Fuchs *et al. EURASIP Journal on Information Security* (2015) 2015:1
DOI 10.1186/s13635-014-0018-0

EURASIP Journal on Information Security
a SpringerOpen Journal

RESEARCH Open Access

Laribus: privacy-preserving detection of fake SSL certificates with a social P2P notary network

Karl-Peter Fuchs*, Dominik Herrmann, Andrea Micheloni and Hannes Federrath

Abstract

In this paper we present Laribus, a peer-to-peer network designed to detect local man-in-the-middle attacks against secure socket layer/transport layer security (SSL/TLS). With Laribus, clients can validate the authenticity of a certificate presented to them by retrieving it from different vantage points on the network. Unlike previous solutions, clients do not have to trust a central notary service nor do they have to rely on the cooperation of website owners. The Laribus network is based on a social network graph, which allows users to form notary groups that improve both privacy and availability. It integrates several well-known techniques, such as secret sharing, ring signatures, layered encryption, range queries, and a distributed hash table (DHT), to achieve privacy-aware queries, scalability, and decentralization. We present the design and core components of Laribus, discuss its security properties, and also provide results from a simulation-based feasibility study.

Keywords: Privacy; Anonymity; Man-in-the-middle attacks; SSL; P2P

1 Introduction

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocol suite [1,2] provides basic security mechanisms such as confidentiality, data integrity, and especially authentication (cf. [3] for a detailed treatment of SSL and its successor TLS). There have been various attempts to attack these protocols, i. e., to eavesdrop on a connection. Attacks on the security of SSL can be categorized into two classes: attacks from the first category exploit security flaws in the protocols, weaknesses in the construction of the cryptographic primitives used or implementation errors (e. g., the BEAST [4], CRIME, BREACH [5], and *Lucky Thirteen* [6] attacks; cf. [7,8] for a comprehensive overview). The second category of attacks, which we are interested in, attacks weaknesses in the *authentication model*.

SSL relies on a X.509 PKI [9] for the purpose of *authenticating* a remote entity's key. This protects clients from disclosing data to adversaries that impersonate a designated destination. Authentication is delegated to certification authorities (CAs) that issue certificates by cryptographically signing the public key of website owners, guaranteeing the authenticity of the association $\langle$*public key*, *website*$\rangle$ or, in some cases, $\langle$*public key*, *website*, *organization*$\rangle$.

In a 2010 study, the Electronic Frontier Foundation found that browsers from Microsoft and Mozilla are (in-)directly trusting more than 650 CAs [10]. Unfortunately, the X.509 model allows *any* CA to certify *any* domain [11]. It only takes a single CA to misbehave (by being hacked, tricked, bribed, or legally forced) to generate a seemingly valid *fake certificate*. Having obtained a fake certificate, an adversary can mount a man-in-the-middle (MitM) attack impersonating any website of his choice. The MitM attack goes undetected because the fake certificate contains a genuine signature and thus appears legitimate. MitM attacks typically affect only users in a confined part of the Internet (cf. Figure 1). The severity of this risk is demonstrated by security incidents involving the two CAs Comodo and DigiNotar. The attackers created fake certificates for high-profile sites (among them Google and Paypal) to intercept the SSL traffic of Internet users in Iran [12]. Solely relying on CAs for authenticating remote servers has shown to be fragile and insecure.

*Correspondence: fuchs@informatik.uni-hamburg.de
Department of Informatics, University of Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany

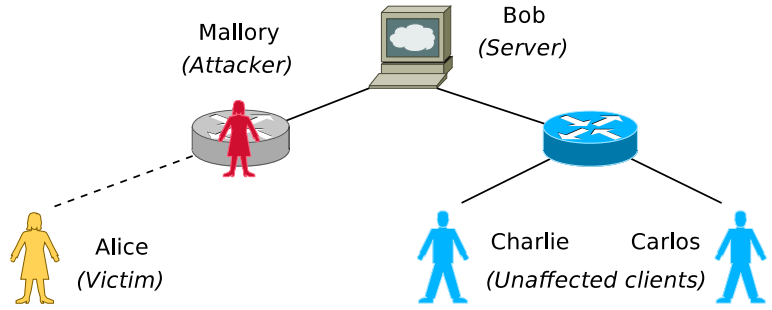

Figure 1 An attacker presents a fake certificate to user Alice during a MitM attack.

Previous proposals aim to fix the CA-based authentication approach either by relying on other hierarchical structures (e. g., the DNS tree) or by introducing *notary servers* that validate certificates on the user's behalf. Both approaches are subject to considerable limitations (cf. Section 2). In contrast, with *Laribus* (referring to guardian deities in ancient Roman religion), we propose that web clients should *collaborate* to validate certificates in a distributed manner. Our *contribution* consists of integrating well-known techniques to organize the clients in *a fully distributed peer-to-peer (P2P) network* that meets security, privacy, and availability expectations of web clients. Laribus provides users with the ability to *model trust relationships that reflect their social relationships*, i. e., users can choose to trust their *friends*, and not unknown organizations, with certificate validation. The P2P architecture of Laribus improves *scalability* and provides *blocking-resistance*. *Privacy-preserving query techniques* protect the user's surfing behavior. A distributed storage mechanism offers *resilience to client churn* and increased performance via *caching*. This paper extends our previously published work in [13], providing a more comprehensive survey of related efforts that aim to improve the security of certificate validation as well as a more detailed description of the privacy-preserving mechanisms of Laribus.

The rest of this paper is structured as follows. In Section 2, we review related work and motivate our design choices. We outline the architecture of Laribus in Section 3 and focus on the details of the cryptographic mechanisms involved in Section 4. We present results of an initial feasibility study in Section 5 and discuss limitations in Section 6. We conclude in Section 7.

2 Related work

The issues with SSL and the CA trust model have been known for years, and various proposals have appeared in the literature, suggesting to replace, amend, or complement the current system. Laribus is a combination and extension of techniques selected from the current state of the art. It does not replace the existing PKI infrastructure but serves as an additional source of trust during certificate validation.

This section provides an overview of the most important research directions, outlining their benefits and limitations. A comprehensive discussion of previous work can be found in [14,15].

In Sections 2.1 and 2.2, we describe approaches that advocate to restrict the domains a CA may issue certificates for. In Section 2.3, we describe the *trust on first use model*. We proceed with proposals that rely on out-of-band information for certificate validation in Section 2.4. Another avenue of research focuses on the application of append-only data structures (cf. Section 2.5). Finally, in Section 2.6, we discuss existing work that relies on *notaries* that provide clients with a third-party perspective of a certificate.

2.1 Limiting the scope of CAs

One of the main limitations of the X.509 PKI is the fact that any CA can issue a certificate for any domain name. The large number of CAs that are trusted by default by common browsers constitutes a significant attack surface. In order to decrease the risk, CAs could be *restricted to specific top-level domains*. Karsten et al. [16] show that in practice most CAs provide online certificates for a small number of top-level domains and that the domains in most top-level domains use certificates from a small set of CAs only. Moreover, some CAs appear to operate for a single organization in a single country only. Browser vendors could incorporate this kind of information into their software in order to warn the user when a certificate for a domain is issued from a CA that has not issued certificates for the corresponding top-level domain so far.

Recent findings from Perl et al. indicate that the number of trusted CAs could be reduced considerably: based on scans of all publicly reachable web servers listening on port 443, they observed that 34% of the trusted CAs have not issued a single certificate yet [17]. They suggest to *remove the root certificates* of these CAs from the browser

trust stores. However, this approach may have undesirable consequences for closed user groups that use non-public HTTPS sites that use a certificate of one of the deleted CAs. These users will see a warning message when they access their secure sites, which weakens the effectiveness of such warnings [18].

These two measures, limiting the scope of CAs as well as removing unnecessary certificates from the browser trust store, may help to decrease the likelihood of an adversary obtaining a fake certificate in practice. However, they cannot prevent MitM attacks reliably and have not been integrated into Laribus. Laribus is supposed to be a complementary certificate validation service that provides an additional layer of security. Therefore, it should be oblivious of the trust relationships within the existing PKI trust model.

2.2 Certificate pinning

A complementary solution to limiting the scope of CAs (i. e., limiting the domain names for which a CA may issue certificates) is *certificate pinning*, which approaches the problem from the perspective of domain name owners. Certificate pinning allows them to limit the CAs that are authorized to issue a certificate for a given domain.

Certificate pinning has been introduced by Google in its Chrome browser for a small set of Google domain names. As a predefined whitelist embedded in the browser software scales rather poorly, there are several ongoing efforts that try to provide operators of web servers with a means to publish certificate pins, either via HTTP headers [19] or via TLS extensions [20]. As these proposals require changes to the server as well as the client software, it will take time until they are widely adopted.

Domain name system (DNS)-based Authentication of Named Entities (DANE) [21] is an IETF standard that proposes to utilize the DNS infrastructure to store the pinning information. The operator of a web server can include DANE resource records for his domain that indicate the CAs that have issued its certificates or the fingerprint of the certificate itself.

DANE relies on an existing decentralized hierarchy, the DNS namespace, to authenticate website certificates. The security of this approach depends on the integrity of the data supplied via DNS, i. e., DNSSEC [22] is a mandatory prerequisite in DANE. As pointed out in [23], DANE would significantly reduce the risk of an MitM attack remaining undetected, since an adversary would have to compromise both, a CA and the respective authoritative DNS server. However, leveraging DNS has some limitations in practice: Firstly, only few DNS servers make use of DNSSEC so far and its widespread deployment has proven to be challenging [24]. Moreover, DANE requires changes to the resolver library on client machines as well as adoption by server operators. Finally, parts of the DNS hierarchy are connected to the X.509 PKI infrastructure: A prominent example is Verisign itself, which maintains a certificate authority providing hundreds of thousands of SSL certificates [25]. At the same time, Verisign acts as a DNS registry for several top-level domains, among them `.com` and `.net`, which makes it a particularly weak spot.

DNS Certification Authority Authorization (CAA) [26] is a complementary proposal. CAA allows a server operator to include DNS records in a domain that specify the CAs that are allowed to issue certificates for that domain. CAA-compliant CAs are required to look up the respective CAA records and issue a certificate for the domain only if the CAA records indicate that they are authorized to proceed. A conceptual limitation of the CAA concept is that it assumes that all trusted CAs are implementing this policy. However, CAA cannot prevent an adversary that has gained access of the signing key of a CA from issuing fake certificates.

Certificate pinning may be an effective measure to detect MitM attacks. However, the aforementioned proposals may scale quite poorly: They have to be implemented by the administrators of each and every website (or CA). In contrast, one of the central design goals for Laribus consists in providing a *client-only solution* that enables clients to validate the certificate of any website without involving its operator.

2.3 Trust on first use

Advocates of the *trust on first use method (TOFU)* reject the idea of having to rely on a third-party-issuing certificates that have to be renewed periodically. Instead users are supposed to make a leap of faith, trusting and pinning the certificate that has been presented by a remote host during the very first connection attempt. This approach is familiar from its use in SSH and it also has been proposed for encrypted e-mails [27]. The application of the TOFU approach for SSL certificate validation has been proposed in [28]. An implementation of the concept is the certificate patrol browser extension [29].

However, certificate renewal is handled poorly in the TOFU trust model. It may be difficult for users to distinguish between the case of a certificate having been changed deliberately and legitimately by the operator of a remote host and the case of an MITM attack. Therefore, Laribus does not rely on the TOFU principle.

2.4 Incorporating out-of-band information

The following approaches try to establish secure connections without a set of fixed, potentially untrusted third parties.

Direct Validation of Certificates (DVCert) [30] allows a client to validate the authenticity of a server certificate without having to rely on third parties at all. DVCert leverages the fact that many websites can be personalized,

i. e., users have to log in with application-level credentials (username and password). In DVcert, the fact that client and server know the user's credentials is exploited by the web server to prove to a connecting client that it is in fact talking to the desired web server and not to a man-in-the-middle.

However, first-time users that do not have an account at a web site yet cannot benefit from DVCert during registration. Moreover, DVCert is not suitable to secure sites that are available without any authentication at all. Finally, as with certificate pinning, DVCert has to be implemented in each and every website, which conflicts with our design goal to build a system that does not rely on any cooperation on the server side.

MonkeySphere [31] builds upon the PGP Web of Trust (WoT) concepts of a network of people who trust other people: Users who never met before can safely authenticate their identities due to the presence of a trust path in the network between them, established by friends who trust their friends, respectively. Whenever the MonkeySphere daemon encounters a self-signed or invalid certificate, it searches public key servers for a PGP key associated with that website's name. The certificate is trusted only if the daemon can construct a trust path from the user's key to the server's key. This approach could effectively abolish the need for CAs and allow users to trust self-signed certificates, also providing a theoretically sound way of trusting remote certificates and detecting MitM attacks.

However, MonkeySphere relies on the cooperation of the administrators of the webservers, which conflicts with our design goals: Clients can only validate the certificates of those servers, whose administrators have signed their certificates and uploaded them to a key server. Moreover, administrators have to ensure that their certificates are extensively connected with other users within the PGP Web of Trust to ensure that as many users as possible will be able to find a trust path to the server.

Strictly relying on trust paths between users and target servers may lead to bootstrapping issues: Users will only be able to validate server certificates with high probability, if PGP and MonkeySphere are widely adopted and all users actively contribute to the Web of Trust. However, so far, PGP suffers from poor adoption due to usability issues resulting from its intrinsic complexity.

In contrast, Laribus users can validate the certificate of any server at any time, even when the server is completely oblivious of Laribus. Moreover, bootstrapping Laribus may be easier because users do not have to establish trust paths from them to individual destination servers. In Laribus certificate validation is possible even if only a few small user groups (cliques based on real-world social relationships) participate in the system.

2.5 Append-only timelines

The following proposals advocate a publicly available log file containing all certificate transactions.

Sovereign keys [32] proposes to maintain a verifiable append-only data structure, which contains the history of all SSL-enabled domains. Server operators are supposed to push their authentication data (newly deployed certificates as well as blacklisted old certificates) to one of 20 redundant *Timeline Servers*. Clients interact with them to validate previously unseen certificates. The basic concept has been extended by the proposals *Certificate Transparency* [33] and *Transparent Key Integrity* [34].

Append-only data structure could indeed be the definitive answer to the trust problems in SSL, since attackers would have to publicly insert fake certificates into the data structure to be successful. However, append-only timelines have to be provided via a set of dedicated servers, they require the cooperation of web server administrators or CAs, and querying timeline servers may infringe the privacy of users, who provide the domain names they want to validate. Therefore, Laribus does not contain a permanent global timeline. Instead Laribus incorporates a distributed cache that contains recently validated certificates to improve its performance and availability.

2.6 Relying on notaries and peers

The following proposals advocate to validate certificates by relying on dedicated notary servers or on peers.

Perspectives [35] and its follow-up *Convergence* [36] employ a set of network servers (so-called notary servers) that fetch and store SSL certificates from Internet hosts. In order to validate a server certificate, users connect to one or more of the notary servers and ask them for the certificates they see from their vantage point. An attacker close to the client could then be easily detected by comparing the certificates provided by the notaries with the certificate the client received from the server (cf. Figure 2).

The notary model is based on the assumption that an attacker cannot interfere with both the connection of the users as well as the connection of (all the) queried network notaries. In the original design of Perspectives and Convergence, notaries are full-blown dedicated servers, which have to be maintained by their operators to be always up and running. As a result, there is only a limited number of them, i. e., more powerful man-in-the-middle adversaries may be able to attack a user as well as all queried notaries. In order to reduce the risk of such attacks, notaries can be spread over multiple autonomous systems (AS) as proposed in [37].

Moreover, users must put considerable trust into notaries. From a security perspective, users have to trust them not to lie (or collaborate with an adversary). From a privacy perspective, notaries have to be trusted to operate responsibly. Without any additional means, they learn all

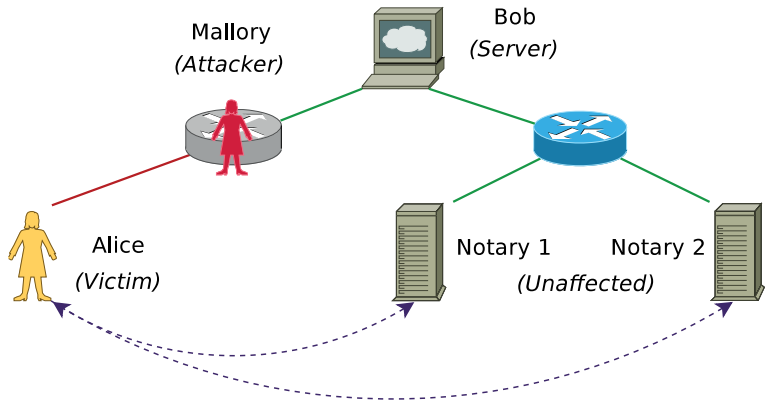

Figure 2 User Alice relies on notaries to detect an MitM attack.

of a user's visited SSL domains. Privacy can be increased by forwarding queries to notaries via the Tor network [38], which requires additional software on clients, though.

These privacy concerns are embraced by *DoubleCheck* [39], which suggests that whenever a client connects to a SSL web server it should retrieve its certificate additionally via the Tor network for comparison. An extension of this design is *DetecTor* [40], which suggests to retrieve each certificate over multiple Tor circuits using different exit nodes for increased trust. It also provides a prototypical implementation. However, using the Tor network for certificate validation may be problematic as users have to trust exit nodes, whose trustworthiness has proven to be questionable in the past [41,42].

While the *ICSI Notary* service [43] is similar in spirit to Perspectives and Convergence, it differs in two important aspects. Firstly, while other approaches either actively scan the Internet [10,44] to obtain certificates or request them from servers upon request, the ICSI Notary collects certificates *passively* by monitoring upstream traffic of various Internet sites. Secondly, the ICSI Notary service can be queried via DNS, which helps to protect the privacy of its users: The notary servers do not see the IP address of a client but only the IP address of their recursive name server. The authors suggest to use a third-party DNS resolver, such as Google Public DNS (8.8.8.8) so that no information about the location of a user is leaked to the notary. While this suggestion succeeds in providing sender anonymity against the notary service, it does not protect the privacy against the *recursive name server*, which can eavesdrop on the users' queries and thus learn which certificates a user wants to validate.

The concept of notaries has also been used in application areas other than SSL certificate validation. We are aware of notary- and peer-based proposals for HTTP queries as well as DNS queries. *Senser* [37] proposes to retrieve websites over multiple proxies in order to validate that the content is not tampered with. However, the concept does not account for privacy, i. e., the selected proxies can observe the IP addresses of the clients as well as the URLs and the content of the websites a user visits. Regarding DNS queries, *DepenDNS* [45] suggests that DNS clients should send their lookup queries to multiple resolvers in order to detect cache poisoning attacks. DepenDNS is based on the cooperative name lookup systems CoDNS [46] and ConfiDNS [47], which advocate that DNS clients should send queries to each other in order to improve integrity, availability, and lookup performance. This peer-to-peer approach is embraced by *DoX* [48], which suggests that DNS resolvers should form a peer-to-peer network to cross-check the responses they obtain. However, all the aforementioned proposals neglect the issue of user privacy. Moreover, they do not implement mechanisms that allow users to express to what degree they are willing to trust any given peer or notary server.

The design of Laribus is derived from the concept of notaries. However, as one of our design goals is to provide a scalable *fully-distributed solution that does not require fixed entities*, we cannot rely on a constrained set of dedicated notary servers. In contrast, in Laribus, multiple clients collaborate to provide notary services.

3 The Laribus proposal

In this section, we will present an overview of the architecture of Laribus and the interactions of the involved components. We will start out with a naïve approach for client-based certificate validation in order to explain the issues that must be overcome. After that, we will explain how the design of Laribus addresses these challenges.

3.1 Shortcomings of a naïve approach

Given the initial scenario in Figure 1, Alice could ask one of the unaffected users, e. g., Carlos, to retrieve the certificate for the server she wants to connect to (Bob). If

Alice receives different certificates from Carlos and Bob, she can detect the MitM attack.

This naïve approach, however, has several shortcomings. First of all, it does not meet users' *security* expectations. Alice wants to be sure that Carlos is neither cooperating with Mallory and therefore not telling her the truth (Problem 1: intentionally false testimonies), nor that Carlos is unknowingly also affected by Mallory's attack (Problem 2: unintentionally false testimonies). Alice can solve the first problem by asking only friends she trusts to validate certificates. The second problem can be solved by asking users that are dispersed throughout the world, e. g., friends living in different countries.

Secondly, the naïve approach does not meet users' *privacy* expectations. If Alice asks her friends to retrieve certificates on her behalf, her friends will know which servers she connects to. While it is a reasonable assumption that her friends will honestly answer requests for certificates, they must be considered to be *curious*; spying on an acquaintance may be more attractive than snooping on a stranger. Security and privacy seem to be conflicting goals in this respect. Solutions for this dilemma do however exist: Alice could ask a client she trusts, but she doesn't know personally, e. g., a client run by a university. Alternatively, she could use a suitable privacy-enhancing technique to hide her identity, e. g., Tor [38].

Thirdly, the naïve approach does not meet users' *availability* expectations. Alice cannot assume that a friend is online every time she wants to validate a certificate. Typically, Alice will also not be willing to wait until a friend comes online again. This availability issue can either be resolved by caching recently obtained validation results or by resorting to delegate certificate validation to less trustworthy clients.

Finally, certificate validation must be *scalable*. Asking one's friends to validate certificates scales well under three assumptions. Firstly, users that *consume* validation services, i. e., requesting other clients to perform certificate validation on their behalf, do also offer validation services to others, i. e., there is no *free-riding*. Secondly, friends trust each other *mutually*, i. e., they are willing to consume and offer validation services among themselves. Thirdly, the available resources offered by a group of friends are sufficient to handle the aggregate query volume issued by all of its members. However, in reality, it is difficult to force users to honor these assumptions. Therefore, a practical system should be able to cope with unbalanced trust relationships as well as uneven resources and loads.

3.2 Layered network structure

Laribus is a decentralized, distributed peer-to-peer system comprised of clients that collaborate for certificate validation. Each client offers validation on behalf of others and each client can request validation by other clients.

The Laribus P2P network is structured into multiple layers of abstraction (cf. Figure 3). Clients are requested to explicitly point out their friends from the real world, to limit the influence of adversaries. These social relationships are captured in the *social network layer*, a directed graph that reflects the *friendship relations* expressed by the clients. The set of close friends of a user (also referred to as his *clique*), makes up his *trusted core* in Laribus. The resulting friendship graph is typically not fully connected. In fact, some parts of the graph may be isolated from the rest, e. g., in case of cliques that do not point out any friends apart from members of their trusted core. We assume that adversaries will be unable to enter the trusted core, because this typically involves establishing a close social relationship with users in the *real world*.

Apart from pointing out their friends, clients organize themselves into *notary groups*. Group memberships are reflected in the *notary group layer*. Clients within a group collaboratively act as a notary. The members of a notary

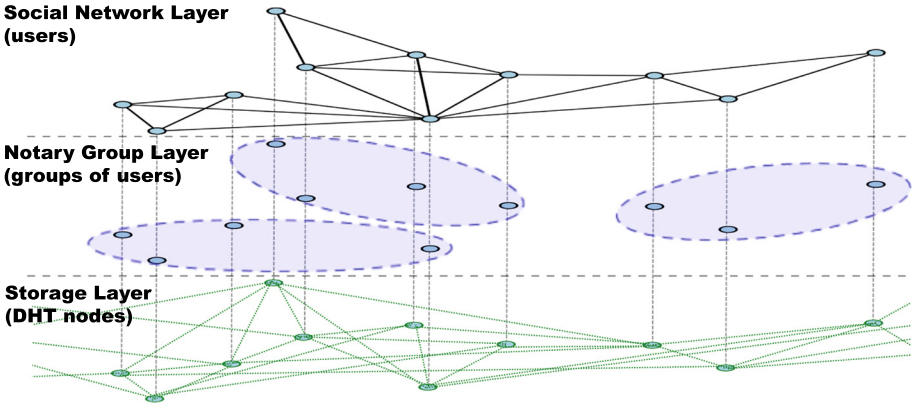

Figure 3 Laribus consists of three independent layers. A social network layer, a notary groups layer, and a storage layer.

group are not supposed to be *fully congruent* with the set of clients within a trusted core.

Finally, clients are expected to contribute resources for distributed storage of validation data. The *storage layer* consists of a global distributed hash table (DHT) based on Kademlia [49], in which Laribus stores all information regarding group memberships and trust relationships. Apart from providing decentralized storage, this layer serves as a cache to provide cheap access to the results obtained from previous certificate validations. Data stored in the DHT is cryptographically signed (cf. Section 4.4).

3.3 Interactions during certificate validation

Figure 4 extends the initial scenario by the components of Laribus. Alice connects to Bob's HTTPS website (Step A in Figure 4). We assume that Alice is subject to an MitM attack perpetrated by Mallory. Other Laribus users, such as members of notary groups 1, 2, and 3 are supposed to be unaffected by the MitM attack. Formally, Alice tries to connect to a server $\mathcal{SRV}$ and is presented with a fake certificate $cert_M$. Some unaffected clients are able to fetch the authentic certificate $cert_B$. In the following, we will only briefly sketch the relevant interactions. For conciseness, we will defer the treatment of the involved security and privacy techniques to Sections 3.5 and 3.7.

The straightforward way to validate certificates in Laribus is via the *direct queries* technique. A requestor, Alice, sends a direct query to a notary group of her choice (Step B in Figure 4). She may choose a random group for this purpose, or a group containing one of her friends, asking members of the selected notary group to retrieve the certificate of $\mathcal{SRV}$ from their respective vantage points. The notary group exchanges the obtained certificates in order to reach a consensus about the certificate of $\mathcal{SRV}$ by means of a majority vote: The winning certificate is signed by the group's *notary signature* and returned to Alice (not shown). Alice validates the certificate presented to her by Bob by comparing it with the result obtained from the notary group.

Issuing direct queries for every certificate validation is inefficient. An alternative way for a client to validate certificates consists of retrieving certificates that have been obtained by other clients via direct queries with the *DHT Lookup* technique. Every time a notary group obtains a consensus about a server's certificate, it signs and stores it in the DHT (Step F in Figure 4, further discussed in Section 4.4). Instead of issuing a direct query, clients can look up both the unknown certificate's hash or hostname of $\mathcal{SRV}$ in the DHT (Step E). Summing up, in Laribus, a client performs the following steps to validate a certificate:

1. Alice performs a DHT lookup to determine whether $cert_M$ has been seen previously. If there are entries for $cert_M$, she retrieves them and (depending on how

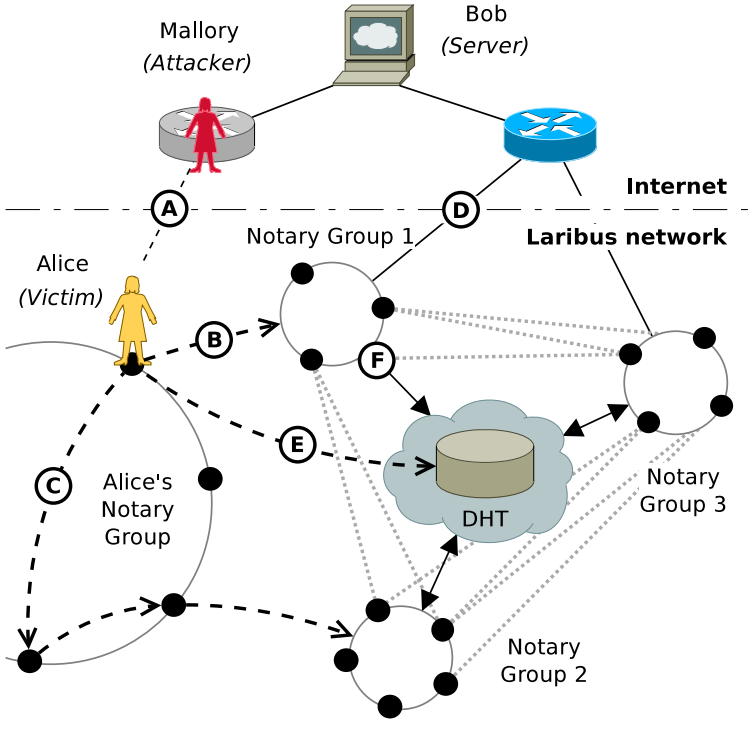

Figure 4 Components of Laribus and their interactions.

much she trusts the notary groups that stored the entries into the DHT) validates the certificate presented to her. If no trusted notary group has seen that particular certificate before, she proceeds with a direct query.
2. Alice contacts a number of notary groups with a direct query asking them to report what certificate server $\mathcal{SRV}$ is offering. The queried group's members connect to $\mathcal{SRV}$ and get $cert_B$, and then sign with a notary signature the fact of having seen $cert_B$ at the moment of fetching.
3. The queried notary groups store the obtained certificates in the DHT.

3.4 User-defined trust

Laribus bases its security upon the social network of its users. Like in the PGP Web of Trust, users are required to define social relationships, i. e., determine to which extent they trust particular users (*user trust level*).

Alice can assign the following *user trust levels* to a friend (Charlie):

Connection: Alice receives and processes Charlie's messages. The consequence is that Charlie gets to know Alice's IP address and connection times.

Direct query: Alice includes Charlie in the list of users she will (try to) send direct queries to. As a result, Charlie may learn about Alice's (group's) surfing habits.

Transitive trust: Alice trusts Charlie's friends to some extent and would receive and forward their messages. In consequence, if Charlie authenticates untrustworthy users, both Alice and Charlie are affected.

Group trust: Alice vouches for Charlie's inclusion into her group (however, Charlie cannot join the group until all members vote for his inclusion). If Charlie is an attacker, he can perform denial-of-service attacks against Alice's group, i. e., by not forwarding messages or not participating in creating notary signatures.

3.5 Security measures

In order to prevent outsiders from forging messages or launching impersonation attacks, Laribus clients make use of public-key cryptography for *message authentication* and identification purposes. Before connecting to the Laribus network for the first time, each client n_i creates a key pair (S_i, P_i, ID_i), where ID_i is a self-signed pseudonym, (ID_i, P_i) is the public key and S_i is the secret key. ID_i does not necessarily have to reveal the actual identity of a user, but friends should be able to discover each other based on these pseudonyms.

As direct queries are quite expensive, they could be used to mount denial of service attacks. Therefore, they cannot be issued anonymously but have to be authenticated by the requestor by a digital signature. This allows notary groups to reject queries received from misbehaving clients or to only accept queries from trusted groups. However, if clients sign direct queries themselves, validation requests can be linked and tracked back to their pseudonym. We will discuss ways to overcome this privacy issue in Section 3.7.1. Moreover, the certificate records stored within the DHT are also signed by the notary groups in order to guarantee their authenticity.

3.6 Availability measures

Given the P2P approach of Laribus, clients cannot be expected to be online at all times. We address this problem with three mechanisms: a ring signature scheme, a threshold signature scheme, and a DHT (i. e., the storage layer, cf. Section 3.2).

The *ring signature scheme* allows Alice to sign a direct query, i. e., to ask another notary group to retrieve a certificate even if Alice is the only (online) member of her group (cf. Section 4.1). With this scheme Alice can prove that she is a member of her group without disclosing her identity (cf. Section 3.7).

The purpose of the *threshold signature scheme*, called *notary signatures* in this paper (cf. Section 4.2.3), is to allow groups to sign replies to direct queries when only a subset of the group members is online. We propose to employ an efficient threshold scheme by Lesueur et al. [50].

The most important availability feature of Laribus is *the DHT*, which serves as a distributed cache. It contains a public *timeline* of the certificates seen by different groups at different locations. Since the DHT is distributed among *all* clients (i. e., independent from the social network and notary group structures), it assures availability of the signed records (i. e., the past validations) of a notary group, even if not a single member of that group is online (cf. Section 4.4). As every record is signed by a notary group before it is put into the DHT, relying on cached information does not introduce additional security issues. As Laribus is meant to complement the existing PKI infrastructure, checking for certificate revocation is out of the scope of the system. Clients have to rely on existing techniques such as certificate revocation lists or OCSP to ensure that they are not relying on stale validation information.

3.7 Privacy-preserving certificate validation

To meet the users' privacy expectations, i. e., to prevent that network nodes will get to know who wants to validate which certificates, we use well-known and approved techniques from the privacy-enhancing technologies research community. Laribus contains privacy-preserving mechanisms for both *direct queries* and *DHT lookups*.

3.7.1 Preserving privacy for direct queries

If Alice asked a notary group to validate a certificate directly (Step B in Figure 4), she could be easily identified by her IP address or by her signature of the direct query. To prevent identification via signatures, we use ring signatures, i. e., other notary groups can validate that *some member* of a certain group has signed a request, but not which member exactly (details follow in Section 4.1).

To obfuscate IP addresses, simple solutions like Convergence's approach to route requests via a low-latency anonymity system (Tor) could be employed. This approach however conflicts with our design goal of a decentralized solution and would be subject to performance problems of such anonymity systems [51]. To this end, we have designed a scheme that utilizes the group structure of our network to build anonymity sets. Before a request is sent to another notary group, it is passed around between the members of Alice's group (cf. Step *C* in Figure 4). As a result, members of other groups will not be able to get to know whether a request was initiated by the requesting host itself or whether the request was sent on behalf of another member of the group.

This mechanism offers privacy towards other groups (e. g., *Group 2* in Figure 4), but it cannot protect Alice from the members of her own group. To this end, we use a layered encryption scheme (cf. [52]). With the layered encryption scheme, Alice chooses *r hops* (i. e., friends) to forward her request (source routing). The layered encryption scheme hides the routing information required by her friends to relay the request. Each hop removes one layer of encryption and, if it is an intermediate node, receives the address of the *next hop* (i. e., the next group member), or the destination address (i. e., the address of the *destination group*) if it is the final hop. Details about the layered encryption scheme used in Laribus follow in Section 4.3.

3.7.2 Preserving privacy of DHT lookups

To meet the users' privacy requirements for DHT lookups, we use a *range query approach* [53]: Alice will not request a specific entry using its key, but instead a set of results by querying for a prefix that refers to a certain subtree within the DHT (cf. Figure 5).

As a result, an attacker that is able to observe this process will only get to know that Alice is interested in one of the records of the subtree, but not in which exactly. The security parameter k determines the size of the result set, which allows to balance performance and privacy (cf. [49] and Section 4.4).

4 Laribus cryptography mechanisms

In this section we will present details about the cryptographic mechanisms of Laribus, especially ring signatures, notary signatures and the layered encryption scheme. We describe how notary group key pairs are generated and (dynamically) shared as well as the structure and data format of the DHT.

4.1 Ring signatures

In Laribus, direct queries have to be signed as notary groups are not required to answer direct queries from any other group (cf. Section 3.5). However, we also want to provide privacy for direct queries (cf. Section 3.7.1). We use ring signatures to solve this problem [54,55]. Ring signatures allow a single member of a group to sign data (m) on behalf of all group members. The signer requires only his own private key (S_s) and the public keys of the other ring members (P_r) as input to create a signature.

To this end, given a signing scheme based on trapdoor one-way permutations (e. g., RSA), the signer constructs a *verification equation* $C_{k,v}(g_1(x_1), g_2(x_2), \ldots, g_r(x_r)) = z$ combining a hash of the data to be signed and the ring's public keys functions with trapdoor $g_1, g_2, \ldots, g_r$. The equation is infeasible to solve for all inputs without inverting any trapdoor function g_i (i. e., not possessing the relative secret key to a key in the ring), therefore, the signature proves the signer's membership in the ring. Only the signer can provide the solution to the equation as the trapdoor is available only to him [54].

4.2 Notary signatures

When Alice asks a notary group to retrieve a certificate on her behalf (*direct query*), the notary group has to sign its answer (*notary signature*) to guarantee authenticity (cf. Section 3.3). In contrast to the case of *ring signatures*, a majority of the notary group must participate to perform the signature. On the other hand, to meet the availability requirements of Laribus, we must ensure that notary signatures can be obtained even if some group members are offline. A suitable solution for this problem is *threshold cryptography* [56].

With *threshold cryptography*, a public/private key pair (P, S) can be jointly generated by n parties. After key generation, each party knows P and holds a share s_i of S, but no party knows S entirely. Signatures can be obtained as long as a *threshold* t of shares is available, i. e., t parties participate. However, t and n must be chosen before the distributed key generation is initialized, and cannot be changed afterwards. As a result, using a *standard* $(t - n)$-threshold scheme directly would require to generate a new key pair every time n changes, i. e., whenever a notary group is resized. Furthermore, the overhead for generating keys increases drastically with the number of nodes [57] and would thus be impractical for our case as we assume that groups may consist of up to about 15 (and at least 3) members.

To overcome this problem, we propose to use a dynamic scheme by Lesueur et al. [50] that allows share merging and splitting. After an initial key generation process (e. g.,

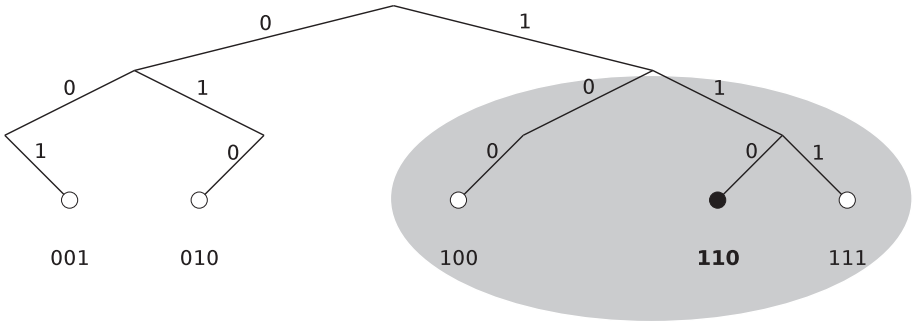

Figure 5 Issuing a range query for the DHT key prefix 1* obfuscates the desired key (110) within the set {100, 110, 111}.

to generate three shares) the number of shares can be adjusted to match the actual number of nodes. Furthermore, its performance is sufficient for larger group sizes, in fact even much larger than required for Laribus [50]. In the following, we will explain details about key generation, dynamic share handling, signing and notary group protection measures against Sybil attackers.

4.2.1 Key generation

To create a key pair for a notary group, we employ Boneh et al.'s efficient method of distributed RSA key pair generation [57]. The protocol allows a set of parties to construct an RSA modulus $N = pq = \sum p_i \sum q_i$ where N is publicly known, and each member N_i only knows about p_i and q_i, not about the factorization of N. To enable sharing of the secret key, they then calculate shares of $d = e^{-1} mod \varphi(N)$ for any given RSA encryption exponent e. Once each of the n group founders has obtained an initial share e_i of the secret d, to which we'll refer as S_G, the following holds: $\sum_{i=1}^{n} e_i = S_G$ [57]. In Laribus, we parameterize Boneh et al.'s method as a $(3-3)$-threshold scheme, i. e., a group will create three shares and all three shares are required to obtain a signature. To meet the security and availability requirements, the number of shares is adjusted with the protocol described in the next section.

4.2.2 Dynamic share handling

To dynamically assign shares to group members, we use the scheme of Lesueur et al. [50]. With this scheme, even though a group is composed of n members, the number E of shares can vary in respect to the number $t' \leq n$ of online members. The scheme allows to specify a fixed *ratio r* of nodes that are required to recover S_G (not a fixed *number* of nodes as in *classical* threshold schemes [50]). To achieve this, shares (e_i) are split and merged and may be replicated on different nodes. Nodes that are assigned the same share e_i compose a *sharing group*.

The share and merge operations require distributed sums and subtractions, since $split(e_i) = (e_{i_1}, e_{i_2})$ and $merge((e_j, e_k)) = e_{jk}$. To prevent an attacker from reconstructing S_G from *old* shares, *old* shares must be rendered useless after each split and merge operation. To this end, if a newly created e_i is mixed with another share e_j, their owners collaboratively calculate $e_i := e_i - \Delta$ and $e_j := e_j + \Delta$, maintaining $\sum_{i=1}^{E} e_i = S_G$. Δ is a chosen random value that hides the original share (cf. [50]).

The ideal number of shares is $E = r \times t'$, with each sharing group composed of $g = \frac{1}{r}$ nodes knowing the same share [50]. We require clients to participate in merge operation only if the number of shares E would not fall under a minimum of $E_{min} = 3$. This way, at least three colluding attackers must be present in a group to recover S_G and forge notary signatures. In practice, when only three group members are online and one member wants to disconnect, the whole group is shut down. The last three shares are assigned to a third of the group members and stored in the DHT encryptedly. Nodes can recover the shares once they re-connect. The notary group ratio is set to $r = \frac{1}{2}$. We will motivate this choice in Section 5.

4.2.3 Obtaining a notary signature

In order to obtain a notary signature for a message D, there must be at least one member per *sharing group* available and willing to sign D, i. e., each share e_i is required for the signature computation. Following the signature protocol (cf. Figure 6), the group signature of D is $D^{S_G}[m]$. With the equality

$$D^{S_G}[m] = D^{(\sum_{i=1}^{E} e_i)}[m] = \left(\prod_{i=1}^{E} D^{e_i}[m] \right)[m]$$

the signature can be calculated collaboratively. One node per *sharing group* signs D with its share, i. e., calculates $s_i = D^{e_i}[m]$ and multiplies the result (mod m) with the result from the previous group (cf. Figure 6 and [50]).

4.2.4 Notary group protection measures

Like any other P2P system, Laribus has to address Sybil attacks that consist of a malicious user joining the network with multiple nodes and *fake* identities in order to

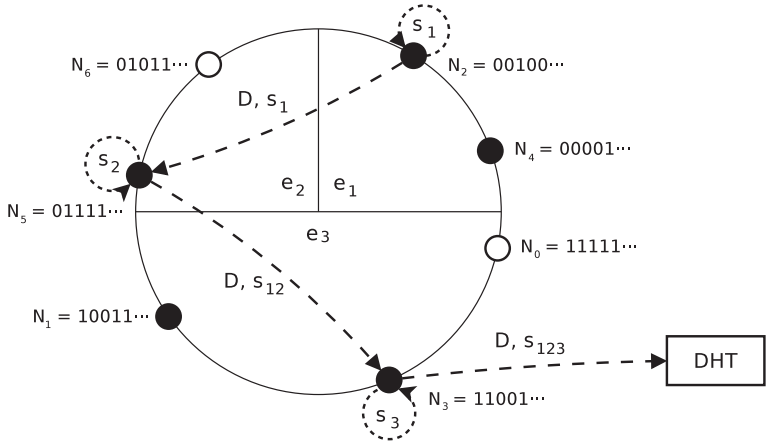

Figure 6 A notary group collaborating for the signature of a message *D*. One node per *sharing group* signs *D* and forwards it along with the calculated incremental product to the next *sharing group*. Shares are stored encryptedly in the DHT for offline nodes (empty circles).

control (large) parts of the network. We plan to employ Gatekeeper [58] as an admission control system to counter Sybil attacks. Gatekeeper uses a *ticket distribution algorithm* to detect attackers and is a suitable choice since, like Laribus, it is based on a social network and does not require users to have a global view of the network.

When a notary group decides about the inclusion of a new candidate (P_i, ID_i), already present members will take into account their own friendship relations (cf. Section 3.4) as well as Gatekeeper. If the new candidate passes both tests, his public key is collaboratively signed by the notary group and stored in the DHT to prove its membership. From that moment on, the candidate's NodeID in the group and DHT is fixed as $sig_{grp,inc}$. Table 1 shows the resulting data structure.

4.3 Layered encryption scheme

As mentioned in Section 3.7.1, direct queries in Laribus are protected by a layered encryption scheme and passed around between members of a notary group to obfuscate IP addresses (cf. Step C in Figure 4). To this end, whenever Alice wants to perform a direct query, she chooses r members of her group at random and establishes a shared secret s_r with each of the hops. The shared secrets are required by the hops involved to derive cryptographic keys for decryption and integrity validation of Alice's query (*request direction*) and for encryption of the corresponding response, i.e., the requested certificate (*response direction*).

To reduce overhead, instead of an iterative channel construction as used in Tor [38], we employ a *none-interactive scheme* that requires only a single message to establish all shared secrets between Alice and the r hops. The scheme is based on the *Sphinx blinding logic* (cf. [59] and [60]): Instead of sending r shared secrets, a single element e of a cyclic group of prime order (satisfying the decisional Diffie-Hellman assumption) is used at each hop to derive the individual secrets. To prevent linkability, the element is *blinded* at each hop with a blinding factor derived from e. Figure 7 shows the operations performed by each hop.

A query consists of e, a header field h, a message authentication code (MAC) and a payload field n. When hop r receives a query, it uses its private key x_r to compute s_r (the shared secret). If s_r was seen before by r, the query is discarded (replay detection). Otherwise, hop r uses a key derivation function (KDF) initialized with s_r to derive $k_{MAC,r}$ and verify the integrity of the query. If the query is valid, another key ($k_{SYM,r}$) and an initialization vector (IV_r) is derived from s_r with KDF in order to decrypt h and n (i. e., to remove one layer of encryption). Further, r derives and stores a key ($k_{Response,r}$) and an initialization vector ($IV_{Response,r}$) from s_r with KDF that will be used

Table 1 The inclusion of a node in a notary group

Row type	Node	Timestamp
Inclusion	(P_i, ID_i)	t_x
	Signature	
	$sig_{grp,inc}$	

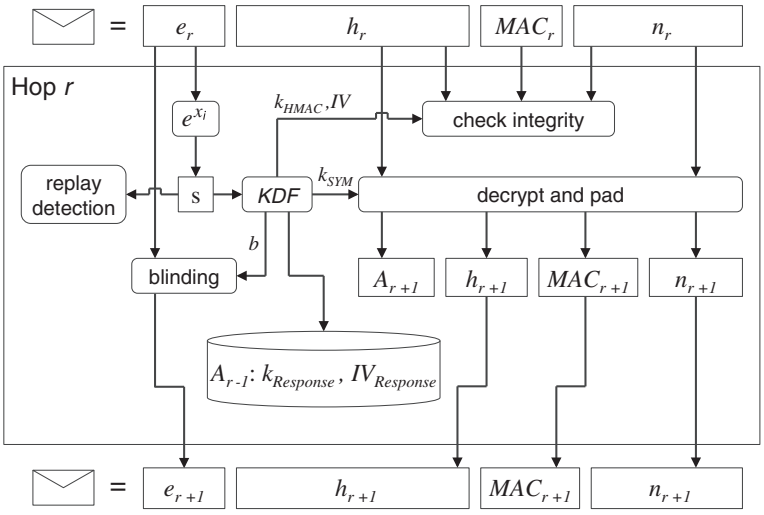

Figure 7 Operations performed by each hop in the layered encryption scheme.

later to encrypt the response for Alice's request (i. e., to add a layer of encryption).

If r is *not* the final hop, the decryption reveals the IP address of the next hop ($A_r + 1$) along with its header field, MAC and payload (h_{r+1}, MAC_{r+1} and n_{r+1}). In this case, hop r will pad the request with random data derived from s_r in order to maintain a constant message length (otherwise, the hops would get to know their position in the path), contact it with e_{r+1} (the blinded element for the next hop) and forward the resulting message to hop $r + 1$. The required blinding factor b is again derived from s_r with *KDF*.

If r *is* the final hop, the plaintext of n will reveal Alice's query along with padding and an identifier of the notary group the query is intended for. In this case, the final hop will retrieve the response from the intended notary group on behalf of Alice, pad it to a constant message length, encrypt it with $k_{Response,r}$ and forward the resulting ciphertext to the previous hop in the path (hop $r - 1$, *response direction*).

Each of the following $r - 1$ reply hops will add another layer of encryption with its corresponding key $k_{Response}$ derived and stored during query processing before. Since all keys, blinding factors etc. are derived from e, and e is chosen by Alice, she is able to create all those items locally and remove the layers of encryption added by the hops.

4.4 Certificate timeline and storage data format

To improve performance and availability of Laribus, clients can retrieve certificates signed by other groups, *certificate validation records* (CVR), from the DHT instead of issuing a direct query (cf. Section 3.3). The DHT stores $\langle key \rightarrow value \rangle$ pairs (cf. Table 2), where multiple values (i. e., CVRs) per key can be returned, e. g., records from different groups or from different times (*certificate timeline*). Laribus employs the Kademlia DHT [49], i. e., keys are truncated to 160 bits (cf. Table 2).

DHT lookups require either a hash of the certificate in question ($cert_X$) or a hash of the domain name ($\mathcal{SRV}_X$, cf. Section 3.3), i. e., the DHT stores two different keys for CVR lookups (cf. Table 2). As mentioned in Section 3.7.2, to meet the users' privacy requirements, clients may request a subtree of the DHT by submitting only a prefix with $160 - k$ bits (*range query*), i. e., they will receive all CVRs with a key that starts with the prefix.

The data format of a CVR is shown in Table 3. The row type *validation* separates *blacklisted* from *whitelisted* CVRs. The *certificate* field identifies the certificates by their SHA_{256} hash. The row type *server* consists of $\mathcal{SRV} = \langle hostname, port \rangle$ values that identify the server which offered $cert_X$. The *timestamp* field states when $cert_X$ was validated. All CVR fields are signed with the respective notary group's signature (sig_{grp}).

Table 2 The stored $cert_X$ validation records in the DHT

DHT keys		
$SHA_{256}(cert_X)[0 \cdots 159]$	$\Longrightarrow$	*CVR*
$SHA_{256}(\boldsymbol{\mathcal{SRV}}_X)[0 \cdots 159]$	$\Longrightarrow$	

Table 3 Certificate validation record storage format

Row type	Certificate	Server	Timestamp
Validation	$SHA_{256}(cert_X)$	$\mathcal{SRV}_X$	t_x
		Signature	
		sig_{grp}	

5 Evaluation

The goal of this section is to provide an initial feasibility study of Laribus. We focus on the computational cost of cryptographic processes as well as availability aspects.

5.1 Cryptographic processes

Laribus makes use of four cryptographic schemes: Distributed key generation (cf. Section 4.2.1), notary signatures (cf. Section 4.2.3), ring signatures (cf. Section 4.1) and a layered encryption scheme (cf. Section 4.3).

The *distributed key generation* scheme is by far the most expensive sub-protocol of Laribus. It requires several rounds of private distributed computation, e. g., to generate an RSA modulus and for biprimality testing. However, key generation is performed only once per group, when a new group is initialized. Given the results of Congos et al. [61], generating a 2,048 bit key that consists of three shares can be expected to be finished in less than 4 min on commodity hardware and with a total network traffic of less than 25 Mbyte. As this process is required only once per group and has no strong real-time requirements, we expect no practical limitations no matter if keys are generated in a LAN or via Internet.

The *notary signature* scheme consists of share assignment and distribution and the signing process itself. The distribution of shares is not expensive, as it only consists of distributed sums and subtractions (cf. Section 4.2.2 and [50]). The signing process requires a single *normal* signature per sharing group, as well as a simple multiplication of the resulting signatures (cf. Section 4.2.3 and [50]). The overhead of both processes seems negligible.

The *ring signature scheme* [55] is practical for our scenario as well: It requires only between n and $2n$ modular multiplications, where n is the group size and can be performed locally by a single node (cf. Section 4.1).

The *layered encryption scheme* of Laribus (cf. Section 4.3) requires essentially $2r$ public key operations to create a message (r is the number of hops) and again two public key operations on each of the r hops (i. e., group members) that forward the message for the Sphinx *blinding logic* and the computation of the shared secrets [59]. It can be used in conjunction with the very efficient Curve25519 elliptic curve library [62]. Results of a recent study [63] indicate that the delay introduced by Sphinx and the relaying of messages should be well below 1 s.

In conclusion, we do not expect performance problems due to cryptographic overhead. The only sub-protocol of Laribus that introduces considerable delay is the distributed key generation, which takes place only once a group is initialized.

5.2 Availability aspects

Whether a group is operational (i. e., it can answer direct queries, for instance) depends on three factors: the group size n, the number of online nodes necessary to recover the group secret (*threshold* t') and the user behavior (i. e., online and offline times). We have performed a simulation-based study to determine adequate values for n and t' and to assess whether these values are indeed practical or not. Source code and configuration files will be made available by the authors on request.

Since Laribus is not deployed yet, we have to estimate the *user behavior* of clients. We model four different types of users: *casual*, *normal*, *office*, and *power users*. *Casual users* connect to the Internet with short session times (on average, 10 min per session) during daytime (between 8:00 and 21:00) and spend 60 min on the Internet per day on average. *Normal users* are connected 5 h per day on average, split into two sessions, one in the morning (around 10:00) and evening (around 19:00). *Office users* connect between 9:00 and 10:00 and end their session between 17:00 and 18:00. *Power users* are either running a server or leaving their computer online most of the day, with 4 h of *downtime* per day on average. User connection times and session durations are sampled from a Gaussian distribution (standard deviation 0.5), except for casual and power users, whose connection times are drawn uniformly from the $[8, 21]$ and $[0, 24]$ hour ranges, respectively. We assume that Laribus is run as a system service, i. e., that clients are available whenever a user is connected to the Internet.

Figures 8, 9, and 10 show the *success percentage* p for different group sizes n, thresholds t' and (mixes of) user types. The *success percentage* is defined as the probability that, whenever at least one node of a notary group is online, the group is operational (i. e., it can perform a notary signature), i. e., at least t' of its nodes are online.

Figures 8 and 9 present the success ratios of groups with uniform type of users, respectively *normal* and *power*

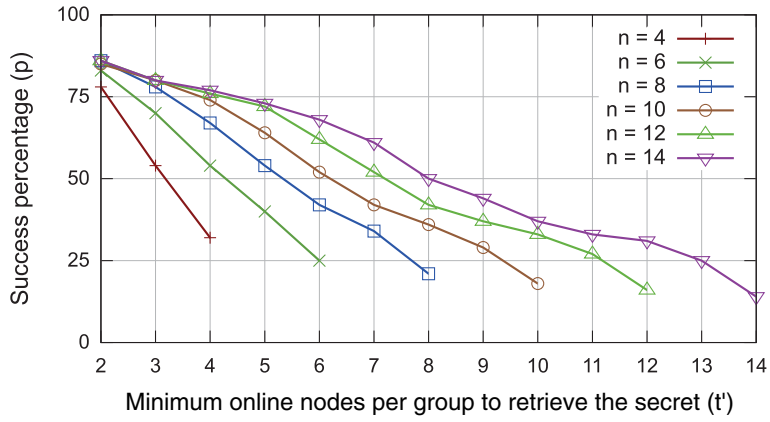

Figure 8 Simulations of online availability for *normal* users.

users. Figure 10 shows results for a more realistic case with mixed user types, i. e., 80% normal and office users and 20% casual and power users. As we can see from Figures 8 and 9, user behavior has a strong influence on the *success percentage*. The results for *normal users* (Figure 8) suggest that for $p = 50\%$ and $n = 6$, $t' \approx 0.7n$ would be a reasonable choice ($\frac{t'}{n} = \frac{4}{6} \approx 0.7$), while the results for *power users* (Figure 9) suggest $t' \approx 0.9n$. The simulation of the most realistic case, the mixed case (Figure 10), suggests $t' \approx 0.5n$. As expected, larger groups can tolerate a bigger fraction of offline nodes for the desired *success percentage* of 50%. For instance, a group with $n = 14$ nodes can tolerate nine offline nodes in the mixed case, which is about 64%, while a group with $n = 6$ nodes can tolerate only about three offline nodes, i. e., about 50% (cf. Figure 10).

In conclusion, the results suggest that $t' \approx \frac{n}{2}$ is a reasonable choice, if $n \geq 6$, i. e., groups should consist of at least 6 members to be operational in 50% of cases.

6 Limitations and discussion

In this section, we will discuss limitations and challenges of Laribus. We focus on the attacker model as well as practical problems, performance tradeoffs, and bootstrapping.

Laribus cannot protect against attackers that control large parts of the Internet. By design, and like any other notary-based MitM detection approach, *Laribus can solely detect local attacks*, i. e., the majority of notaries must not be affected by an attack and must be able to retrieve valid certificates. Attacks that affect a single country (e. g., a repressive regime) or continent can be detected as long as notaries from different parts of the world are available. Moreover, strong attackers may try to block all traffic pertaining to certificate validation. While this is straightforward given static, centralized notaries, Laribus is less vulnerable to blocking due to its decentralized architecture.

A general problem of collaborative distributed certificate validation is that some hosts and especially

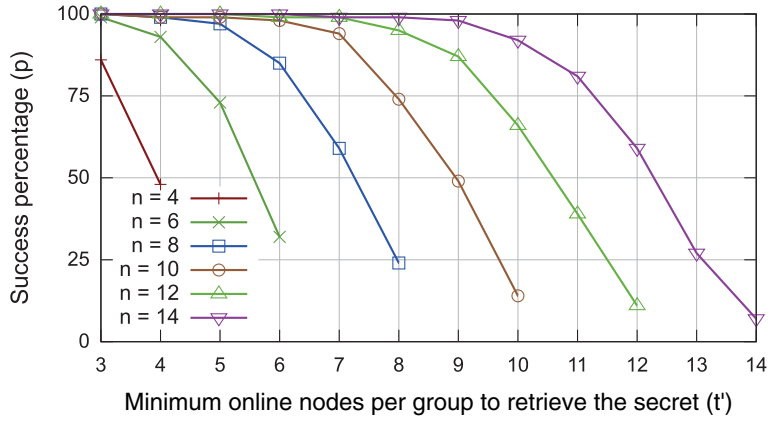

Figure 9 Simulations of online availability for *power* users.

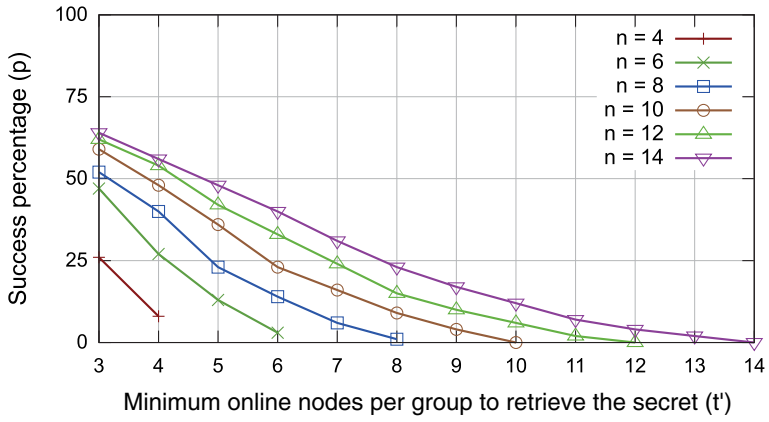

Figure 10 **The simulation of online availability for *mixed* users.**

content distribution networks (CDNs) *use different valid certificates for the same host*. As a result, a 1:1-comparison of certificates is not appropriate. However, since the *timeline* of seen certificates of Laribus allows to store several certificates for a host (cf. Section 4.4), clients can still judge the validity of a certificate by the number of notaries that have seen the same certificate.

Newly issued or renewed certificates cannot be answered with DHT lookups as the *timeline* will not contain any records for them. In this case direct queries (cf. Section 3.3) have to be issued, i. e., enough notaries that the client trusts have to be available.

The results of our initial feasibility study suggest that groups should consist of at least six members to be operational in 50% of cases (cf. Section 5.2). Composing a group of at least six friends should not be a problem in practice. An average 50% availability of a notary group should be practical as well, especially since the storage and caching mechanisms of the DHT do allow to access the *timeline* of the certificates seen by different groups even if all members of those groups are offline. Further, previous studies indicate that the popularity of web hosts follows a power-law distribution [64], i. e., a small set of popular hosts is responsible for the vast majority of all requests, while the *long tail* of remaining hosts is requested rarely. Thus, we expect that the DHT's cache hit ratio will be high; direct queries will not be needed in most cases. Our analysis of the cryptographic processes of Laribus showed that *cryptographic overhead is moderate*, except for distributed key generation which is required only during initialization (cf. Section 5.1).

In conclusion, results for both availability and performance are quite promising. They indicate that a Laribus deployment would indeed be practical. However, future work should focus on analyzing the cache hit ratio using different caching strategies (such as 'most recently used' or based on a time-to-live value like in DNS) and the resulting user-perceived latencies. The results would be of particular interest for the parameterization of Laribus, especially the security parameter k of the range query protection mechanism (cf. Section 3.7.2).

To offer adequate performance and availability during the initial deployment of Laribus, we suggest to *bootstrap the network* with a set of dedicated servers operated by different universities around the world until enough users participate. In fact, a permanent operation of these servers could be an option as well as it would allow to combine the benefits of P2P (especially blocking resistance) and client-server solutions and thus further increase the attractiveness of Laribus. Furthermore, snapshots of the *timeline* (including group signatures) could be mirrored and made publicly available for all users on the Internet. This resembles the approach of *Sovereign Keys* (cf. Section 2 and [32]) and would also help to reduce lookup latencies as well as the size of the DHT.

7 Conclusions

In this paper, we presented Laribus, a social peer-to-peer network that addresses the problem of man-in-the-middle attacks on SSL. Laribus integrates approved techniques from current state-of-the-art solutions and combines them with well-known proposals from the privacy enhancing technologies research community to improve both privacy and availability.

To the best of our knowledge, Laribus is the first fully distributed solution for the detection of fake SSL certificates. It does not require cooperation of website owners and is blocking-resistant due to its decentralized architecture. Further, clients do not have to trust a central notary service. The Laribus architecture is fundamentally different from previous solutions since users can model trust relationships that reflect their social relationships and may

create virtual notaries that consist of their friends. The resulting groups of friends are utilized both as anonymity sets and to increase availability.

Our initial evaluation results are promising and suggest that Laribus is feasible. Future work will focus on an implementation of its components as well as an in-depth analysis of the effectiveness of caching and user-perceived latencies in order to prepare the practical deployment of Laribus.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
The authors are grateful to the anonymous reviewers for their constructive feedback and insightful comments. A preliminary version of this article has been published in the ARES 2013 proceedings [13].

Received: 15 March 2014 Accepted: 10 July 2014
Published online: 18 February 2015

References

1. T Dierks, E Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2 RFC 5246*. (IETF, Fremont, CA, USA, 2008)
2. A Freier, P Karlton, P Kocher, *The Secure Sockets Layer (SSL) Protocol Version 3.0 RFC 6101*. (IETF, Fremont, CA, USA, 2011)
3. E Rescorla, *SSL and TLS: Designing and Building Secure Systems*. (Addison-Wesley, Boston, Toronto, Paris, 2001)
4. J Rizzo, T Duong, Here Come The XOR Ninjas. Unpublished manuscript (2011). http://netifera.com/research/beast/beast_DRAFT_0621.pdf
5. Y Gluck, N Harris, A Prado, BREACH: Reviving THE CRIME Attack. Unpublished manuscript (2013). http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf
6. NJ AlFardan, KG Paterson, in *IEEE Symposium on Security and Privacy (S&P 2013), Berkeley, CA, USA, May 19–22, 2013, Proceedings*. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols (IEEE New York, NY, USA, 2013), pp. 526–540
7. C Meyer, J Schwenk, Lessons Learned From Previous SSL/TLS Attacks – A Brief Chronology Of Attacks And Weaknesses. Cryptology ePrint Archive, Report 2013/049 (2013). http://eprint.iacr.org/2013/049/20130201:021008
8. C Meyer, J Schwenk, in *Information Security Applications – 14th International Workshop (WISA 2013), Jeju Island, Korea, August 19–21, 2013, Revised Selected Papers,* Lecture Notes in Computer Science, ed. by Y Kim, H Lee, and A Perrig. SoK: Lessons Learned from SSL/TLS Attacks, vol. 8267 (Springer Berlin Heidelberg, 2013), pp. 189–209
9. D Cooper, S Santesson, S Farrell, S Boeyen, R Housley, W Polk, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280*. (IETF, Fremont, CA, USA, 2008)
10. Electronic Frontier Foundation, EFF SSL Observatory. https://www.eff.org/observatory. accessed 2014-03-14
11. C Soghoian, S Stamm, in *Financial Cryptography and Data Security – 15th International Conference (FC 2011), Gros Islet, St. Lucia, February 28 – March 4, 2011, Revised Selected Papers.* Lecture Notes in Computer Science, ed. by G Danezis. Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL, vol. 7035 (Springer Berlin Heidelberg, 2012), pp. 250–259
12. N Leavitt, Internet Security Under Attack: The Undermining of Digital Certificates. IEEE Computer. **44**(12), 17–20 (2011)
13. A Micheloni, K-P Fuchs, D Herrmann, H Federrath, in *International Conference on Availability, Reliability and Security (ARES 2013), Regensburg, Germany, September 2–6, 2013, Proceedings*. Laribus: Privacy-Preserving Detection of Fake SSL Certificates with a Social P2P Notary Network (IEEE New York, NY, USA, 2013), pp. 1–10
14. J Clark, PC van Oorschot, in *IEEE Symposium on Security and Privacy (S&P 2013), Berkeley, CA, USA, May 19–22, 2013, Proceedings*. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements (IEEE New York, NY, USA, 2013), pp. 511–525
15. H Tschofenig, E Lear, *Evolving the Web Public Key Infrastructure. Internet Draft*. (IETF, Fremont, CA, USA, 2013). http://tools.ietf.org/html/draft-tschofenig-iab-webpki-evolution-01
16. J Kasten, E Wustrow, JA Halderman, in *International Conference on Financial Cryptography and Data Security (FC 2013), Okinawa, Japan, April 1–5, 2013, Revised Selected Papers*. Lecture Notes in Computer Science, ed. by A-R Sadeghi. Cage: Taming Certificate Authorities by Inferring Restricted Scopes, vol. 7859 (Springer Berlin Heidelberg, 2013), pp. 329–337
17. H Perl, S Fahl, M Smith, in *18th International Conference on Financial Cryptography and Data Security (FC 2014), Barbados, March 3—7, 2014, Proceedings*. You Won't Be Needing These Any More: On Removing Unused Certificates From Trust Stores, (2014)
18. J Sunshine, S Egelman, H Almuhimedi, N Atri, LF Cranor, in *USENIX Security Symposium, Montreal, Canada, August 10–14, 2009, Proceedings*. Crying Wolf: An Empirical Study of SSL Warning Effectiveness (USENIX Association Berkeley, CA, USA, 2009), pp. 399–416
19. C Evans, C Palmer, R Sleevi, *Public Key Pinning Extension for HTTP. Internet Draft*. (IETF, Fremont, CA, USA, 2014). http://tools.ietf.org/html/draft-ietf-websec-key-pinning-11
20. M Marlinspike, T Perrin, *Trust Assertions for Certificate Keys. Internet Draft*. (IETF, Fremont, CA, USA, 2013). http://tools.ietf.org/html/draft-perrin-tls-tack-02
21. P Hoffman, J Schlyter, *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698*. (IETF, Fremont, CA, USA, 2012)
22. R Arends, R Austein, M Larson, D Massey, S Rose, *DNS Security Introduction and Requirements RFC 4033*. (IETF, Fremont, CA, USA, 2005)
23. E Osterweil, B Kaliski, M Larson, D McPherson, in *Workshop on Securing and Trusting Internet Names (SATIN 2012), March 22–23, 2012, Teddington, UK, Proceedings*. Reducing the X.509 Attack Surface with DNSSEC's DANE, (2012). http://conferences.npl.co.uk/satin/papers/satin2012-Osterweil.pdf
24. H Yang, E Osterweil, D Massey, S Lu, L Zhang, Deploying Cryptography in Internet-scale Systems: A Case Study on DNSSEC. Dependable and Secure Computing, IEEE Transactions on. **8**(5), 656–669 (2011)
25. VeriSign, Licensing Verisign Certificates. Technical Report (2005). http://www.verisign.com/static/001496.pdf
26. P Hallam-Baker, R Stradling, *DNS Certification Authority Authorization (CAA) Resource Record. RFC 6844*. (IETF, Fremont, CA, USA, 2013)
27. W Koch, M Brinkmann, STEED – Usable End-to-End Encryption. Unpublished manuscript (2011). http://g10code.com/docs/steed-usable-e2ee.pdf
28. GX Toth, T Vlieg, Public Key Pinning for TLS – Using a Trust on First Use Model. Technical report, University of Amsterdam (2013). http://www.delaat.net/rp/2012-2013/p56/report.pdf
29. Certificate Patrol Homepage. http://tg-x.net/code/certpatrol. accessed 2014-03-14
30. I Dacosta, M Ahamad, P Traynor, in *European Symposium on Research in Computer Security (ESORICS 2012), Pisa, Italy, September 10–12, 2012, Proceedings.* Lecture Notes in Computer Science, ed. by S Foresti, M Yung, and F Martinelli. Trust No One Else: Detecting MITM Attacks against SSL/TLS without Third-Parties, vol. 7459 (Springer Berlin Heidelberg, 2012), pp. 199–216
31. The Monkeysphere Project. http://web.monkeysphere.info/. accessed 2014-03-14
32. The Sovereign Keys Project. https://www.eff.org/sovereign-keys. accessed 2014-03-14
33. B Laurie, A Langley, E Kasper, *Certificate Transparency. RFC 6962*. (IETF, Fremont, CA, USA, 2013)
34. TH-J Kim, L-S Huang, A Perrig, C Jackson, V Gligor, Transparent Key Integrity (TKI): A Proposal for a Public-Key Validation Infrastructure. Technical Report CMU-CyLab-12-016, Carnegie Mellon University (July 2012). https://www.cylab.cmu.edu/files/pdfs/tech_reports/CMUCyLab12016.pdf
35. D Wendlandt, DG Andersen, A Perrig, in *USENIX Annual Technical Conference, Boston, MA, USA, June 22–27, 2008, Proceedings*. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing (USENIX Berkeley, CA, USA, 2008), pp. 321–334
36. The Convergence Project. http://convergence.io/. accessed 2014-03-14
37. J Wilberding, A Yates, M Sherr, W Zhou, in *Annual Computer Security Applications Conference (ACSAC '13), New Orleans, LA, USA, December 9-13, 2013, Proceedings*, ed. by CNP Jr. Validating Web Content with Senser (ACM New York, NY, USA, 2013), pp. 339–348

38. R Dingledine, N Mathewson, PF Syverson, in *USENIX Security Symposium, August 9–13, 2004, San Diego, CA, USA, Proceedings*. Tor: The Second-Generation Onion Router (USENIX Berkeley, CA, USA, 2004), pp. 303–320
39. M Alicherry, AD Keromytis, in *IEEE Symposium on Computers and Communications (ISCC 2009), July 5–8, Sousse, Tunisia, Proceedings*. DoubleCheck: Multi-path Verification Against Man-in-the-Middle Attacks (IEEE New York, NY, USA, 2009), pp. 557–563
40. K Engert, DetecTor (version 1.02 2013). http://detector.io/DetecTor.pdf accessed 2014-03-14
41. D McCoy, KS Bauer, D Grunwald, T Kohno, DC Sicker, in *Privacy Enhancing Technologies, 8th International Symposium (PETS 2008), Leuven, Belgium, July 23–25, 2008, Proceedings*. Lecture Notes in Computer Science, ed. by N Borisov, I Goldberg. Shining Light in Dark Places: Understanding the Tor Network, vol. 5134 (Springer Berlin Heidelberg, 2008), pp. 63–76
42. P Winter, R Köwer, M Mulazzani, M Huber, S Schrittwieser, S Lindskog, E Weippl, in *Privacy Enhancing Technologies, 14th International Symposium (PETS 2014), Amsterdam, The Netherlands, July 16–18, 2014, Proceedings*. Lecture Notes in Computer Science, ed. by E De Cristofaro, SJ Murdoch. Spoiled Onions: Exposing Malicious Tor Exit Relays, vol. 8555 (Springer Berlin Heidelberg, 2014), pp. 304–331
43. B Amann, M Vallentin, S Hall, R Sommer, *Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. Technical Report TR-12-014*. (International Computer Science Institute, Berkeley, CA, USA, 2012). http://www.icsi.berkeley.edu/pubs/techreports/ICSI_TR-12-014.pdf
44. R Holz, T Riedmaier, N Kammenhuber, G Carle, in *European Symposium on Research in Computer Security (ESORICS 2012), Pisa, Italy, September 10–12, 2012, Proceedings*. Lecture Notes in Computer Science, ed. by S Foresti, M Yung, and F Martinelli. X.509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-Middle, vol. 7459 (Springer Berlin Heidelberg, 2012), pp. 217–234
45. H-M Sun, W-H Chang, S-Y Chang, Y-H Lin, in *Cryptology and Network Security, 8th International Conference (CANS 2009), Kanazawa, Japan, December 12–14, 2009, Proceedings*. Lecture Notes in Computer Science, ed. by JA Garay, A Miyaji, and A Otsuka. DepenDNS: Dependable Mechanism against DNS Cache Poisoning, vol. 5888 (Springer Berlin Heidelberg, 2009), pp. 174–188
46. K Park, VS Pai, LL Peterson, Z Wang, in *Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6–8, 2004, Proceedings*. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups (USENIX Association Berkeley, CA, USA, 2004), pp. 199–214
47. L Poole, VS Pai, in *USENIX Annual Technical Conference, Boston, MA, USA, June 22–27, 2008, Proceedings*. ConfiDNS: Leveraging Scale and History to Detect Compromise (USENIX Berkeley, CA, USA, 2008), pp. 99–112
48. L Yuan, K Kant, P Mohapatra, C-N Chuah, in *International Conference on Communications (ICC 2006), Istanbul, Turkey, 11-15 June 2006, Proceedings*. DoX: A Peer-to-Peer Antidote for DNS Cache Poisoning Attacks (IEEE New York, NY, USA, 2006), pp. 2345–2350
49. P Maymounkov, D Mazières, in *Peer-to-Peer Systems, First International Workshop (IPTPS 2002), Cambridge, MA, USA, March 7–8, 2002, Revised Papers*. Lecture Notes in Computer Science. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, vol. 2429 (Springer Berlin Heidelberg, 2002), pp. 53–65
50. F Lesueur, Mé, VVT Tong, in *Resilient Networks and Services, Second International Conference on Autonomous Infrastructure, Management and Security (AIMS 2008), Bremen, Germany, July 1–3, 2008, Proceedings*. Lecture Notes in Computer Science. A Distributed Certification System for Structured P2P Networks, vol. 5127 (Springer Berlin Heidelberg, 2008), pp. 40–52
51. R Dingledine, SJ Murdoch, Performance Improvements on Tor, or, Why Tor is Slow and What We're Going to Do About It. Unpublished manuscript (2009). https://svn.torproject.org/svn/projects/roadmaps/2009-03-11-performance.pdf
52. D Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM. **24**(2) (1981)
53. Y Lu, G Tsudik, in *International Conference on Peer-to-Peer Computing (P2P 2010), Delft, The Netherlands, 25–27 August 2010, Proceedings*. Towards Plugging Privacy Leaks in the Domain Name System (IEEE New York, NY, USA, 2010), pp. 1–10
54. RL Rivest, A Shamir, Y Tauman, in *Advances in Cryptology (ASIACRYPT 2001), 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9–13, 2001, Proceedings*. Lecture Notes in Computer Science, ed. by C Boyd. How to Leak a Secret, vol. 2248 (Springer Berlin Heidelberg, 2001), pp. 552–565
55. RL Rivest, A Shamir, Y Tauman, in *Theoretical Computer Science, Essays in Memory of Shimon Even*. Lecture Notes in Computer Science, ed. by O Goldreich, AL Rosenberg, and AL Selman. How to Leak a Secret: Theory and Applications of Ring Signatures, vol. 3895, vol. 3895 (Springer Berlin Heidelberg, 2006), pp. 164–186
56. A Shamir, How to share a secret. Commun. ACM. **22**(11), 612–613 (1979)
57. D Boneh, MK Franklin, in *Advances in Cryptology (CRYPTO '97), 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 1997, Proceedings*. Lecture Notes in Computer Science. Efficient Generation of Shared RSA Keys (Extended Abstract), vol. 1294 (Springer Berlin Heidelberg, 1997), pp. 425–439
58. DN Tran, J Li, L Subramanian, SSM Chow, in *International Conference on Computer Communications (INFOCOM 2011), Joint Conference of the IEEE Computer and Communications Societies, 10–15 April 2011, Shanghai, China, Proceedings*. Optimal Sybil-Resilient Node Admission Control (IEEE New York, NY, USA, 2011), pp. 3218–3226
59. G Danezis, I Goldberg, in *IEEE Symposium on Security and Privacy (S&P 2009), 17–20 May 2009, Oakland, California, USA, Proceedings*. Sphinx: A Compact and Provably Secure Mix Format (IEEE New York, NY, USA, 2009), pp. 269–282
60. A Kate, I Goldberg, in *Financial Cryptography and Data Security, 14th International Conference (FC 2010), Tenerife, Canary Islands, January 25–28, 2010, Revised Selected Papers*. Lecture Notes in Computer Science, ed. by R Sion. Using Sphinx to Improve Onion Routing Circuit Construction, vol. 6052 (Springer Berlin Heidelberg, 2010), pp. 359–366
61. T Congos, F Lesueur, in *Workshop on Security and Trust Management (STM 2009), Proceedings. Electronic Notes in Theoretical Computer Science*. Experimenting with Distributed Generation of RSA Keys (Elsevier Saint-Malo, France, 2009)
62. DJ Bernstein, in *Public Key Cryptography – 9th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2006), New York, NY, USA, April 24–26, 2006, Proceedings*. Lecture Notes in Computer Science. Curve25519: New Diffie-Hellman Speed Records, vol. 3958 (Springer Berlin Heidelberg, 2006), pp. 207–228
63. K-P Fuchs, D Herrmann, H Federrath, in *European Symposium on Research in Computer Security (ESORICS 2012), Pisa, Italy, September 10–12, 2012, Proceedings*. Lecture Notes in Computer Science, ed. by S Foresti, M Yung, and F Martinelli. Introducing the gMix Open Source Framework for Mix Implementations, vol. 7459 (Springer Berlin Heidelberg, 2012), pp. 487–504
64. J Jung, E Sit, H Balakrishnan, R Morris, DNS Performance and the Effectiveness of Caching. Networking, IEEE/ACM Transactions on. **10**(5), 589–603 (2002)