BackRef: Accountability in Anonymous Communication Networks

Michael Backes[1,3], Jeremy Clark[2], Aniket Kate[1], Milivoj Simeonovski[1], and Peter Druschel[3]

[1] Saarland University, Germany
[2] Concordia University, Canada
[3] Max Planck Institute for Software Systems (MPI-SWS), Germany

Abstract. Many anonymous communication networks (ACNs) rely on routing traffic through a sequence of proxy nodes to obfuscate the originator of the traffic. Without an accountability mechanism, exit proxy nodes may become embroiled in a criminal investigation if originators commit criminal actions through the ACN. We present BackRef, a generic mechanism for ACNs that provides practical repudiation for the proxy nodes by tracing back the selected outbound traffic to the predecessor node (but not in the forward direction) through a cryptographically verifiable chain. It also provides an option for full (or partial) traceability back to the entry node or even to the corresponding originator when all intermediate nodes are cooperating. Moreover, to maintain a good balance between anonymity and accountability, the protocol incorporates whitelist directories at exit proxy nodes. BackRef offers improved deployability over the related work, and introduces a novel concept of pseudonymous signatures that may be of independent interest.

We exemplify the utility of BackRef by integrating it into the onion routing (OR) protocol, and examine its deployability by considering several system-level aspects. We also present the security definitions for the BackRef system (namely, anonymity, backward traceability, no forward traceability, and no false accusation) and conduct a formal security analysis of the OR protocol with BackRef using ProVerif, an automated cryptographic protocol verifier, establishing the aforementioned security properties against a strong adversarial model.

Keywords: anonymity, malicious users, accountability, repudiation, traceability, formal verification.

1 Introduction

Anonymous communication networks (ACNs) are designed to hide the originator of each message within a larger set of users. In some systems, like DC-Nets [1] and Dissent [2], the message emerges from aggregating all participants' messages. In other systems, like onion routing [3], mix networks [4], and peer-to-peer anonymous communication networks [5], messages are routed through volunteer nodes

I. Boureanu, P. Owesarski, and S. Vaudenay (Eds.): ACNS 2014, LNCS 8479, pp. 380–400, 2014.

that act as privacy-preserving proxies for the users' messages. We call this latter class proxy-based ACNs and concentrate on it henceforth.

Proxy-based ACNs provide a powerful service to their users, and correspondingly they have been the most successful ACNs so far [6,7]. However the nature of the properties of the technology can sometimes be harmful for the nodes serving as proxies. If a network user's online communication results in a criminal investigation or a cause of action, the last entity to forward the traffic may become embroiled in the proceedings [8,9], whether as the suspect/defendant or as a third party with evidence. While repudiation in the form of a partial or full traceability has never been a component of any widely-deployed ACN, it may become the case that new anonymity networks, or a changing political climate, initiate an interest in providing a verifiable trace to users who misuse anonymity networks according to laws or terms of service.

While several proposals [10,11,12,13,14,15,16] have been made to tackle or at least to mitigate this problem under the umbrella term of *accountable anonymity*, as we discuss in the next section some of them are broken, while others are not scalable enough for deploying in low latency ACNs.

Our Contributions. In this work, we design BackRef, a novel practical repudiation mechanism for anonymous communication, which has advantages in terms of deployability and efficiency over the literature. To assist in the design of BackRef, we propose a concept of pseudonymous signatures (§3), which employ pseudonyms (or half Diffie-Hellman exponents) as temporary public keys (and corresponding temporary secrets) employed or employable in almost all ACNs for signing messages. These pseudonym signatures are used to create a verifiable *pseudonym-linkability* mechanism where any proxy node within the route or path, *when required*, can verifiably reveal its predecessor node in time-bound manner. We use this property to design a novel repudiation mechanism (§4), which allows each proxy node, in cooperation with the network, to issue a cryptographic guarantee that a selected traffic flow can be traced back to its originator (*i.e.,* predecessor node) while maintaining the eventual forward secrecy of the system. Unlike the related work, which largely relies on group signatures and/or anonymous credentials, BackRef avoids the logistical difficulties of organizing users into groups and arranging a shared group key, and does not require access to a trusted party to issue credentials. While BackRef is applicable to all proxy-based ACNs, we illustrate its utility by applying it to the onion routing (OR) protocol. We observe that it introduces a small computational overhead and does not affect the performance of the underlying OR protocol (§5). BackRef also includes a *whitelisting* option; *i.e.,* if an exit node considers traceability to one or more web-services unnecessary, then it can include those services in a *whitelist* directory such that accesses to those are not logged.

We formally define the important security properties of the BackRef network (§6). In particular, we formalize anonymity and no forward traceability as observational equivalence relations, and backward traceability and no false accusation as trace properties. We conduct a formal security analysis of BackRef using ProVerif, an automated cryptographic protocol verifier, establishing the aforementioned security and privacy properties against a strong adversarial model. We believe both the definitions and the security analysis are of independent interest, since they are the first for the OR protocol.

2 Background and Related Work

Anonymous communication networks (ACNs) aim at protecting personally identifiable information (PII), in particular the network addresses of the communicating parties by hiding correlation between input and output messages at one or more network entities. For this purpose, the ACN protocols employ techniques such as using a series of intermediate routers and layered encryptions to obfuscate the source of a communication, and adding fake traffic to make the 'real' communication difficult to extract.

Anonymous Communication Protocols. Single-hop proxy servers, which relay traffic flows, enable a simple form of anonymous communication. However anonymity in this case requires, at a minimum, that the proxy is trustworthy and not compromised, and this approach does not protect the anonymity of senders if the adversary inspects traffic through the proxy [17]. Even with the use of encryption between the sender and proxy server, timing attacks can be used to correlate flows.

Starting with Chaum [4], several ACN technologies have been developed in the last thirty years to provide stronger anonymity not dependent on a single entity [6,3,7,2,1,18,19,20,21]. Among these, mix networks [4,7] and onion routing [6] have arguably been most successful. Both offer user anonymity, relationship anonymity and unlinkability [22], but they obtain these properties through differing assumptions and techniques.

An onion routing (OR) infrastructure involves a set of *routers* (or *OR nodes*) that relay traffic, a *directory service* providing status information for OR nodes, and *users*. Users benefit from anonymous access by constructing a *circuit*—a small ordered subset of OR nodes—and routing traffic through it sequentially. The crucial property for anonymity is that an OR node within the built circuit is not able to identify any portion of the circuit other than its predecessor and successor. The user sends messages (to the first OR node in the circuit) in a form of an *onion*—a data structure multiply encrypted by symmetric session keys (one encryption layer per node in the circuit). The symmetric keys are negotiated during an initial *circuit construction* phase. This is followed by a second phase of *low latency* communication (opening and closing streams) through the constructed circuit for the session duration. An OR network does not aim at providing anonymity and unlinkability against a global passive observer, which in theory can analyze end-to-end traffic flow. Instead, it assumes an adversary that adaptively compromises a small fraction of OR nodes and controls a small fraction of the network.

A mix network achieves anonymity by relaying messages through a path of mix nodes (or mixes) in a latency-tolerant manner. The user encrypts a message to be partially decrypted by each mix along the path. Mixes accept a batch of encrypted messages, which are partially decrypted, randomly permuted, and forwarded. Unlike onion routing, an observer is unable to correlate incoming and outgoing messages at the mix; thus, mix networks provide anonymity against a powerful global passive adversary. In fact, as long as a single mix in the user's path remains uncompromised, the message will maintain some anonymity.

Accountable Anonymity Mechanisms. The literature has examined several approaches for adding accountability to ACN technologies, allowing misbehaving users to be selectively traced [10,11,12], exit nodes to deny originating traffic it forwards [13,14], misbehaving users to be banned [15,16], and misbehaving participants to be discovered [2,23,24]. All of these approaches either require users to obtain credentials or do not extend to interactive, low-latency, internet-scale ACNs. A number also partition users into subgroups, which reduces anonymity and requires a group manager. BackRef does not require credentials, subgroups, and is compatible with low-latency ACNs like onion routing, adding minimal overhead.

Kopsell *et al.* [10] propose traceability through threshold group signatures. A user logs into the system to join a group, signs messages with a group signature, and a group manager is empowered to revoke anonymity. The system also introduces an external proxy to inspect all outbound traffic for correct signatures and protocol compliance. The inspector has been criticized for centralizing traffic flows, which enables DOS, censorship, and increases observability [25].

Von Ahn *et al.* [11] also use group signatures as the basis for a general transformation for traceability in ACNs and illustrate it with DC networks. Users are required to register as members of a group capable of sending messages through the network. Our solution can be viewed as a follow-up to this paper, with a concentration on deployability: we do not require users to be organized into groups or introduce new entities, and we concentrate on onion routing.

Diaz and Preneel [12] achieve traceability through issuing anonymous credentials to users and utilizing a traitor tracing scheme to revoke anonymity. It is tailored to high-latency mix networks and requires a trusted authority to issue credentials—both impede deployability. Danezis and Sassaman [25] demonstrate a bypass attack on this and the Kopsell *et al.* scheme [10]. The attack is based on the protocols' assumption that there can be no leakage of information from inside the channel to the world unless it passes through the verification step. Our protocol does not rely on such a strong assumption, namely any exit node (or any node who leaks the information) with enabled BackRef can always activate the repudiation mechanism and shift liability to its predecessor node.

Short of revoking the anonymity of misbehaving users, techniques have been proposed to at least allow exit nodes to deny originating the traffic. Golle [13] and Clark *et al.* [14] pursue this goal, with the former being specific to high-latency mix networks and the latter requiring anonymous credentials. Tor offers a service called ExoneraTor [26] that provides a record of which nodes were online at a given time, but it does not explicitly prove that a given traffic flow originated from Tor. Other techniques, such as Nymble [15] and its successors (see a survey [16]), enable users to be banned. However these systems inherently require some form of credential or pseudonym infrastructure for the users, and also require web-servers to verify user requests. Finally, Dissent [2] and its successors [23,24] presents an interesting approach for accountable anonymous communication for DC Nets [1], however even when highly optimized [23], DC Nets are not competitive for internet-scale application.

3 Design Overview

In this section we describe our threat model and system goals, and present our key idea and design rationale.

3.1 Threat Model and System Goals

We consider the same threat model as the underlying ACN in which we wish to incorporate the BackRef mechanism. Our active adversary $\mathcal{A}$ aims at breaking some anonymity property by determining the ultimate source and/or destination of a communication stream or breaking unlinkability by linking two communication streams of the same user. We assume that some, but not all, of the nodes in the path of the communication stream are compromised by the adversary $\mathcal{A}$, who knows all their secret values, and is able to fully control their functionalities. For high latency ACNs like mix networks, we assume that the adversary can also observe all traffic in the network, as well as intercept and inject arbitrary messages, while for low latency ACNs like onion routing, we assume the adversary can observe, intercept, and inject traffic in some parts of the network.

While maintaining the anonymity and unlinkability properties of an ACN, we wish to achieve the following goals when incorporating BackRef in the ACN:

Repudiation: For a communication stream flowing through a node, the node operator should be able to prove that the stream is coming from another predecessor node or user.

Backward traceability: Starting from an exit node of a path (or circuit), it should be possible to trace the source of a communication stream back to the entry node when all nodes in the path verifiably reveal their predecessors.

No forward traceability: For a compromised node, it should not be possible for the adversary $\mathcal{A}$ to use BackRef to verifiably trace its successor in any completed anonymous communication session through it.

No false accusation: It should not be possible for a compromised node to corrupt the BackRef mechanism to trace a communication stream:
1. to a path different from the path employed for the stream, and
2. to a node other than its predecessor in the path.

Non-goals. We expect our accountability notion to be reactive in nature. We do not aim at proactive accountability and do not try to stop an illegal activity in an ACN in a proactive manner, as we believe perfect black-listing of web urls and content to be an infeasible task. Moreover, some nodes may choose not to follow the BackRef mechanism locally (e.g., they may not maintain

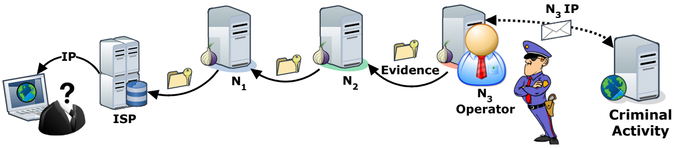


Fig. 1. Backward Traceability Verification

or share the required evidence logs), and full backward traceability cannot be ensured in those situations; nevertheless, the cooperating nodes can still prove their innocence in a verifiable manner.

Due to its reactive nature, our repudiation mechanism inherently requires evidence logs containing verifiable routing information. Encrypting these logs and regularly rotating the corresponding keys can provide us eventual forward secrecy [27]. However, we cannot aim for *immediate* forward secrecy due to the inherently *eventual* forward secret nature of the encryption mechanism.

3.2 Design Rationale and Key Idea

Fig. 1 presents a general expected architecture to achieve the above mentioned goals. It is clear the network level logs and the currently cryptographic mechanism in the ACNs cannot be used for verifiably backward traceability purpose as they cannot stop false accusations (or traceability) by compromised nodes: a compromised node can tamper with its logs to intermix two different ACN paths as there is no cryptographic association between different parts of an ACN path.

We observe that almost all OR protocols [19,27,28,29,30,31] (except TAP) and mix network protocols [32,33,34,20,7,21] employ (or can employ[1]) an element of a cyclic group of prime order satisfying some (version of) Diffie-Hellman assumption as an authentication challenges or randomization element per node in the path. In particular, it can be represented as $X = g^x$, where g is a generator of a cyclic group $\mathbb{G}$ of prime order p with the security parameter κ and $x \in_R \mathbb{Z}_p$ is a random secret value known only to the user. This element is used by each node on the path to derive a secret that is shared with the user and is used to extract a set of (session) keys for encryption and integrity protection. In the anonymity literature, these authentication challenges X are known as user *pseudonyms*.

The key idea of our BackRef mechanism is to use these pseudonyms $X = g^x$ and the corresponding secret keys x as signing key pairs to sign pseudonym's for successor nodes at entry and middle nodes, and to sign the communication stream headers at the exit nodes. Signatures that use (x, g^x) as the signing key pair are referred to as *pseudonym signatures*. As pseudonyms are generated independently for every single node, and the corresponding secret exponents are random elements of $\mathbb{Z}_p$, they do not reveal the user's identity. Moreover, it also is not possible to link two or more pseudonyms to a single identity. Therefore,

[1] Although some these have been defined using RSA encryptions, as discussed in [20] they can be modified to work in the discrete logarithm (DL) setting.

pseudonym signatures become particularly useful in our BackRef mechanism, where users utilize them to sign messages without being identified by the verifier.

We can employ a CMA-secure [35] signature scheme against a computationally bounded adversary (with the security parameter κ) such that, along with the usual existential unforgeability, the resultant pseudonym signature scheme satisfies the following property:

Unconditional Signer Anonymity: The adversary cannot determine a signer's identity, even if it is allowed to obtain signatures on an unbounded number of messages of its choice.

We use such temporary signing key pairs (or pseudonym signatures) to sign consecutively employed pseudonyms in an ACN path and the web communication requests leaving the ACN path. Pseudonym signatures provide linkability between the employed pseudonyms and the communicated message on an ACN path. However, these pseudonyms are not sufficient to link the node employed in the ACN path: for a pseudonym received by a node, its predecessor node can always deny sending the pseudonym in the first place. We solve this problem by introducing *endorsement signatures*: We assume that every node signs the pseudonym while sending it to the successor so that it cannot plausibly deny this transfer during backward tracing.

3.3 Scope of Solution

To understand the scope of BackRef, first consider traceability in the context of the simplest ACN: a single-hop proxy. Any traceability mechanism from the literature implicitly assumes a solution to the problem of how users can be traced through a simple proxy. We dub this the 'last mile' problem. The proxy can keep logs, but this requires a trusted proxy. Alternatively the ISP could observe and log relevant details about traffic to the proxy, requiring trust in the ISP. The solution more typically used in the literature is to assume individual users have digital credentials or signing keys—essentially some form of PKI is in place to certify the keys of individual users. [10,11,12,13,14]

None of these last mile solutions are particularly attractive. The assumption of a PKI provides the best distribution of trust but short-term deployment appears infeasible. We believe the involvement of ISPs is the most readily deployable. Such a solution involves an ISP with a packet attestation mechanism [36] which acts as a trusted party capable of proving the existence of a particular communication. We discuss the packet attestation mechanism further in §5.

For selected traffic flows, BackRef provides traceability to the entrance node. This is effectively equivalent to reducing the strong anonymity of a distributed cryptographic ACN to the weak anonymity of a single hop proxy. For full traceability, we then must address the 'last mile' problem: tracing the flow back to the individual sender. Thus BackRef is not a full traceability mechanism, but rather an essential component that can be composed with any *systems* solution to the last mile problem. While we later discuss a solution that involves ISPs, we emphasize that BACKREF itself is concentrated on, arguably, the more difficult problem of offering ensured traceability within the ACN.

4 Repudiation (or Traceability)

In this section, we present our BACKREF repudiation scheme. For ease of exposition, we include our scheme in an OR protocol instead of including it in the generic ACN protocol. Nevertheless, our scheme is applicable to almost all ACNs mentioned in §3.2. We start our discussion with a brief overview of the OR protocol in the Tor notions [37]. We then discuss the protocol flow for BACKREF and describe our cryptographic components.

4.1 The OR Protocol: Overview

The OR protocol is defined in two phases: circuit construction and streams relay.

OR Circuit Construction. The circuit construction phase involves the user onion proxy (OP) randomly selecting a short circuit of (*e.g.,* 3) OR nodes, and negotiating a session key with each selected OR node using one-way authenticated key exchange (1W-AKE) [31] such as the ntor protocol. When a user wants to create a circuit with an OR node N_1, she runs the *Initiate* procedure of the ntor protocol to generate and send an authentication challenge to N_1. Node N_1 then runs the respond procedure and returns the authentication response. Finally, the user uses the *ComputeKey* procedure of ntor along with the response to authenticate N_1 and to compute a session key with it. To extend the circuit further, the user sends an extend request to N_1 specifying the address of the next node N_2 and a new ntor authentication challenge for N_2. The process continues until the user exchanges the key with the exit node N_3.

Relaying Streams. Once a circuit (denoted as $\langle U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3 \rangle$) has been constructed through N_1, N_2 and N_3, the user-client U routes traffic through the circuit using onion-wrapping *WrOn* and onion-unwrapping *UnwrOn* procedures. *WrOn* creates a layered encryption of a payload (plaintext or onion) given an ordered list of (three) session keys. *UnwrOn* removes one or more layers of encryptions from an onion to output a plaintext or an onion given an input onion and a ordered list of one or more session keys. To reduce latency, many of the user's communication streams employ the same circuit [6].

The structure and components of communication streams may vary with the network protocol. For ease of exposition, we assume the OR network uses TCP-based communication in the same way as Tor, but our schemes can easily be adapted for other types of communication streams.

In Tor, the communication between the user's TCP-based application and her Tor proxy takes place via SOCKS. To open a communication stream (i.e., to start a TCP connection to some web server and port), the user proxy sends a *relay begin* cell (or packet) over the circuit to the exit node N_3. When N_3

receives the TCP request, it makes a standard TCP handshake with the web server. Once the connection is established, N_3 responds to the user with a *relay connected* cell. The user then forwards all TCP stream requests for the server as *relay data* cells to the circuit. (See [6,37] for a detailed explanation.)

4.2 The BackRef Protocol Flow

Consider a user U who wishes to construct an OR circuit $\langle U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3 \rangle$, and use it to send communication stream m. BackRef adds the repudiation mechanism as a layer on the top of the existing OR protocol. We assume that every OR node possesses a signing (private) key for which the corresponding verification (public) key is publicly available through the OR directory service.

The corresponding OR protocol with the BackRef scheme works according to the following five steps:

1. Circuit Construction with an Entry Node: The user U creates a circuit with the entry node N_1 using the ntor protocol. If the user is an OR node, then it endorses its pseudonym X_1 by signing it with its public key and sending the signature along with X_1.

However, if the user U is not an OR node, it cannot endorse the pseudonym X_1 as no public-key infrastructure (PKI) or credential system is available to him. We solve this systems problem by entrusting the ISP with a packet attestation mechanism [36] such that the ISP can prove that a pseudonym was sent by U to N_1. We discuss the packet attestation mechanism in §5.

2. Circuit Extension: To extend a circuit to N_2, U generates a new pseudonym X_2 of an ntor instance, signs X_2 and the current timestamp with the secret value x_1 associated with X_1, and sends an extend request to N_1 along with the identifier for N_2, $\{X_2||\mathrm{ts}_{x_2}\}_{\sigma_{X_1}}$ and a timestamp ts_{x_2}. Notice that the extension request is encrypted by a symmetric session key negotiated between U and N_1.

Upon receiving a message, N_1 decrypts and verifies $\{X_2||\mathrm{ts}_{x_2}\}_{\sigma_{X_1}}$ using the previously received pseudonym X_1 and timestamp. We call this verification *pseudonyms linkability verification.* If the signature is valid, it creates an evidence record as discussed in Step 4, signs X_2 using its private key to generate $\{X_2||\mathrm{ts}2\}_{\sigma_{sk_1}}$ and sends a circuit create request to the node N_2 with $\{X_2||\mathrm{ts}2\}_{\sigma_{sk_1}}$.

Node N_2, upon receiving a circuit creation request along with $\{X_2||ts2\}_{\sigma_{sk_1}}$, verifies the signature. Upon a successful verification, it replies to N_1 with an ntor authentication response for the OR key agreement and generates the OR session key for its session with (unknown) user U. N_1 sends the authentication response back to U using their OR session, who then computes the session key with N_2 and continues to build its circuit to N_3 in a similar fashion.

Notice that we carefully avoid any conceptual modification of the OR circuit construction protocol; the above signature generation and verification steps are the only adjustments that BackRef makes to this protocol.

3. Stream Verification: Once a circuit $\langle U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3 \rangle$ has been established, the user U can utilize it to send her web stream requests. To open a

TCP connection, the user sends a *relay begin* cell to the exit node N_3 through the circuit. The user U includes a pseudonym signature (or stream request signature) on the cell contents signed with the secret exponent x_3 of X_3. The user also includes a timestamp in her stream request. When the *relay* cell reaches the exit node N_3, the exit node verifies the pseudonym signature with X_3. Once the verification is successful and the timestamp is current, N_3 creates the evidence log (Step 4) and proceeds with the TCP handshake to the destination server. The *relay* stream request is discarded otherwise.

This stream verification helps N_3 to prove linkability between its handshakes with the destination server and the pseudonym X_3 it has received from N_2. When a whitelist directory exists, the exit node first consults the directory and if the request (*i.e.,* web stream request) is whitelisted, the exit node just forwards it to the destination server. In such a case, the exit node does not require any signature verification and also does not create an evidence log. We further discuss the server whitelisting in §4.4.

4. Log Generation: After every successful pseudonym linkability or stream verification, the evidence record is created. A pseudonym linkability verification evidence record associates linkability between two pseudonyms X_i and X_{i+1} and an endorsement signature on X_i, while a stream verification evidence record associates a stream verification with an endorsement signature on X_3 for N_3.

5. Repudiation or Traceability: The verifier contacts the exit node N_3 with the request information (e.g., IP address, port number, and timestamp) for a malicious stream coming out of the exit node N_3. The operator of N_3 can determine a record using the stream request information. This evidence record verifiably reveals the identity of the middle node N_2.

As an *optional* next step, using the evidence records, it is possible for N_2 to verifiably reveal the identity of its predecessor N_1. Then, the last mile of a full traceability is to reach from N_1 to the user U in a verifiable manner using the record on N_1 and the request information on the ISP [36]. When the user U is an OR node a record at N_1 is sufficient and the last mile problem does not exist.

4.3 Cryptographic Details

BLS Signatures. For pseudonym and endorsement signatures, we use the short signature scheme of Boneh, Lynn and Shacham (BLS) [38]. Consider two Gap co-Diffie-Hellman groups (or co-GDH group) $\mathbb{G}_1$ and $\mathbb{G}_2$ and a multiplicative cyclic group $\mathbb{G}_T$, all of the same prime order p, associated by a bilinear map [39] $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Let g_1, g_2, and g_T be generators for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ respectively and let a full-domain hash function $H : \{0,1\}^* \to \mathbb{G}_1$. The BLS signature scheme [38] comprises following three algorithms:

Key Generation: Choose random $sk \in_R \mathbb{Z}_p$ and compute $pk = g_2^{sk}$. The private key is sk, and the public key is pk.

Signing: Given a private key $pk \in \mathbb{Z}_p$, and a message $m \in \{0,1\}^*$, compute $h = H(m) \in \mathbb{G}_1$ and signature $\sigma = h^{sk}$, where $\sigma \in \mathbb{G}_1$.

Verification: Given a public key $pk \in \mathbb{G}_2$, message $m \in \{0,1\}^*$, and signature $\sigma \in \mathbb{G}_1$, compute $h = H(m) \in \mathbb{G}_1$ and verify that (g_2, pk, h, σ) is a valid co-Diffie-Hellman tuple.

We choose the BLS signature scheme due to the shorter size of their signatures; however, if signing and verification efficiency is more important, we can choose faster signature schemes such as [40].

Circuit Extension. To extend the circuit $\langle U \leftrightarrow N_1 \rangle$ to the next hop N_2, the user U chooses $x_2 \in_R \mathbb{Z}_p$ and generates a pseudonym $X_2 = g_2^{x_2}$, where $g_2 \in \mathbb{G}_2$. U then signs the pseudonym X_2 and the current timestamp[2] value ts_{x_2} with pseudonym X_1 as public key to obtain a signature $\sigma_{X_1} = H(X_2||\mathrm{ts}_{x_2})^{x_1}$. Upon receiving the signed pseudonym $\{X_2||\mathrm{ts}_{x_2}\}_{\sigma_{X_1}}$ along with the timestamp ts_{x_2}, the node N_1 checks if the timestamp is current and verifies it as follows:

$$e(H(X_2||\mathrm{ts}_{x_2}), X_1) \stackrel{?}{=} e(\sigma_{X_1}, g_2)$$

Pseudonym Endorsement. After successful verification, N_1 creates an endorsement signature $\sigma_1 = H(X_2||\mathrm{ts}_2)^{sk_1}$ for pseudonym X_2 and current timestamp ts_2 using its signing key sk_1 and sends it along with X_2 and ts_2 to N_2.

The node N_2 then follows the pseudonym endorsement step. Upon receiving the signed pseudonym $\{X_2||\mathrm{ts}_2\}_{\sigma_1}$, N_2 verifies it as follows:

$$e(H(X_2||\mathrm{ts}_2), pk_1) \stackrel{?}{=} e(\sigma_1, g_2).$$

On a successful verification, N_2 continues with the OR protocol.

Stream Verification. To generate a stream request signature, the user signs the stream request (*i.e.,* selected contents of the *relay begin* cell) using the pseudonym $X_3 = g_2^{x_3}$ where x_3 is the secret corresponding to X_3. For contents of the *relay* cell $m = \{\mathrm{address}\|\mathrm{port}\|\mathrm{ts}_{x_m}\}$, the stream request signature σ_{X_3} is defined as $\sigma_{X_3} = H(m)^{x_3}$. The user sends the signature along with the *relay* cell and the current timestamp ts_{x_m} to the exit node through the already-built circuit.

Once the signed stream request reaches N_3, it verifies the signature as follows:

$$e(H(m), X_3) \stackrel{?}{=} e(\sigma_{X_3}, g_2).$$

Upon a successful verification, the exit node N_3 proceeds with the TCP handshake. A verified request allows the node to link X_3 and the request.

Note that when the destination server ensures an authenticated end-to-end connection with the user, stream verification of the stream request (*relay begin*) suffices; otherwise, the user should sign and the exit node should verify each *relay data* cell to avoid any content modification attack by the exit node.

Log Generation. After every successful pseudonym or stream verification, an evidence record is added to the evidence log. The evidence records differ with nodes' positions within a circuit, and we define two types of evidence logs.

---

[2] Here, in presence of evidence records, we require only coarse-grained timestamps (e.g., dd/mm/yyyy:hh) for replay prevention. Moreover, in the low-latency ACNs, we avoid fine-grained timestamps as they may lead to (offline) traffic-analysis attacks.

Exit node log: For every successful stream verification, an evidence record is added to the evidence log at the exit node. A single evidence record consists of the signature on X_3 (i.e., $\{X_3||\text{ts}_3\}_{\sigma_2}$), and the stream request $H(m)$ coupled by the pseudonym signature $\{m\}_{\sigma X_3}$ and the timestamp ts_{x_m}.

Middle and entry node log: The middle and entry node evidence record comprises two pseudonyms X_i, X_{i+1}, and a timestamp value $\text{ts}_{x_{i+1}}$ coupled with the appropriate signatures and the IP address of N_{i-1}. The pseudonym X_i is coupled with an endorsement signature $\{X_i||\text{ts}_i\}_{\sigma_{i-1}}$ from node N_{i-1}, and the pseudonym X_{i+1} is coupled by a pseudonym signature $\{X_{i+1}||\text{ts}_{x+1}\}_{\sigma X_i}$. When the user is not an OR node and does not posse a verifiable signature key pair, the corresponding record at N_1 consists of a signed pseudonym $\{X_2||\text{ts}_{x_2}\}_{\sigma X_1}$, pseudonym X_1, timestamp value ts_{x_2}, and the IP of the user.

Repudiation or Traceability. Given the server logs of a stream request, an evidence record corresponding to the stream request can be obtained. In the first step, it is checked whether the timestamp matches the stream request under observation. In the next step, the association between the stream request and the pseudonym of the exit node X_3 is verified using the pseudonym signature. Then, the association of the pseudonym X_3 and N_2 is checked using the pseudonym endorsement signature. Given the pseudonym X_3 and a timestamp ts_{x_m}, the backward traceability verification at node N_2 is carried out as follows:

1. Do a lookup in the evidence log to locate the signed pseudonym $\{X_3||\text{ts}_{x_3}\}_{\sigma X_2}$ and the timestamp ts_{x_3}, where X_3 is the lookup index.
2. Compare the timestamps (ts_{x_m} and ts_{x_3}) under observation and prove the linkability between X_2 and X_3 by verifying the signature $\{X_3||\text{ts}_{x_3}\}_{\sigma X_2}$.
3. If verification succeeds, reveal the IP address of the node N_1 who has forwarded X_2 and verify $\{X_2||\text{ts}_2\}_{\sigma_1}$ with pk_1.

The above three steps can be used repeatedly to reach the entry node. However, they cannot be used to verifiably reach the user if we do not assume any public key and credential infrastructure for the users. Instead, our protocol relies on the ISP between user U and N_1 to use packet attestation [36] to prove that the pseudonym X_1 was sent from U to N_1.

4.4 Exit Node Whitelisting Policies

To provide a good balance between anonymity and accountability, we include a whitelisting option for exit nodes. This option allows a user to avoid the complete verification and logging mechanisms if her destination is in the whitelist directory of her exit node. In particular, we categorize the destinations into two groups:

Whitelisted Destinations: For several destinations such as educational `.edu` websites, an exit node may find traceability to be unnecessary. The exit node includes such destinations in a whitelist directory such that, for these destinations, it does not require any endorsement and pseudonym signatures. Traffic sent to these whitelisted destinations through the circuit remains anonymous in the current ACNs sense as the sender does not have to employ BackRef. In that case, to protect malicious user's access, such destinations may use end-to-end blacklisting systems such as Nymble [15] and its successors [16].

Non-listed Destinations: For destinations that are not listed in the exit-node whitelist directory, the user has to use BackRef while building the circuit to it; otherwise, the exit node will drop her requests to the non-listed destinations.

We emphasize that BackRef is *not* an "all-or-nothing" design alternative: it allows an ACN to conveniently disable the complete verification and logging mechanisms for some pre-selected destinations. In particular, an exit node with "Sorry, it is an anonymity network, no logs" opinion can still whitelist the whole Internet, while others employ BackRef for non-whitelisted sites. The use of BackRef is transparent, and users can choose if they wish to use a BackRef node for their circuits.

5 Systems Aspects and Discussion

Communication Overhead. Communication overhead for BackRef is minimal: every circuit creation, circuit extension, and stream request carries a 32 byte BLS signature and additional 4 byte timestamp.

Computation Overhead. In a system with BackRef, every node has to verify a signature and generate another. Using the pairing-based cryptography (PBC) library, a BLS signature generation takes less than $1ms$ while a verification requires nearly $3ms$ for 128-bit security on a commodity PC with an Intel i5 quad-core processor with 3.3 GHz and 8 GB RAM. Signing and verification time (and correspondingly system load) can be further reduced using faster signature schemes (*e.g.,* [40]).

Log Storage. BackRef requires nodes to maintain logs of cryptographic information for potential use by law enforcement. These logs are not innocuous, and the implications of publicly disclosing a record need to be considered. The specificity of the logs should be carefully designed to balance minimal disclosure of side-information (such as specific timings) while allowing flows to be uniquely identified. It must also be possible to reconstruct the logged data from the types of information available to law enforcement. The simplest entry would contain the destination IP, source (exit node) IP, a coarse timestamp, as well as the signature. Logs should be maintained for a pre-defined period and then erased.

No single party can hold the logs without entrusting this entity with the anonymity of all users. The OR nodes can retain the logs themselves. This, however, would require law enforcement to acquire the logs from every such node and consequently involve the nodes in the investigation—a scenario that may not be desirable. Furthermore, traceability exposes nodes of all types, not just exit nodes, to investigation. We are aware of a number of entities who deliberately run middle nodes in Tor to avoid this exposure. An alternative is to publish encrypted logs, where a distributed set of trustees share a decryption key and act as a liaison to law enforcement, while holding each other accountable by refusing to decrypt logs of users who have not violated the traceability policy. Such an entity acts in a similar fashion to the group manager schemes based on group signatures [11].

Non-cooperating Nodes. Given the geographic diversity of the ACNs, it is always possible that some proxy nodes will cooperate with the BackRef mechanism, while others will not. The repudiation property of BackRef ensures that a cooperating node can always at least correctly shift liability to a non-cooperating node. Moreover, such a cooperating node may also *reactively* decide to block any future communication from the non-cooperating node as a policy.

Venturing the Last Mile. In the scenarios where full traceability is required, we need a mechanism for solving the last mile problem addressed in the previous sections. BackRef does not introduce any PKI for the users, therefore our protocol has to rely on some trust mechanism to prove the linkability between the IP address of the user and the entry node pseudonym. For this purpose, we consider an ISP with a packet attestation mechanism [36] to be a proper solution that adds a small overhead for the existing ISP infrastructure and at the same time does not harm any of the properties provided by the ACN. In some countries there is an obligation for the ISPs to retain data that identify the user. In other countries the ISPs are not obligated by law, but it is nevertheless common practice. The protocol is designed in a way that the ISP has to attest only to the *ClientKeyExchange* message (this message is a part of the TLS establishing procedure, and also is public and not encrypted message) which is used to establish the initial TLS communication. This message does not reveal any sensitive information related to the identity of the user. By its design, we reuse this message as a pseudonym for the entry OR node.

6 Security Analysis

We conduct a formal security analysis of BackRef. We model our protocol from §4 (in a restricted form) in the applied pi calculus [41] and verify its important properties, *i.e.,* anonymity, backward traceability, no forward traceability, and no false accusation with ProVerif [42], a state-of-the-art automated theorem prover that provides security guarantees for an unbounded number of protocol sessions.

We model backward traceability and no false accusation as trace properties, and anonymity and no forward traceability as observational equivalence relations. The employed ProVerf scripts as well as an extended version of the paper are available online [43],[44].

Basic Model. We model the OR protocol in the applied pi calculus to use circuits of length three (*i.e.,* one user and three nodes); the extension to additional nodes is straightforward. To prove different security properties we upgrade the model to use additional processes and events. To solve the last mile problem, our model involves an honest ISP which can prove the existence of a communication channel between the user and the entry node. This channel is modeled as private, preventing any ISP log forgeries. The cryptographic log collection model is designed in a decentralized way such that nodes retain the logs themselves in a table that is inaccessible to the adversary.

We model the flow of the pseudonyms and the onion, together with the corresponding verification. However, we do not model the underlying, cryptographically verified 1W-AKE ntor [31] protocol, and assume that the session key between the user and the selected OR process is exchanged securely. The attacker is a standard Dolev-Yao active adversary with full control over the public channels: It learns everything ever send on the network, and can create and insert messages on the public channels. It also controls network scheduling.

Backward Traceability. The goal of our protocol is to trace the source of the communication stream starting from an exit node. We verify that the property of backward traceability arrives from the correctness of the (backward) verification mechanism. The correctness property can be formalized in ProVerif as follows:

$$\begin{aligned} &TraceUser(IP) \Longrightarrow (LookupISP(X_1, IP) \Longrightarrow (RevealPred(IP)) \Longrightarrow \\ &(RevealPred(ipN1)) \Longrightarrow (RevealPred(ipN2)) \wedge CheckSig \wedge LookupN3(m))), \end{aligned} \tag{1}$$

where the notation A $\Longrightarrow$ B denotes the requirement that the event A must be preceded by a event B. In our protocol, the property says that the user U is traced if all nodes in the circuit verifiably trace their predecessors and the ISP solves the last mile problem. The traceability protocol P starts with the event $LookupN3(m)$ which means that for a given message m (stream request) the verifier consults the log, and if such a request exists, it checks the signature $CheckSig$. Finally, when all conditions are fulfilled, the verifier reveals the identity of the predecessor node $RevealPred(ipN2)$. This completes the nested correspondence $(CheckSig \wedge LookupN3(m) \wedge RevealPred(ipN2))$ which verifiably traces N_2. In a similar fashion, the verifier traces N_1 and U.

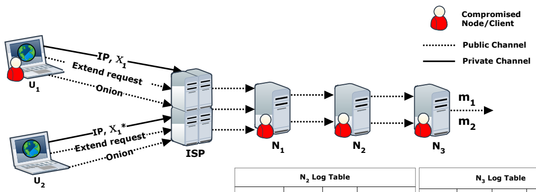
To solve the last mile problem, after the identity of U is revealed the verifier lookup into the evidence table of the ISP ($LookupISP$) to prove the connection between the identity of the user IP and the pseudonym of the entry node X_1. If such a record exists, the event $TraceUser(IP)$ is executed.

Theorem: The trace property defined in equation (1) holds true for all possible executions of process P.

Proof. Automatically proven by ProVerif. □

No False Accusation. There are two aspects associated with false accusations:

1. It should not be possible for a malicious node N_A to trace a communication stream to an OR node N_C other than to its predecessor in the corresponding circuit. Informally, to break this property, N_A has to be obtain a signature of N_C on a particular pseudonym associated with the circuit. This requires N_A to forge a signature for N_C, which is not possible due to the unforgeability property of the signature scheme.
2. It should not be possible for a malicious node N_A to trace a communication stream to a circuit C_1 other than the circuit C_2 employed for the communication stream. Consider a scenario where two concurrent circuits (C_1 and



ISP Log Table	
IP Address	Pseudo. N_1
U_1IP	X_1
U_2IP	X_1*

N_1 Log Table			
Hashed pseudonym	IP Address	Pseudo.	Signed Pseudo.
$H(X_2)$	U_1IP	X_1	$(X_2 \| ts_1)\sigma$
$H(X_2*)$	U_2IP	X_1*	$(X_2* \| ts_2)\sigma$

N_2 Log Table			
Hashed pseudonym	IP Address	Pseudo.	Signed Pseudo.
$H(X_2)$	N_1IP	X_2	$(X_2 \| ts_1)\sigma$
$H(X_2*)$	N_1IP	X_2*	$(X_2* \| ts_2)\sigma$
$H(X_3)$	N_1IP	X_2	$(X_3 \| ts_1)\sigma$
$H(X_3*)$	N_1IP	X_2*	$(X_3* \| ts_2)\sigma$

N_3 Log Table			
Lookup Key	IP Address	Pseudo.	Signed Pseudo.
$H(X_3)$	N_2IP	X_3	$(X_3 \| ts_1)\sigma$
$H(X_3*)$	N_2IP	X_3*	$(X_3* \| ts_2)\sigma$
$H(m_1)$	N_2IP	X_3	$(m_1)\sigma$
$H(m_2)$	N_2IP	X_3*	$(m_2)\sigma$

Fig. 2. No False Accusation adversarial model

C_2), established by two different users U_1 and U_2, pass through a malicious node N_A. Suppose that N_A collaborates with U_2 who is misbehaving and has used the OR network for criminal activities. To help U_2 by falsely accusing a different predecessor, N_A must forge two signatures: To link two pseudonyms X_{1i-1} and X_{2i} from circuits C_1 and C_2 respectively, N_A has to forge the pseudonym signature on X_{2i} with X_{1i-1} as a public key, or he has to know the temporal signing key pair for the predecessor in C_1.

Intuitively, the first case is ruled out by the unforgeability property of the signature scheme. We model the later case as a trace property. Here, even when N_A collaborates with U_2, it cannot forge the signed pseudonym received from its predecessor. The property remains intact as long as one of the nodes on C_1 and the packet attesting ISP [36] remain uncompromised. In the absence of a PKI or credential system for users, the last condition is unavoidable.

We formalize and verify the latter case of the property in an adversarial model where the attacker has compromised one user (U_1 or U_2). Figure 2 provide a graphical representation of the protocol P. We upgrade the basic model involving additional user U_2 who sends additional message m_2. As mentioned before, to simulate the packet attesting mechanism [36] we involve a honest ISP between the user and the entry node. The ISP only collects data that identifies the user (IP address of the user) and the pseudonym for the entry node (X_1), which is send in plain-text. The adversary does not have access to the log stored by the ISP. We want to verify that for all protocol executions the request m_i cannot be associated with any user U_i other than the originator. To formalize the property in ProVerif, we model security-related protocol events with logical predicates. The event CorrISP defines the point of the protocol where the ISP is corrupted. In absence of support for timestamp in ProVerif, we model timestamp values ts for circuits as fresh *nonces*. The property can be formalized as follows:

$$Accuse(IP, m) \Longrightarrow CorrISP. \quad (2)$$

It says that if a user with address IP is falsely accused for a message m, i.e. $Accuse(IP, m)$, then indeed the ISP has to be corrupted.

Theorem: The trace property defined in equation (2) holds true for all possible executions of process P.

Proof. Automatically proven by ProVerif. □

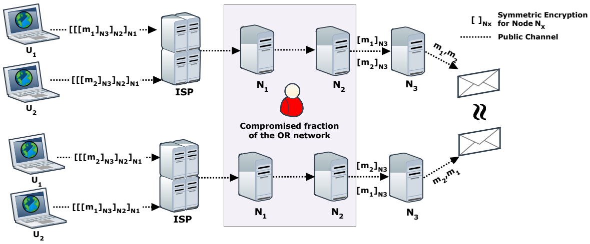


Fig. 3. Anonymity Game

Anonymity. We use observational equivalence to formalize privacy related properties such as in [45], [46]. We model anonymity as an equivalence relation between two processes that are replicated an unbounded number of times and execute in parallel. In the first process P, users U_1 and U_2 send two messages m_1 and m_2, respectively. While in the second process Q the two messages are swapped. If the two defined processes are observationally equivalent (P $\approx$ Q), then we say that the attacker cannot distinguish between m_1 and m_2 i.e. cannot learn which message is sent by which user. In our scenario we assume that the attacker can compromise some fraction of the OR nodes, but not all of them. Figure 3 provides a graphical representation of the anonymity game where the exit node N_3 is honest. The game works as follows:

1. U_1 and U_2 create an onion data structure O_1 and O_2, respectively, intended for N_3 and send via previously built circuits C_1 ($U_1 \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3$) and C_2 ($U_2 \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3$). Nodes communicate between each other through public channels. 2. Two of the intermediate nodes are corrupted and the attacker has full control over them. The intermediate compromised nodes (in our case N_1 and N_2) remove one layer of encryption from O_1 and O_2 and send the onion to the exit node N_3. 3. After receiving these two onions from the users U_1 and U_2 and possibly other onions from compromised users, the exit OR node N_3 removes the last layer of the encryption and publishes the message on a public channel.

Note that the ISP does not affect the anonymity game and only acts as a proxy between the users and the outside world. For the verification, we assume that U_1 and U_2 are honest and they follow the protocol. Nevertheless, the action of any compromised user and honest users can be interleaved in any order.

Theorem: The observational equivalence relation $P \approx Q$ holds true.

Proof. Automatically proven by ProVerif. □

No Forward Traceability. The evidence log of the backward traceability protocol in BackRef does not store any information (*i.e.,* IP addresses) that can identify or verifiably reveal the identity of a node's successor. The log contains only the pseudonym for the successor node which does not reveal anything about the identity of the node.

We formalize this property as an observational equivalence relation between two distinct processes and verify that an adversary cannot distinguish them. Figure 4 provides a graphical representation of the game. To prove the observational equivalence, we model a scenario with concurrent circuit executions.

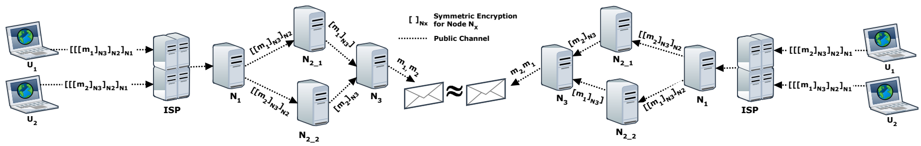


Fig. 4. No Forward Traceability

In this game, the adversary can corrupt parties and extract their secrets only after the message transmission over the circuit has completed. For this game, our model involves an additional middle node and user U_2. Two users U_1 and U_2 send two different messages m_1 and m_2 via two circuits. We verify that it is impossible for an attacker to deduce any meaningful information about the successor node for a particular request. Our game works as follows: 1. U_1 and U_2 start the protocol and construct two different circuits, $C_1(U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3)$ and $C_2(U \leftrightarrow N_1 \leftrightarrow N_2^* \leftrightarrow N_3)$ respectively, with adequate values (x_1, x_2, x_3) for a circuit C_1 and (x'_1, x'_2, x'_3) for C_2. 2. U_1 and U_2 create an onion data structure O_1 and O_2 and send to the exit node N_3 via previously built circuits C_1 and C_2. Nodes communicate with each other through public channels. 3. After receiving the two onions from the users and possibly other onions from compromised users, N_3 removes the last layer of the encryption and publishes the messages on a public channel. 4. After protocol completion, the entry node N_1 is compromised and the adversary obtains the evidence log.

In the first process P, U_1 sends m_1 and U_2 sends m_2, while the process Q is reversed process P. For the no forward traceability verification, we assume that all other parties in the protocol remain honest, except the compromised N_1. For example, if two neighbor nodes are compromised, the no forward traceability can be easily broken by activating the backward traceability mechanism.

Theorem: The observational equivalence relation $P \approx Q$ holds true.

Proof. Automatically proven by ProVerif. □

Finally, to the best of our knowledge, our formal analysis is the first ProVerif-based analysis of the OR protocol; it can be of independent interest towards formalizing and verifying other properties of the OR protocol.

7 Conclusion

We presented BackRef, an accountability mechanism for ACNs that provides practical repudiation for the proxy nodes, allowing selected outbound traffic flows to be traced back to the predecessor node. It also provides a full traceability option when all intermediate nodes are cooperating. While traceability mechanisms have been proposed in the past, BackRef is the first that is both compatible with low-latency, interactive applications (such as anonymous web browsing) and does not require group managers or credential issuers. BackRef is provably secure, requires little overhead, and can be adapted to a wide range of anonymity systems. We also analyzed some important systems issues (namely, white-listing, log storage, non-cooperating nodes, and the last mile problem) with any reactively accountable ACN, and presented plausible options towards deploying BackRef in practice.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable comments. We would also like to thank Kim Pecina for his assistance with ProVerif proofs. This work was supported by the German Ministry for Education and Research through funding for the Center for IT-Security, Privacy and Accountability and the German Universities Excellence Initiative.

References

1. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. J. Cryptology 1(1) (1988)
2. Corrigan-Gibbs, H., Ford, B.: Dissent: accountable anonymous group messaging. In: CCS, pp. 340–350 (2010)
3. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: IEEE Symposium on Security and Privacy (1997)
4. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. CACM 24(2) (1981)
5. Mittal, P., Borisov, N.: Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies. In: CCS, pp. 161–172 (2009)
6. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: USENIX Security (2004)
7. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster Protocol— Version 2. IETF Internet Draft (2003), `http://mixmaster.sourceforge.net/`
8. Janssen, A.W.: Tor madness reloaded (2007), `http://itnomad.wordpress.com/2007/09/16/tor-madness-reloaded/` (accessed January 2014)
9. AccusedOperator: Raided for operating a Tor exit node (2012), `http://raided4tor.cryto.net/`
10. Köpsell, S., Wendolsky, R., Federrath, H.: Revocable anonymity. In: Müller, G. (ed.) ETRICS 2006. LNCS, vol. 3995, pp. 206–220. Springer, Heidelberg (2006)
11. von Ahn, L., Bortz, A., Hopper, N.J., O'Neill, K.: Selectively traceable anonymity. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 208–222. Springer, Heidelberg (2006)
12. Diaz, C., Preneel, B.: Accountable anonymous communication. In: Security, Privacy, and Trust in Modern Data Management (2007)
13. Golle, P.: Reputable mix networks. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 51–62. Springer, Heidelberg (2005)
14. Clark, J., Gauvin, P., Adams, C.: Exit node repudiation for anonymity networks. In: On the Identity Trail: Privacy, Anonymity and Identity in a Networked Society. Oxford University Press (2009)
15. Johnson, P.C., Kapadia, A., Tsang, P.P., Smith, S.W.: Nymble: Anonymous IP-address blocking. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 113–133. Springer, Heidelberg (2007)
16. Henry, R., Goldberg, I.: Formalizing anonymous blacklisting systems. In: IEEE Symposium on Security and Privacy, pp. 81–95 (2011)
17. Goldberg, I., Wagner, D., Brewer, E.: Privacy-enhancing technologies for the internet. In: IEEE Compcon. (1997)

18. Goldberg, I., Shostack, A.: Freedom network 1.0 architecture and protocols. Technical report, Zero-Knowledge Systems (2001)
19. Kate, A., Zaverucha, G.M., Goldberg, I.: Pairing-based onion routing with improved forward secrecy. ACM Trans. Inf. Syst. Secur. 13(4) (2010)
20. Danezis, G., Goldberg, I.: Sphinx: A compact and provably secure mix format. In: IEEE Symposium on Security and Privacy (2009)
21. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: design of a type iii anonymous remailer protocol. In: IEEE Symposium on Security and Privacy (2003)
22. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization v0.34, http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf (August 2010)
23. Wolinsky, D.I., Corrigan-Gibbs, H., Ford, B., Johnson, A.: Dissent in numbers: making strong anonymity scale. In: OSDI (2012)
24. Corrigan-Gibbs, H., Wolinsky, D.I., Ford, B.: Proactively accountable anonymous messaging in verdict. In: USENIX Security (2013)
25. Danezis, G., Sassaman, L.: How to bypass two anonymity revocation schemes. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 187–201. Springer, Heidelberg (2008)
26. TorProject: Exonerator Service (2012), https://exonerator.torproject.org/ (accessed January 2014)
27. Øverlier, L., Syverson, P.F.: Improving efficiency and simplicity of tor circuit establishment and hidden services. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 134–152. Springer, Heidelberg (2007)
28. Kate, A., Goldberg, I.: Using sphinx to improve onion routing circuit construction. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 359–366. Springer, Heidelberg (2010)
29. Backes, M., Kate, A., Mohammadi, E.: Ace: an efficient key-exchange protocol for onion routing. In: WPES (2012)
30. Catalano, D., Fiore, D., Gennaro, R.: Certificateless onion routing. In: CCS (2009)
31. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Designs, Codes and Cryptography (2012)
32. Camenisch, J.L., Lysyanskaya, A.: A formal treatment of onion routing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 169–187. Springer, Heidelberg (2005)
33. Danezis, G., Diaz, C., Troncoso, C., Laurie, B.: Drac: An architecture for anonymous low-volume communications. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 202–219. Springer, Heidelberg (2010)
34. Shimshock, E., Staats, M., Hopper, N.: Breaking and provably fixing minx. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 99–114. Springer, Heidelberg (2008)
35. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
36. Haeberlen, A., Fonseca, P., Rodrigues, R., Druschel, P.: Fighting cybercrime with packet attestation. Technical report, MPI-SWS (2011)
37. Dingledine, R., Mathewson, N.: Tor Protocol Specification (2008), https://gitweb.torproject.org/torspec.git/tree/HEAD (accessed January 2014)
38. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 514. Springer, Heidelberg (2001)

39. Blake, I., Seroussi, G., Smart, N., Cassels, J.W.S.: Advances in Elliptic Curve Cryptography. Cambridge University Press (2005)
40. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (2011)
41. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: POPL (2001)
42. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: CSFW (2001)
43. BackRef: Introducing accountability to anonymity networks (proverif scripts), `http://crypsys.mmci.uni-saarland.de/projects/BackRef/`
44. Backes, M., Clark, J., Kate, A., Simeonovski, M., Druschel, P.: Backref: Introducing accountability to anonymity networks, `http://arxiv.org/abs/1311.3151`
45. Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols, 435–487 (2009)
46. Chothia, T.: Analysing the MUTE anonymous file-sharing system using the pi-calculus. In: Najm, E., Pradat-Peyre, J.-F., Donzeau-Gouge, V.V. (eds.) FORTE 2006. LNCS, vol. 4229, pp. 115–130. Springer, Heidelberg (2006)