversary can watch the edges of a Tor circuit, then he can confirm who is communicating due to the low-latency of traffic flowing over the circuit. The main idea behind this attack is that an adversary can open multiple connections to the hidden server, sequentially or in parallel, and control the flow of traffic towards the server. Through a single hostile Tor node, it is possible to reveal the location of a hidden server. Fortunately, they also recommended fixes in [61]. These changes require no operational increase in network overhead and are simple to make. Recently, an IO-automata formal model of an onion-routing protocol was proposed [62] to characterize when anonymity and unlinkability are guaranteed.

4.5. Web MIXes

Web Mixes was proposed by Berthold, Federrath, and Köpsell in [56]. It was designed for anonymous and unobservable real-time Internet access that can prevent traffic analysis as well as flooding attacks.

The complete system of Web Mixes consists of three logical parts: the Java Anon Proxy (JAP) on the client-side, the MIXes, and the cache-proxy on the server-side. The JAP is a program that is installed locally on each user's computer. A MIX scrambles the order of data streams and changes their coding using cryptographic techniques to make traffic correlation attacks difficult. The MIXes are simple computers connected via the Internet. They form a logical chain, called *MIX-cascade*. The first MIX receives data sent by the JAPs. The last MIX sends the data to the cache-proxy. By means of constant dummy traffic, all senders send messages at any time to create the same anonymous group. If necessary, random data indistinguishable from genuine encrypted traffic is generated. The cache-proxy sends the data to the Internet and receives answers from the servers. These three parts are concatenated into a chain to build an anonymous tunnel. All network traffic to be anonymized is sent through this tunnel.

Several attacks against the basic concept of MIXes have been proposed to obtain user information or stop the service. To prevent passive attacks, MIXes proposed that dummy messages would be sent from the client of a communication into the MIX network to make traffic analysis harder. Sending of dummy messages guarantee that all users send the same amount of data during each time slice. The encryption of traffic also makes it infeasible for the attackers to distinguish between dummy and real information. MIXes also proposed that each MIX sends dummy messages to the user if the user does not receive real data. This ensures that each user receives the same amount of data during each time slice.

The MIXes also proposed a chop-and-slice algorithm so that large messages are chopped into short pieces of a specific constant length *slice*. Each slice is transmitted through an anonymous MIX channel. When there is no active communication request, the active users send dummy messages so that nobody can get the information of the starting time or duration of a communication, since all active users start and end their communications at the same time. This design can prevent an adversary from determining the active anonymous channel and the communication parties.

5. DC-net systems

DC-net was first proposed by Chaum in [21]. It allows participants to send and receive messages in an arbitrary network anonymously. Under the assumption of a reliable broadcast network, the anonymity of the senders is proved to be unconditional. DC-net is a secure multi-party computation protocol. It provides provable sender and receiver anonymity without relying on a trusted third party. A very attractive feature of the basic DC-net is its non-interactivity. This means that any party in the group can publish its message in a single broadcast round, with no party to party communications required. This feature is not possible for mixnet-based communication protocols.

Mixnets have seen broad exploration in literature and have served as the basis for several anonymous systems. However, by contrast, DC-nets have remained relatively neglected, apart from a small scattering of papers [63–67]. One possible reason for this is that DC-nets cannot operate in proxy like mixnets.

The main idea of DC-net is that each participant shares secret coin flips with other participants in pairs. The parity of the flips a participant has seen is then broadcasted to the entire group of participants. Since each flip is broadcasted twice, the total parity is, therefore, even. If a participant needs to send a message, he deliberately flips the parity and broadcasts. This causes the total parity to be odd, which indicates transmission of a bit. Unless all of the nodes who flipped coins with the sender reveal their coin flips among themselves, no one except for the initiator knows who sent the message.

The basic techniques for the sender untraceability can also be described using superposed sending, which was first proposed by Waidner and Pfitzmann in [66]. The main idea is that between each pair of participants there is a shared secret key k over a finite group $\mathbb{F}_p$. One participant will use the key as k, and the other participant will use the key as $p - k$. For message sending, each participant adds his messages and all his secret keys and broadcasts to every participant. Since each secret key has been added and subtracted

exactly once, the global sum is just the sum of all message characters. In particular, if only one of these is non-zero, the global sum is this character.

The untraceability of the message senders is proved to be unconditional. This means that the sender is untraceable, even if the attacker is computationally unbounded, able to eavesdrop on communications between any two of the participants, and can control an arbitrary subset of the set of participants.

Waidner and Pfitzmann also pointed out that it is unrealistic to require a reliable broadcast network in some networks. Waidner proposed an improvement in [67] to remove this requirement by using the fail-stop Byzantine key generation protocol. The resulting DC-net is called DC^+-net.

Unfortunately, DC-nets suffer from the transmission collision problem through straightforward jamming by malicious players. The malicious players can launch a denial-of-service attack by choosing to send a message every round of coin flips, or by simply dropping out of the protocol to disrupt the entire DC-net without being identified. In this way, he can prevent the honest participants from delivering the messages. This attack cannot be detected easily since each node is as anonymous as any initiator.

To solve this problem, Chaum proposed to detect dishonest players via a system of traps in a multi-round protocol. Prior to message transmission, a reservation protocol takes place. In this protocol, each participant must reserve exactly one slot in each reservation phase. Before each reservation phase, he decides whether to use this slot to send a real message or to send a trap. To jam the DC-net, a dishonest player must transmit a message in a slot that was not assigned to him. But if the slot that he tries to transmit is a trap, then the attacker may be detected during a decommitment phase.

Unfortunately, even a computationally limited attacker can forge a trap for an arbitrary slot. In [66], Waidner and Pfitzmann presented a multi round solution. It is based on the idea of setting traps during a reservation phase. However, similar to the Chaum's protocol, this scheme can only identify one dishonest player for a given trap. It seems that no obvious method for fault recovery is available, apart from re-broadcasting.

In [65], Bos and den Boer proposed a collision resolve scheme. In their solution, a different header is added to each message so that the participants know whether a message is already in the transmission to prevent further message transmission. However, this scheme is very complex since the size of the field must be sufficiently large to give a reasonable probability that the header messages are different.

In [64], von Ahn, Bortz, and Hopper considered a constant-round anonymous broadcast protocol. Their scheme was essentially a DC-net variant with an initial partitioning of players into autonomous groups. The secret distribution of each invocation is accomplished through a secret-sharing protocol. The correction of pads is proven via a *cut-and-choose* protocol. In the optimal case, the protocol requires three broadcast rounds, and has computational complexity $\mathcal{O}(n^2)$ where n is the total number of parties in the DC-net. In the presence of cheating players, the communicational complexity rises to $\mathcal{O}(n^4)$ [63,64].

Herbivore is a peer-to-peer scalable anonymous communication system [68]. It was introduced in 2002 by Goel et al. There are two components in the Herbivore system. At the lower level, a *round protocol* governs how bits are sent among the participating nodes. Herbivore also employs a *global topology control* algorithm to divide the network into smaller anonymizing cliques. Users from the small cliques communicate within them using DC-nets. Herbivore guarantees that each clique will have at least k nodes, where k is a predetermined constant that describes the degree of anonymity offered by the system. Later on, in [63], Golle and Juels proposed asymmetric constructions to detect cheating and make DC-net robust to disruption. Their scheme is similar to the approach described in [64] in that they employed cryptographic proofs of correctness rather than traps to detect cheating. However, a different strategy for padding computation was employed. Their protocol may be completed in just two broadcast rounds and with computations complexity $\mathcal{O}(n^2)$ even in the presence of faults.

Other protocols based on DC-net also include Xor-trees [69,70]. Xor-trees was proposed by Dolev and Ostrovsky to provide sender-, receiver-, and sender–receiver anonymity. Xor-trees only allows a single user to send at any one time in a Xor-tree. Therefore, it is also subject to performance degradation due to collisions as the number of users increase.

6. Network routing-based techniques

In this section, we will describe some routing-based anonymous communications schemes.

6.1. Crowds

Crowds was designed by Reiter and Rubin [71] to defend against internal attackers and a corrupt receiver. It provides users with a mechanism for anonymous Web browsing. Crowds introduced the concept of users blending into a crowd of computers, and many of the concepts are used in newer systems (e.g. Tarzan [72]). The main idea behind Crowds' anonymity protocol is to hide each user's communications within the actions of many others by routing them randomly within a group of similar users. Crowds makes it hard for a corrupt group member or local eavesdropper to tell whether the user is the actual sender, or is simply routing another user's message. In addition, Crowds was designed to offer security against attacks from collaborating Crowds members, and the end server.

In the Web transaction model, a user first joins a "crowd" of other users. The user's request to a Web server is first passed to a random member of the crowd. That member can either submit the request directly to the end server or forward it to another randomly chosen member, and in the latter case this process can be repeated in the next members. When the request is eventually submitted, it is submitted by a random member. This design prevents an adversary, or even the crowd members, from identifying its true initiator, since the initiator is indistinguishable from a member that simply forwards a request from another.

Comparing to proxy-based anonymous web transactions, such as Anonymizer [55], Crowds provides protection against a wider range of attacks than proxies do. In particular, proxy-based systems are vulnerable to passive attackers in control of the proxy and single point failure attacks.

Crowds works by collecting a group of nodes into a geographically diverse Crowds that performs Web transactions on behalf of its members. Each node servers as a proxy for a given initiator from the group. Each node in the crowd can employ the crowd to issue requests to Web servers in a way that prevents Web servers and other crowd members from determining who initiated those requests.

An initialization message is routed from the initiator to a series of nodes in the crowd, forming a path for all future messages from the initiator. Whenever a crowd member receives a request from another node in the crowd, it makes a random choice to either forward the request to another crowd member it chooses randomly, or to submit the request to the end server. The random choice is biased in favor of forwarding. That is, there is a system parameter larger than 1/2 that indicates the probability with which a node will choose to forward. The value of this parameter also impacts the anonymity of this system. This path is maintained for a limited period of time, after which all paths must be reformed.

Based on random forwarding, Crowds can provide communication anonymity when the adversary is the end server or even a node on the routing path, since the node is unable to learn the initiator of a request sent along that path. The use of a crowd offers receiver anonymity against eavesdroppers who can observe all communications involving the user's machine.

Crowds provides no anonymity against a global attacker or a local eavesdropper. The anonymity of initiators is based on the assumption that the members in the crowd cannot know if the previous node was the actual requester or was just passing the request along. However, it has been demonstrated that Crowds is vulnerable to predecessor attacks [73,74]. This is because, if a node repeatedly requests a particular resource, they can eventually be linked.

6.2. Buses for anonymous delivery

Anonymous communication that can be viewed as a bus system was first introduced by Bos and den Boer in [65]. This idea was further extended by Beimel and Dolev in [75] for messages to travel on the network so that each piece of information is allocated a seat within the bus. Routers are chosen and buses traverse these routes either through deterministic or randomized schedules. Since the buses traverse the network in fixed routes, the adversary cannot learn whether there is any communication between the nodes or not.

Since time and communication complexity are two conflicting design issues, Beimel and Dovel proposed both a communication optimal protocol and a time optimal protocol. In the communication optimal protocol, the message complexity is 1, time complexity is $\mathcal{O}(n)$, and the buffer complexity is $\mathcal{O}(n^2)$. In each time unit, only one node sends a message to another node, which means only one bus traverses the communication graph. The bus rotates through the ring that has n^2 seats, where n is the number of nodes. Seat $s_{i,j}$ is used to communicate a message from node i to node j encrypted either using the symmetric key shared between these two nodes, or the public key of node j.

Each time the bus gets to node i, node i checks which messages were sent to it by decrypting the n messages located in the ith column and ignoring the ones containing dummy information. Then it changes the message in its seat (i,j) to either a message it wants to send to node j or a dummy. The semantic security of encryption makes the passive adversary unable to tell whether a seat contains a real message, and therefore, whether two nodes are communicating.

In the time optimal protocol, the time complexity is the distance between the two nodes. In this protocol, two buses travel through every node in each direction. The nodes transfer seats from one bus to another according to the shortest path criteria. Similar to the previous protocol, privacy is provided through, first, all messages are encrypted before they are assigned to seats, and second, dummy messages are sent if there is no real message. This protocol has optimal time for message arrival, which is the number of links in the shortest path between the receiver node and sender node. The communication complexity is the number of buses, i.e., $2m$, where m is the number of edges in the graph.

For routing-based anonymous communication protocols, a general routing selection strategy was described in [76]. In this research, the concept of anonymity degree was defined and used to evaluate an optimal route selection strategy. A natural result is that variable path-length strategies perform better than fixed-length strategies.

7. Peer-to-Peer networks

In this section, we briefly describe two representative peer-to-peer anonymous communications systems. In [1], Chaum pointed out that if each participant in the mix acts as a mix for others, then the overall security can be improved. This has been implemented in both systems.

7.1. Tarzan

Tarzan was designed by Freedman and Morris in 2002 [72]. It is a peer-to-peer anonymous IP network overlay. A message initiator chooses a path of peers pseudo-randomly in a way that adversaries cannot easily influence. It achieves its anonymity with a layered onion encrypted connection, replayed through a sequence of intermediate nodes. Each intermediate node acts as a mix for others. In addition, a scalable cover traffic is used to prevent global adversaries from identifying the communication source through traffic analysis.

Tarzan uses a somewhat restricted network topology for packet routing. Each node maintains persistent connections with a small set of other nodes, called a *mimics*. Routes of anonymous messages are created only through mimics and between mimics to avoid traffic analysis through links with insufficient traffic. In this way, it provides a tradeoff between efficiency and security.

Tarzan enables client applications on participating hosts to communicate with non-participating Internet servers through special IP tunnels. The two ends of a tunnel are a Tarzan node running a client application and a Tarzan node running a network address translator (NAT), which is also responsible for forwarding the client's traffic to the ultimate Internet destination.

Tarzan operates in three stages. First, a node running a client application selects a set of nodes and establishes a routing path through the overlay network. Second, the source routing node establishes a tunnel using these nodes. Finally, it routes data packets through this tunnel. Unfortunately, a security weakness of Tarzan is that the selection of neighboring nodes for the mimic's structure is done on the basis of a network identifier or address, which is vulnerable to spoofs.

In a preliminary version of the Tarzan [77], each node is required to know a random subset of other nodes in the peer-to-peer network. Since the network is very large, the nodes have a high churn rate. As a result, any single node only knows a small subset of other nodes. An adversary node can distinguish the source node of the connection with a very high probability if it can corrupt one node and also get knowledge of its successor and predecessor nodes [78]. This problem has been fixed in the final version by requiring each node to know all others. However, this version is clearly less practical.

7.2. MorphMix

MorphMix is another peer-to-peer system for anonymous Internet usage. It was developed by Rennhard and Plattner also in 2002. The architecture and the threat model of MorphMix is similar to Tarzan. A crucial difference between MorphMix and Tarzan is that in Tarzan, the route is specified by the source, while in MorphMix, the route is chosen by the intermediate nodes. The initiator only selects the first intermediate node. Each node along the anonymous tunnel then picks the following node. The advantage of this design is that each node only has to manage its local environment consisting of its current neighbors, which is nearly independent of the system size.

MorphMix is an application-level mixnet using TCP between mixes. It operates completely in the user space. In MorphMix, each participant is also a mix at the same time. In other words, all participants are peers. The set of mixes is a dynamic system of unreliable nodes that may join and leave at any time.

In MorphMix, each node does not know if the previous node in the path is the initiator or if that node is merely a relaying node. Therefore, MorphMix can provide plausible deniability. MorphMix

allows intermediate nodes to choose the route through the network. This design makes MorphMix vulnerable to colluding attacks since the first colluding mix on the path can choose only colluding mixes. For this reason, MorphMix employs a collusion detection mechanism to identify the misbehavior of any cliques in the selection of nodes in the path. This prevents subverted nodes from routinely running attacks on the network, but does not provide security in every case.

Tabriz and Borisov presented an attack on the collusion resistance mechanism of MorphMix [79]. They demonstrated that colluding adversaries can compromise a significant fraction, or even a majority in some case, of all anonymous tunnels. Their results suggest that mechanisms based solely on a node's local knowledge of the network are not sufficient to detect colluding adversarial behavior in a peer-to-peer system.

8. Verifiable mixnets and applications to E-voting

Verifiable mixnets are specially designed mixnets which are robust against subverted servers from denying service. The correct functioning of any given mix can be verified with a proof of its correct functions. The primary motivation of verifiable mixnets is to enhance the security of mix-based election protocols, where both universal verifiability of vote delivery and privacy are of great importance.

Most of the design of verifiable mix is implemented through a cascade of mixes. The first verifiable scheme was proposed in 1993 by Park et al. [27]. In this scheme, messages are encrypted using the ElGamal public-key cryptosystem. Unlike Chaum's original mixes for anonymous channels, in which the message length is proportional to the number of mixes, in this scheme, all messages have the same fixed length that is irrelevant to the number of mixes. They used the proposed *cut-and-choose* strategy to provide fairness in e-voting, which means that if some vote is disrupted, no one can obtain any information about all the other votes. This property assumes that partial results will not affect a re-election. For this scheme, the computational complexity for verification is proportional to the number of mixes, more precisely, $\mathcal{O}(m \log \varepsilon)$, where m is the number of mixes, and ε is the acceptable error probability ($\sim 2^{80}$).

In 1994, Pfitzmann proposed two attacks to the scheme proposed in [27] against the anonymous channels and the secrecy of the voter in the election scheme [80]. In the first attack, the efficient channel was broken completely. This attack is based on analysis of the invariants at different mixes, which is similar to the attacks discussed in Section 4.1 [23]. The second attack is a chosen ciphertext attack. The main idea is that if a dishonest sender Alice wants to find out what Bob sends, she can raise a ciphertext of Bob to a power of some random number and transmit the message as her own message, and the received final output will also be raised to this power. By comparing the final list of messages, the source of the message can be discovered. As a result, the mixnet is not secure even if all mixes are honest. However, this attack can be easily fixed by adding redundancy into the encrypted messages.

An active attack was also presented in [23]. In addition, Pfitzmann also pointed out that the threat model is somewhat weaker than the original mixnet proposed by Chaum. In particular, a dishonest sender can disrupt the whole network.

In 1995, Sako and Kilian proposed a receipt-free voting scheme [28] by adding universal verifiability to the mixnet proposed by Park, Itoh, and Kurosawa in [27]. The universal verifiability is obtained by requiring each center to prove that they correctly processed their messages. The result can be verified by any interested party to confirm that the messages have been handled correctly. This design helps to thwart the security attack proposed in [80] without adding extra redundancy. Sako and Kilian's scheme is based on *chameleon blobs*, which is a zero-knowledge bit-commitment scheme [81] with an additional property that the verifier knows how to open a blob in two ways. In their scheme, each mix needs to commit to their inputs and outputs, and prove in zero-knowledge that they performed the decryption and shuffle correctly. Since this scheme is based on [27], the voters' work can be independent of the number of mixes. However, the verifier must verify that each server behaved correctly using the cut-and-choose strategy. In this way, the verifier's complexity to prove the correctness of a shuffle remains to be $\mathcal{O}(m \log \varepsilon)$ [82], or roughly $642n$ for n input data [39]. An attack was discovered by Michels and Horster in 1996 [83] that the scheme is not receipt-free if a sender collaborates with a mix. This is because the attack proposed by Pfitzmann [80] can still be applied. In fact, if one sender colludes with a mix, then they can even see together what Bob has sent.

In 1998, Abe [82] proposed a universally verifiable mixnet. In this scheme, the verification is independent of the number of mix-servers and the computation of each mix-server is constant against the number of mix-servers except for some negligible tasks like addition. In this scheme, the users first post their messages encrypted using ElGamal encryption with the public key of the mixes to the bulletin board. The mix system works in two phases. The first phase is randomization and permutation, and the second phase is threshold decryption. The verification complexity for this system is $\mathcal{O}(\log \varepsilon)$ [82], or $22n \log n$ [39]. In the subsequent year, Abe [84] proposed two universally verifiable mixnets for small to middle-scale secret-ballot electronic elections to improve the efficiency by eliminating the cut-and-choose protocol proposed in [64]. The construction is based on a permutation network composed of a network of switches that transposes two inputs. For n inputs and t tolerable corrupt mixes, the complexity for computation and communication is $\mathcal{O}(tn \log n)$. However, it was proved later on that the original construction yields biased permutation. This means the adversary can get advantage in violating anonymity that is more efficient than random guessing [29]. In this paper, an alternative solution was also proposed to achieve uniform distribution over all permutations.

In 2001, Furukawa and Soka proposed a new protocol for proving the correctness of a shuffle, without leaking how shuffle was performed [39]. This protocol can prove the correctness of a shuffle of n data with $18n + 18$ exponentiations [40]. The proposed protocol will be a building block of an efficient, universally verifiable mixnet, whose application to the voting system is prominent. The construction is based on a permutation by a matrix. Also in 2001, Neff presented a mathematical construction to verifiably shuffle a sequence of n modular integers [40]. This construction provides a linear size proof of correctness for the output sequence that can be checked by any verifiers. This protocol is shown to be honest-verifier zero-knowledge in a special case, and is computationally zero-knowledge in general. The number of exponentiations required to construct the proof is $8n + 5$.

The systems that provide universal verifiability based on proofs of permutations, and general zero-knowledge proofs are computationally very expensive. Some design tradeoffs have been proposed in the literature. Boneh and Golle [85] proposed a two-phase verification scheme in 2002. In the first phase, a design tradeoff between efficiency and correctness was presented for scenarios where a guarantee of almost entirely correct mixing may be sufficient for the result of a large election. This phase guarantees that the mixnet processes all inputs correctly with a high, but not overwhelming, probability. The technique consists of computing the product of a random subset of inputs to a mix server, then requires the mix server to produce a subset of outputs of equal product. The cost of this phase requires only a constant number of exponentiations independent of the number of inputs mixed. In the second

phase, the correctness of the result can be verified beyond a doubt using any of the existing proofs of perfect correctness, which is generally much slower. For this phase, even the fastest proofs of perfectly correct mixing require computation of the number of exponentiations linear in the number of inputs.

Another design tradeoff is the *exit-poll* mix designed by Golle et al. in 2002 [34]. The exit-poll mix is a general ciphertext-to-ciphertext scheme. In this scheme, the mixnet is efficient and produces an output only if no server cheats. When one or several mix servers cheat, the inputs are converted to a format and verified using any of the proposed mixnet. However, a series of attacks for this scheme have been identified [50]. The first two attacks [86,87] are closely related to [80] and can break the anonymity of any user. The second attack is related to [31] and can break the anonymity of all users and compromise the robustness. The last attack is based on improperly checking the ElGamal elements [88].

In 2002, Jakobsson et al. proposed a mixnet design based on randomized partial checking (RPC) [89]. In this scheme, each mix server provides a strong evidence of its correct operation by revealing a pseudo-randomly selected subset of its input/output relations. This design works with mixnets based on any encryption scheme and also with Chaum's original mixnet. This scheme provides voter privacy as a global property of the mixnet. It also provides high assurance of a correct election result, since a corrupt server is very likely to be caught if it attempts to tamper with even a couple of ballots.

In 2004, Golle and Juels proposed a parallel mixing design [90]. The proposed re-encryption mixnet allows servers to mix inputs in parallel so that the overall mixing time can be reduced for moderate-to-large numbers of servers. More precisely, for n inputs and M servers, the parallel re-encryption mixnet produces output in time at most $2n$, and only around n assuming a majority of honest servers. Though their techniques can drastically reduce the latency of mixing, Borisov [91] proves that when multiple input messages are known to the adversary, the anonymity provided by this technique is far from optimal.

Finally, we consider a universal re-encryption technique that permits universal re-encryption of the ciphertexts [92]. This scheme relies on a new public-key re-encryption primitive to eliminate the requirement of the public key. The mix servers hold no public or private keying material, and may therefore dispense with the cumbersome requirements of key generation, key distribution, and private-key management. Since then, many schemes have been developed based on the newly developed primitive [93–97]. Unfortunately, its mathematical properties were found to be vulnerable to tagging attacks by Danezis in 2006 [98].

9. Other systems

A number of other systems have also been proposed through the years. P5 was proposed by Sherwood et al. in 2002 for anonymous communications over the Internet [99]. P5 stands for Peer-to-Peer Personal Privacy Protocol. It uses broadcast-trees to achieve anonymity. P5 provides individual participants a tradeoff between degree of anonymity and communication efficiency, which can be used to scalably implement large anonymous groups.

P5 creates a broadcast hierarchy so that different levels of hierarchy provide different levels of anonymity, at the cost of communication bandwidth and reliability. Users of the systems locally select a level (mask) of anonymity and communication efficiency based on their expected performance. The P5 logical broadcast hierarchy is a binary tree constructed using the public keys of each user. Each node of the tree consists of a bitstring of a specific length to represent its hierarchy level (in horizontal) and also the group (in vertical). By applying a secure public hash function on the public key of each user, the user is being mapped to a node and a group. For a user A to send messages to user B, A could try to send messages on to some groups towards the root direction. If B receives the message, B replies back to A. However, in doing so, B sacrifices some anonymity since A can map B to a smaller set (branch) of users in the hierarchy. In P5, the user always has the flexibility to decrease their level of anonymity in order to increase their performance.

ISDN-mixes [100] was proposed in 1991 to provide communication untraceability for ISDN networks with small bandwidth overhead. ISDN-mixes is a combination of Chaum's mixes, dummy traffic on the subscriber lines, and broadcasts of incoming-call messages in the subscriber-area. It is based on mix-channels using cascades of mixes to make sure that each message is processed by all mixes in the same order. Each mix-channel consists of a mix-sending channel from the sender and a mix-receiving channel from the recipient. Each half of a mix-channel protects only the participant who has established it. Thus, to prevent both the sender and the recipient from being able to trace each other, the two halves are connected to generate a mix-channel through their respective channel labels and also the routing information to the mix cascade on the sending side.

Real-Time MIXes [101] was generalized from ISDN-mixes as a bandwidth-efficient anonymity protocol with real-time constraints. It was proposed by Jerichow et al. in 1998. Real-Time MIXes essentially presents protocols for narrow-band ISDN based on mixes. Similar to the ISDN-mixes, it is also based on mix-channels, which distinguish mix-sending channels that keep the sender anonymous and mix-receiving channels that keep the recipient anonymous. In the real-time mixes, each user is connected to a local exchange via an exclusive wire, whereas the bandwidth is shared in the long-distance network.

The route setup messages are separated from the actual data traveling in the network. The signaling channel is used to transmit the onion encoded message that contains the session keys for each intermediary mix.

In both schemes, each connection is divided into a sequence of time-slide channels, that look completely unrelated to everybody except for the respective sender and recipient. With each new time slice, users can release connections and/or establish new ones. Each participant who does not need a channel during a time slice establishes a dummy time-slice channel instead. This costs no additional bandwidth because this channel is on the subscriber line only.

Within each time slice, each subscriber maintains two mix-sending channels and two mix-receiving channels through the mixes at the local exchange. To set up a call, the sender needs to deliver a connection-setup message to the recipient so that dummy channels can be stopped. The recipient may hide his real location address from the sender by allowing implicit addresses and broadcasting short connection-setup messages in his anonymity set.

Hordes [102] is a multicast based protocol that provides initiator anonymity on the Internet. Hordes was proposed by Levine and Shields in 2002. It uses forward mechanisms similar to those used in previous protocols for sending data, but it is the first protocol that makes use of multicast routing to anonymously receive data.

Hordes employs multiple proxies similar to those used in the Crowds protocol to anonymously route a packet towards the responder. However, Hordes makes use of multicast communication for the reverse path of anonymous connections, which can provide additional initiator privacy protection. The asymmetry of the forward and reverse paths in Hordes poses a challenge for providing TCP service, since the forward path has a higher latency and an increased chance for packet loss as compared to the reverse path. Overall, it was shown that this design results in shorter transmis-

sion latencies and requires less work from protocol participants, in terms of messages processed while providing anonymity to a degree similar to that of Crowds and Onion Routing.

In addition, traffic analysis is a powerful tool in attacking anonymous communication systems. A robust anonymous communication system should be able to safeguard traffic analysis. The research in this area includes traffic flow confidentiality [103–105] and traffic flow analysis [106–109].

10. Conclusions

In this survey, we reviewed the major techniques in the field of anonymous communications. We focused on cryptographic primitives, mixnet-based systems, DC-net-based systems, network routing based systems, and peer-to-peer networks. We also briefly described the applications of verifiable mixnets and applications to electronic voting.

Though research in anonymous communications has been going on extensively for nearly three decades, it is still a very new and active research area. Design of practical, low-latency, and robust anonymous communication systems is of high practical importance. As portable wireless devices and wireless sensor networks continue to grow and become more common, scalability and efficiency are becoming the next impetus and the most challenging design issues in this area. We expect anonymous communications will have a much wider arena of applications in the near future.

Acknowledgements

The work of Jie Wu is supported in part by NSF Grants CNS 0422762, CNS 0626240, CCF 0839289, and CNS 0948184. The work is Jian Ren is supported in part by NSF Grants CNS 0845812, CNS 0848569, and CNS 0716039.

References

[1] D. Chaum, Untraceable electronic mail return addresses and digital pseudonyms, Communications of the ACM 24 (2) (1981) 84–88. February.
[2] A. Pfitzmann, M. Hansen. Anonymity, unlinkability, unobservability, pseudonymity, and identity management a proposal for terminology. <http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.31.pdf>, February 15 2008.
[3] R. Rivest, A. Shamir, Y. Tauman, How to leak a secret, in: Advances in Cryptology – ASIACRYPT, Lecture Notes in Computer Science, vol. 2248/2001, Springer, Berlin/Heidelberg, 2001.
[4] D. Chaum, E. van Heyst, Group signatures, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 547, 1991, pp. 257–265.
[5] L. Chen, T.P. Pedersen. New group signature schemes, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 950, 1995, pp. 171–181.
[6] L. Chen, T.P. Pedersen. On the efficiency of group signatures providing information-theoretic anonymity, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 921, 1995, pp. 39–49.
[7] J.L. Camenisch, Efficient and generalized group signatures, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 1233, 1997, pp. 465–479.
[8] J.L. Camenisch, M.A. Stadler, Efficient group signature schemes for large groups, in: Advances in Cryptology – Crypto'97, Lecture Notes in Computer Science, vol. 1294, 1997, pp. 410–424.
[9] J. Ren, L. Harn, Generalized ring signatures, IEEE Transaction on Dependable and Secure Computing 5 (3) (2008) 155–163. July–September.
[10] J. Herranz, G. Saez. Forking lemmas in the ring signatures' scenario. Technical Report 067, International Association for Cryptologic Research, 2003. <http://eprint.iacr.org/2003/067.ps>.
[11] C.P. Schnorr, Efficient identification and signatures for smart cards, in: Advances in Cryptology – Crypto'89, Lecture Notes in Computer Science, vol. 435, 1989, pp. 239–252.
[12] J. Xu, Z. Zhang, D. Feng, A ring signature scheme using bilinear pairings, in: Lecture Notes in Computer Science, vol. 3325/2005, 2005.
[13] A. Awasthi, S. Lal, A new proxy ring signature scheme, 2004. <http://arxiv.org/PS_cache/cs/pdf/0410/0410010v1.pdf>.
[14] C. Lin, T. Wu, An identity-based ring signature scheme from bilinear pairings, in: AINA, 2004, pp. 182 – 185.
[15] W. Cheng, W. Lang, Z. Yang, G. Liu, Y. Tan, An identity-based proxy ring signature scheme from bilinear pairings, in: Ninth International Symposium on Computers and Communications (ISCC), vol. 1, 28 June–1 July 2004, pp. 424–429.
[16] C. Gamage, B. Gras, B. Crispo, A. Tanenbaum, An identity-based ring signature scheme with enhanced privacy, in: Securecomm and Workshops, August 28–September 1 2006, pp. 1 – 5.
[17] C. Hu, D. Li, Forward-secure traceable ring signature, in: SNPD, July 30–August 1 2007, pp. 200–204.
[18] J. Li, T. Yuen, X. Chen, Y. Wang, Proxy ring signature: formal definitions, efficient construction and new variant, in: International Conference on Computational Intelligence and Security, vol. 2, 3–6 November 2006, pp. 1259–1264.
[19] C. Hu, D. Li. A new type of proxy ring signature scheme with revocable anonymity, in: SNPD, vol. 1, July 30–August 1 2007, pp. 866–868.
[20] C. Zhang, Y. Liu, D. He, A new verifiable ring signature scheme based on nyberg-rueppel scheme, in: The 8th International Conference on Signal Processing, 2006.
[21] D. Chaum, The dinning cryptographer problem: unconditional sender and recipient untraceability, Journal of Cryptology 1 (1) (1988) 65–75.
[22] D. Goldschlag, M. Reed, P. Syverson, Hiding routing information, Workshop on Information Hiding, 1465, LNCS, Cambridge, UK, 1996, p. 214C226.
[23] B. Pfitzmann, A. Pfitzmann, How to break the direct RSA-implementation of MIXes, Advances in Cryptology - EUROCRYPT, vol. 434, Lecture Notes in Computer Science, 1989, pp. 373–381.
[24] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the Association of Comp. Mach. 21 (2) (1978) 120–126.
[25] C. Gülcü, G. Tsudik, Mixing email with babel, in: Proceedings of the Symposium on Network and Distributed System Security, San Diego, CA, 1996.
[26] T.A. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory 31 (4) (1985) 469–472.
[27] C. Park, K. Itoh, K. Kurosawa, Efficient anonymous channel and all/nothing election scheme, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 765, 1993, pp. 248–259.
[28] K. Sako, J. Kilian, Receipt-free mix-type voting scheme a practical solution to the implementation of a voting booth, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 921, 1995, pp. 393–403.
[29] M. Abe, F. Hoshino, Remarks on mix-network based on permutation network, PKC 2001, 1992 of Lecture Notes in Computer Science, Springer-Verlag, 2001, pp. 317–324.
[30] M. Jakobsson, A practical mix, in: Advances in Cryptology – EUROCRYPT, 1403 of Lecture Notes in Computer Science, 1998, pp. 448–461.
[31] Y. Desmedt, K. Kurosawa, How to break a practical mix and design a new one, in: Advances in Cryptology – EUROCRYPT, 1807 of Lecture Notes in Computer Science, 2000, pp. 557–572.
[32] A. Juels, M. Jakobsson, An optimally robust hybrid mix network, in: Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computation, 2001.
[33] M. Abe, H. Imai, Flaws in some robust optimistic mix-nets, Proceedings of the 2003 ACISP, Springer-Verlag, 2003, pp. 39–50.
[34] P. Golle, S. Zhang, D. Boneh, M. Jakobsson, A. Juels. Optimistic mixing for exit-pools, in: Advances in Cryptology – ASIACRYPT, 2501 of Lecture Notes in Computer Science, 2002, pp. 451–465.
[35] D. Wikström, Four practical attacks for "optimistic mixing for exit-polls", Technical Report SICS-T-2003/04-SE, Swedish Institute of Computer Science (SICS), December February 2003.
[36] M. Jakobsson, Flash mixing, in: Principles of Distributed Computing (PODC), 1999. <http://citeseer.nj.nec.com/jakobsson99flash.html>.
[37] M. Jakobsson, A. Juels, an optimally robust hybrid mix network, in: Principles of Distributed Computing (PODC), 2001. <http://citeseer.nj.nec.com/492015.html>.
[38] M. Ohkubo, M. Abe, A length-invariant hubrid mix, in: Advances in Cryptology – ASIACRYPT, 2000.
[39] J. Furukawa, K. Sako, An efficient scheme for proving a shuffle, in: Advances in Cryptology – CRYPTO, 2139 of Lecture Notes in Computer Science, 2001, pp. 368–387.
[40] A. Neff, A verifiable secret shuffle and its application to e-voting, in: ACM CCS, 2001, pp. 116–125.
[41] J. Camenisch, A. Mityagin, Mix-network with stronger security, in: PET, LNCS 3856, 2005, pp. 128–146.
[42] J. Helsingius, Anon.penet.fi press release, <http://www.penet.fi/press-english.html>.
[43] S. Parekh, Prospects for remailers. First Monday, 1(2), August 1996, <http://www.firstmonday.dk/issues/issue2/remailers>/.
[44] L. Cottrell, Mixmaster and remailer attacks, <http://www.obscura.com/loki/remailer/remailer-essay.html>.
[45] U. Möller, L. Cottrell, P. Palfrader, L. Sassaman, Mixmaster protocol, July 2003, Version 2.
[46] G. Danezis, R. Dingledine, N. Mathewson, Mixminion: design of a type III anonymous remailer protocol, in: IEEE Symposium on Security and Privacy, 2003, pp. 2–15.
[47] J. Postel, Simple mail transfer protocol, April 2001, <http://www.ietf.org/rfc/rfc2821.txt>.

[48] T. Dierks, C. Allen, RFC 2246: The TLS Protocol, January 1999, <http://www.ietf.org/rfc/rfc2246.txt>.
[49] W. Diffle, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22 (6) (1996) 644–654.
[50] G. Danezis, C. Diaz, A survey of anonymous communication channels, Technical Report MSR-TR-2008-35, Microsoft Research, January 2008.
[51] M. Reed, P. Syverson, D. Goldschlag, Anonymous connections and onion routing, IEEE Journal on Selected Areas in Communications 16 (4) (1998) 482–494.
[52] D. Goldschlag, M. Reed, P. Syverson, Onion routing for anonymous and private Internet connections, Communications of the ACM 42 (2) (1999) 39–41.
[53] R. Dingledine, N. Mathewson, P. Syverson. Tor: The second-generation onion router, in: Proceedings of the 13th USENIX Security Symposium, August 2004.
[54] R. Dingledine, N. Mathewson, P. Syverson, Challenges in deploying low-latency anonymity, Technical Report 5540-625, NRL CHACS, 2005.
[55] The Anonymizer, <http://anonymizer.com>/.
[56] O. Berthold, H. Federrath, S. Köpsell, Web MIXes: a system for anonymous and unobservable Internet access, Lecture Notes in Computer Science (2001) 115–129.
[57] D. Martin, A. Schulman, Deanonymizing users of the safeweb anonymizing service, in: Proceedings of the 11th USENIX Security Symposium, San Francisco, August 2002.
[58] Z. Brown, Cebolla: Pragmatic IP anonymity, <http://www.linuxinsight.com/files/ols2002/brown-reprint.pdf>.
[59] N. Feamster, R. Dingledine, Location diversity in anonymity networks, in: Proceedings of the Workshop on Privacy in the Electronic Society (WPES), Washington, DC, USA, 2004.
[60] S.J. Murdoch, G. Danezis, Low-cost traffic analysis of Tor, in: Proceedings of the 2005 IEEE Symposium on Security and Privacy, IEEE CS, 2005.
[61] L. Øverlier, P. Sverson, Locating hidden servers, in: Proceedings of the 2006 IEEE Symposium on Security and Privacy, IEEE CS, 2006.
[62] J. Feigenbaum, A. Johnson, P. Syverson, A model of onion routing with provable anonymity, in: 11th International Conference on Financial Cryptography and Data Security FC 2007, 2007.
[63] P. Golle, A. Juels, Dining cryptographers revisited, in Advances in Cryptology – Eurocrypt 2004, LNCS 3027, 2004, pp. 456–473.
[64] L. von Ahn, A. Bortz, N. Hopper, k-anonymous message transmission, in: Proceedings of CCS, Washington DC, USA, 2003, pp. 122–130.
[65] J.N.E. Bos, B. den Boer, Detection of disrupters in the DC protocol, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 434, 1989, pp. 320–328.
[66] M. Waidner, B. Pfitzmann, The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure serviceability, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 434, 1989, pp. 690–690.
[67] M. Waidner, Unconditional sender and recipient untraceability in spite of active attacks, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 434, 1989, pp. 302–319.
[68] S. Goel, M. Robson, M. Polte, E. Sirer, Herbivore: A Scalable and Efficient Protocol for Anonymous Communication, Technical Report 2003-1890, Cornell University, Ithaca, NY, February 2003.
[69] S. Dolev, R. Ostrovsky, Efficient anonymous multicast and reception, in: Advances in Cryptology – Crypto'97, Lecture Notes in Computer Science, vol. 1294, 1997, pp. 395–409.
[70] S. Dolev, R. Ostrovsky, Xor-trees for efficient anonymous multicast and reception, ACM Transactions on Information and System Security 3 (2) (2000) 63–84. May.
[71] M. Reiter, A. Rubin, Crowds: anonymity for web transaction, ACM Transactions on Information and System Security 1 (1) (1998) 66–92.
[72] M.J. Freedman, R. Morris, Tarzan: a peer-to-peer anonymizing network layer, CCS'02: Proceedings of the 9th ACM Conference on Computer and Communications Security, ACM, New York, NY, USA, 2002, pp. 193–206.
[73] M. Wright, M. Adler, B. Levine, C. Shields, The predecessor attack: an analysis of a threat to anonymous communications systems, ACM Transactions on Information and System Security 7 (2004) 2004.
[74] V. Shmatikov, Probabilistic analysis of anonymity, in: IEEE Computer Security Foundations Workshop (CSFW), 2002, pp. 119–128.
[75] A. Beimel, S. Dolev, Buses for anonymous message delivery, Journal of Cryptology 16 (2003) 25–39.
[76] Y. Guan, X. Fu, R. Bettati, W. Zhao, An optimal strategy for anonymous communication protocols, in: IEEE ICDCS, 2002.
[77] M.J. Freedman, J. Sit, J. Cates, R. Morris, Introducing tarzan: a peer-to-peer anonymizing network layer, in: International Workshop on Peer-to-Peer Sysems (IPTPS), vol. 2429, LNCS, 2002, pp. 121–129.
[78] G. Danezis, R. Clayton, Route fingerprinting in anonymous communications, in: IEEE International Conference on Peer-to-Peer Computing (PTP), 2006.
[79] P. Tabris, N. Borisov, Breaking the collusion detection mechanism of MorphMix, in: Privacy Enhancing Technologies Workshop (PET), LNCS, 2006.
[80] B. Pfitzmann, Breaking an efficient anonymous channel, in: Advances in Cryptology – EUROCRYPT, Lecture Notes in Computer Science, vol. 950, 1995, pp. 332–340.
[81] G. Brassard, D. Chaum, C. Crépeau, Minimum disclosure proofs of knowledge, JCSS, 1988, pp. 156–189.
[82] M. Abe, Universally verifiable mix with verification work independent of the number of mix servers, in: Advances in Cryptology – EUROCRYPT, 1998, pp. 437–447.
[83] M. Michels, P. Horster, Some remarks on a receipt-free and universally verifiable mix-type voting scheme, in: Advances in Cryptology –ASIACRYPT, 1996, pp. 125–132.
[84] M. Abe, Mix-networks on permutation networks, in: Advances in Cryptology – ASIACRYPT, 1999, pp. 258–273.
[85] D. Boneh, P. Golle, Almost entirely correct mixing with applications to voting, in: ACM Conference on Computer and Communications Security, 2002, pp. 68–77.
[86] D. Wikström, How to break, fix, and optimize "optimistic mix for exitpolls", Technical Report T2002-24, Swedish Institute of Computer Science, SICS, Box 1263, SE-164 29 Kista, SWEDEN, 2002.
[87] D. Wikström, Four practical attacks for "optimistic mixing for exitpolls", Technical Report T2003-04, Swedish Institute of Computer Science, SICS, Box 1263, SE-164 29 Kista, SWEDEN, 2003.
[88] D. Wikström. Elements in $\mathbb{Z}_p^* \setminus G_q$ are dangerous, Technical Report T2003-05, Swedish Institute of Computer Science, SICS, Box 1263, SE-164 29 Kista, SWEDEN, 2003.
[89] M. Jakobsson, A. Juels, R. Rivest, Making mix nets robust for electronic voting by partial checking, in: Proc. USENIX Security, 2002, pp. 339–353.
[90] P. Golle, A. Juels, Parallel mixing, in: Proceedings of the 11th ACM Conference on Computer and Communications Security, Washingto DC, USA, 2004.
[91] N. Borisov, An analysis of parallel mixing with attacker-controlled inputs, in: Proceedings of Privacy Enhancing Technologies workshop (PET), 2005.
[92] P. Golle, M. Jakobsson, A. Juels, P. Syverson, Universal reencryption for mixnets, in: Proceedings of the 2004 RSA Conference, Cryptographer's track', 2004.
[93] M. Gomulkiewicz, M. Klonowski, M. Kutylowski, Anonymous communication with on-line and off-line onion encoding, in: Proceedings of Workshop on Information Security Applications (WISA 2004), 2004.
[94] M. Klonowski, M. Kutylowski, A. Lauks, F. Zagorski, Universal re-encryption of signatures and controlling anonymous information flow, in: WARTACRYPT'04 Conference on Cryptology, Bedlewo/Poznan, 2004.
[95] M. Gomulkiewicz, M. Klonowski, M. Kutylowski, Onions based on universal re-encryption – anonymous communication immune against repetitive attack, in: Proceedings of Workshop on Information Security Applications (WISA 2004), 2004.
[96] T. Lu, B. Fang, Y. Sun, X. Cheng, Performance analysis of wongoo system, in: CIT, IEEE Computer Society, 2005, pages 716–723.
[97] T. Lu, B. Fang, Y. Sun, L. Guo, Some remarks on universal re-encryption and a novel practical anonymous tunnel, in: CIT, IEEE Computer Society, Lecture Notes in Computer Science, vol. 3619, Springer, 2005, pp. 716–723.
[98] G. Danezis. Breaking four mix-related schemes based on universal re-encryption, in: Proceedings of Information Security Conference, 2006.
[99] R. Sherwood, B. Bhattacharjee, A Srinivasan, P5: a protocol for scalable anonymous communication, in: Proceedings of the 2002 IEEE Symposium on Security and Privacy, 2002.
[100] A. Pfitzmann, B. Pfitzmann, M. Waidner, ISDN-mixes: untraceable communication with very small bandwidth overhead, in: Proceedings of the GI/ITG Conference on Communication in Distributed Systems, February 1991, pp. 451–463.
[101] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, M. Waidner, Real-time MIXes: a bandwidth-efficient anonymity protocol, IEEE Journal on Selected Areas in Communications 16 (4) (1998).
[102] B. Levine, C. Shields, Hordes: a multicast based protocol for anonymity, Journal of Computer Security 10 (3) (2002) 213–240.
[103] B. Timmerman, A security model for dynamic adaptive traffic masking, in: New Security Paradigms Workshop, Langdale, Cumbria, UK, 1997, pp. 107–115.
[104] B. Timmerman, Secure dynamic adaptive traffic masking, in: New Security Paradigms Workshop, Ontario, Canada, 1999, pp. 13–24.
[105] A. Back, U. Möller, A. Stiglic, Traffic analysis attacks and trade-offs in anonymity providing systems, in: Proceedings of the 4th International Workshop on Information Hiding, Lecture Notes in Computer Science, vol. 37, Springer, Berlin/Heidelberg, 2001, pp. 245–257.
[106] R.E. Newman, I.S. Moskowitz, P. Syverson, A. Serjantov, Metrics for traffic analysis prevention, in: Privacy Enhancing Technologies Workshop, Dresden, Gernamy, 2003.
[107] B. Venkatraman, R. Newman-Wolfe, Performance analysis of a method for high level prevention of traffic analysis using measurements from a campus network, Proceeding of the IEEE/ACM Tenth Annual Computer Security Applications Conference, IEEE CS Press, Orlando, FL, 1994, pp. 288–297.
[108] Y. Guan, X. Fu, R. Bettati, W. Zhao, A quantitative analysis of anonymous communications, in: IEEE INFOCOM, 2002.
[109] Y. Guan, X. Fu, R. Bettati, W. Zhao, A quantitative analysis of anonymous communications, IEEE Transactions on Reliability 53 (1) (2004) 103–115. March.