Anonymous Webs of Trust

Conference Paper · July 2010

DOI: 10.1007/978-3-642-14527-8_8 · Source: DBLP

Matteo Maffei
Universität des Saarlandes

Kim Pecina
Universität des Saarlandes

Anonymous Webs of Trust

Michael Backes[1,2], Stefan Lorenz[1], Matteo Maffei[1], and Kim Pecina[1]

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

Abstract. Webs of trust constitute a decentralized infrastructure for establishing the authenticity of the binding between public keys and users and, more generally, trust relationships among users. This paper introduces the concept of anonymous webs of trust – an extension of webs of trust where users can authenticate messages and determine each other's trust level without compromising their anonymity. Our framework comprises a novel cryptographic protocol based on zero-knowledge proofs, a symbolic abstraction and formal verification of our protocol, and a prototypical implementation based on the OpenPGP standard. The framework is capable of dealing with various core and optional features of common webs of trust, such as key attributes, key expiration dates, existence of multiple certificate chains, and trust measures between different users.

1 Introduction

Over the last years, the Web has evolved into the premium forum for freely disseminating and collecting data, information, and opinions. Not all information providers, however, are willing to reveal their true identity: For instance, some may want to present their opinions anonymously to avoid associations with their race, ethnic background, or other sensitive characteristics. Furthermore, people seeking sensitive information may want to remain anonymous to avoid being stigmatized or other negative repercussions. The ability to anonymously exchange information, and hence the inability of users to identify the information providers and to determine their credibility, raises serious concerns about the reliability of exchanged information. Ideally, one would like to have a mechanism for assigning trust levels to users, allowing them to anonymously exchange data and, at the same time, certifying the trust level of the information provider.

Webs of trust. Webs of trust (WOT) constitute a well-established approach to bind public keys to their owners and, more generally, to establish trust relationships among users in a decentralized manner: Each participant decides which public keys are considered trustworthy. This trust is expressed by signing the trustworthy public keys along with a set of user and key attributes (e.g., user name and key expiration date). These certificates can be chained in order to express longer trust relationships:[3] For instance, the certificate chain

[3] In the OpenPGP standard [15], trust relationships may be transitive and their validity is ruled by *trust signatures*, which we describe in Appendix A. In our setting, the

$$\mathsf{sig}((\mathsf{pk}_1, \mathcal{A}_1), \mathsf{sk}_2), \mathsf{sig}((\mathsf{pk}_2, \mathcal{A}_2), \mathsf{sk}_3)$$

says that the owner of pk_3 has certified the binding between the public key pk_2 and the set $\mathcal{A}_2$ of attributes, and the owner of pk_2 has certified the binding between pk_1 and $\mathcal{A}_1$. Such certificate chains are a salient technique for expressing transitive trust relationships, e.g., to use webs of trust to implement friendship relations in social networks such as Facebook, where transitive friendship relations are common; in this example, the owner of pk_1 would be a friend of a friend of the owner of pk_3.

After receiving a signature on message m that can be verified using pk_1, the owner of pk_3 knows that m comes from a user of trust level 2 bound to the attributes $\mathcal{A}_1$.[4] Hence for authenticating a message in the context of a WOT, the sender has to find a chain of certificates starting with a certificate released by the intended recipient and ending with a certificate for the sender's key.

Our contributions. In this work we introduce the concept of *anonymous webs of trust* – an extension of webs of trust that allows users to authenticate messages and determine each other's trust level without compromising their anonymity. Our framework comprises:

- a *cryptographic protocol* based on the Camenisch-Lysyanskaya signature scheme [16] and a novel zero-knowledge proof[5] that allows users to efficiently prove the existence of certificate chains without compromising user anonymity. For instance, given the certificate chain $\mathsf{sig}((\mathsf{pk}_1, \mathcal{A}_1), \mathsf{sk}_2), \mathsf{sig}((\mathsf{pk}_2, \mathcal{A}_2), \mathsf{sk}_3)$ and a message m that the owner of pk_1 wants to authenticate with the owner of pk_3, our protocol allows the owner of pk_1 to prove a statement of the form "there exist certificates C_1, C_2, a signature S, keys K_1, K_2, and attributes A_1, A_2 such that (*i*) C_1 is a certificate for (K_1, A_1) that can be verified with key K_2, (*ii*) C_2 is a certificate for (K_2, A_2) that can be verified with key pk_3, and (*iii*) S is a signature on m that can be verified with K_1". This statement reveals only the length of the chain, i.e., the trust level of the sender, the authenticated message m,

trust relationship is more sophisticated and, in fact, it is parametrized by a number of factors including the length of the chain (i.e., the longer the chain, the smaller the conveyed trust). This allows us to accommodate fine-grained trust models, as discussed in Section 4.

[4] For the sake of simplicity, we identify the trust level of a certificate chain with its length here. In Section 4, we will consider the more sophisticated trust measure proposed in [21]. We refer the interested reader to [43,3,40,42,21,19,54,4,39] for additional trust models.

[5] A zero-knowledge proof combines two seemingly contradictory properties. First, it is a proof of a statement that cannot be forged, i.e., it is impossible, or at least computationally infeasible, to produce a zero-knowledge proof of a wrong statement. Second, a zero-knowledge proof does not reveal any information besides the bare fact that the statement is valid [36]. A non-interactive zero-knowledge proof is a zero-knowledge protocol consisting of one message sent by the prover to the verifier.

and the public key pk_3 of the intended recipient. We provide a prototypical implementation of our protocol as an extension of the OpenPGP standard. The tool is freely available at [7].

- a number of *extensions* of our protocol to achieve fine-grained anonymity and trust properties. In some situations, a controlled release of additional information is desired or even required, e.g., proving that the keys involved in a chain have not expired. We propose variants of our zero-knowledge proof that allow for selectively revealing additional properties of the certificate chains, such as the validity of the keys with respect to their expiration date, the existence of multiple certificate chains, and the trust level that the certificate chains are assigned according to a realistic trust model. These extensions demonstrate the expressiveness and generality of our approach.

  The potential application scenarios of our protocol include distributed social networks, where people may want to share opinions or information in an anonymous fashion while being able to prove their trust relationships, applications for anonymous message exchange, and services for anonymous yet trustworthy reports or reviews.
- a *symbolic abstraction* and a *formal verification* of our protocol. We specify our protocol in the applied pi-calculus [2], and we formalize the trust property as an authorization policy and the anonymity property as an observational equivalence relation. We consider a strong adversarial setting where the attacker has the control over the topology of the web of trust, some of the protocol parties, and the certificate chains proven in zero-knowledge by honest parties. Security properties are verified using ProVerif [12], an automated theorem prover based on Horn clause resolution that provides security proofs for an unbounded number of protocol sessions and protocol parties.

Related work. Although the setting is different, our approach may at a first glance resemble the delegatable anonymous credential scheme [10]. This protocol relies on an *interactive* protocol between *each* pair of users along the certificate chain. In contrast, our protocol is fully non-interactive, and provers do not need any interactions with other principals except for the intended recipient. In addition, our approach allows the prover to selectively reveal partial information on attributes in the certificate chain, which is crucial to achieve anonymity in realistic trust models without compromising their expressiveness.

Group signature schemes [25,49,6,11] provide a method for allowing a member of a group to anonymously sign a message on behalf of the group. In contrast to our approach, these schemes require the presence of a group manager; moreover, two users in the same group are completely interchangeable. A similar argument holds for HIBE/HIBS schemes [34,13], where anonymity could be obtained by replacing user identifiers with generic anonymous attributes.

Ring signature schemes [45,38,41] are similar to group signatures but do not require a group manager. As for group signatures, two users in the same group are completely interchangeable. It would be interesting, nevertheless, to explore the usage of ring signature schemes to achieve k-anonymity in webs of trust.

Social networks constitute a particularly promising application scenario for our protocol; we thus briefly relate our approach to recent works on privacy and anonymity in social networks. The (somewhat) orthogonal problem of creating encrypted data that can be read by people who are n degrees away in a social network has been recently addressed [31]. Several techniques have been proposed to keep the social network graph private while enforcing access control policies based on trust degrees [28,27,53]. In contrast to our approach, the proposed protocols are interactive, similar to the delegatable anonymous credential scheme [10]. In other works, trust relationships are instead assumed to be public, e.g., [46,5,20]. Our approach does not put any constraints on the way certificates are distributed (for instance, they could be exchanged by private communication). We just assume that the prover can retrieve the certificates composing the chain proven in zero-knowledge. In the specific context of webs of trust such as GnuPG [50], public keys and attached certificates are uploaded on key servers and are thus publicly available. Finally, the recently proposed Lockr protocol [52] achieves access control and anonymity in social networks and file-sharing applications, such as Flickr and BitTorrent. Lockr provides weaker anonymity guarantees compared to our framework, since the prover has to reveal her identity to the verifier; moreover, Lockr does not support certificate chains but only direct trust relationships.

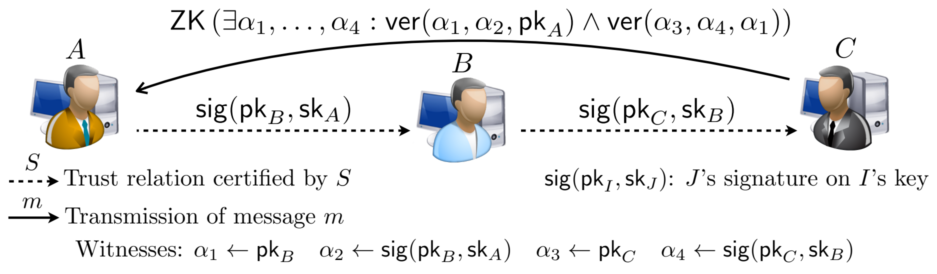
Outline of the paper. Section 2 introduces the notion of anonymous webs of trust and provides a high-level overview of our protocol. Section 3 describes the cryptographic setup, conducts a complexity analysis, and describes the implementation. Section 4 presents extensions of our protocol that accommodate some advanced properties of webs of trust. Section 5 proposes a symbolic abstraction of our protocol and conducts a formal security analysis. Section 6 concludes and gives directions of future research.

2 Anonymous Webs of Trust

In this section, we introduce the notion of anonymous webs of trust and we give an overview of our protocol.

A web of trust is a decentralized public-key infrastructure. Each user u holds a public key $\mathsf{pk_u}$ and a secret key $\mathsf{sk_u}$. Trust is distributed via certificates. User u expresses her belief that a given public key $\mathsf{pk_v}$ actually belongs to user v by signing $\mathsf{pk_v}$ along with a set $\mathcal{A}_\mathsf{v}$ of user and key attributes. Hence, certificates establish the relation between public keys and users and, depending on the applications, they can also be used to witness specific trust relationships between users. These certificates are attached to the signed public key and uploaded all together onto key servers. Every user having access to such a server can participate in the web of trust.

Trust into public keys not directly signed by a user is established using *certificate chains*. A certificate chain from A to B consists of all the certificates that link $(\mathsf{pk_A}, \mathcal{A}_\mathsf{A})$ to $(\mathsf{pk_B}, \mathcal{A}_\mathsf{B})$, thus establishing a trust relation between those keys.



$\mathsf{ZK}\,(\exists \alpha_1, \dots, \alpha_4 : \mathsf{ver}(\alpha_1, \alpha_2, \mathsf{pk}_A) \wedge \mathsf{ver}(\alpha_3, \alpha_4, \alpha_1))$
A
B
C
$\mathsf{sig}(\mathsf{pk}_B, \mathsf{sk}_A)$
$\mathsf{sig}(\mathsf{pk}_C, \mathsf{sk}_B)$
$\xrightarrow{S}$ Trust relation certified by S
$\mathsf{sig}(\mathsf{pk}_I, \mathsf{sk}_J)$: J's signature on I's key
$\xrightarrow{m}$ Transmission of message m
Witnesses: $\alpha_1 \leftarrow \mathsf{pk}_B$ $\alpha_2 \leftarrow \mathsf{sig}(\mathsf{pk}_B, \mathsf{sk}_A)$ $\alpha_3 \leftarrow \mathsf{pk}_C$ $\alpha_4 \leftarrow \mathsf{sig}(\mathsf{pk}_C, \mathsf{sk}_B)$

Fig. 1. Protocol for anonymous proof of a certificate chain of length 2

Definition 1 (Certificate Chain). *A* certificate chain *or simply* chain *from* $(\mathsf{pk}_1, \mathcal{A}_1)$ *to* $(\mathsf{pk}_\ell, \mathcal{A}_\ell)$ *is a sequence of certificates* $\mathcal{C} = (\mathsf{C}_1, ..., \mathsf{C}_{\ell-1})$ *of length* $\ell - 1$*, where* $\mathsf{C}_i = \mathsf{sig}((\mathsf{pk}_{i+1}, \mathcal{A}_{i+1}), \mathsf{sk}_i)$ *and* $\ell \geq 2$*. We say that* $(\mathsf{pk}_\ell, \mathcal{A}_\ell)$ *has* trust level $\ell - 1$*. We assume to know the binding between* sk_1 *and* $(\mathsf{pk}_1, \mathcal{A}_1)$*, which can be captured by an additional self-generated certificate* $\mathsf{sig}((\mathcal{A}_1, \mathsf{pk}_1), \mathsf{sk}_1)$*.*

The fundamental idea of our approach is to provide anonymity in webs of trust by deploying zero-knowledge proofs to demonstrate the existence of valid certificate chains without revealing any information that might compromise the anonymity of users. We consider a setting where users want to anonymously exchange messages, yet guaranteeing the receiver the trust level of the sender.

For the sake of simplicity, we initially focus on certificates on public keys without attributes. In Section 4, we will extend our zero-knowledge proof scheme to certificates binding a key to a set of attributes, and subsequently show how to selectively hide some of them while revealing the others.

In order to authenticate a message m with the owner of pk_1, the owner of pk_ℓ has to retrieve a certificate chain from pk_1 to pk_ℓ and to prove in zero-knowledge the existence of this chain as well as the knowledge of a signature on message m done with the signing key corresponding to pk_ℓ. Notice that the signature cannot be sent in plain, since this would compromise the anonymity of the sender. If we denote by $\mathsf{ver}(\mathsf{m}, \mathsf{C}, \mathsf{pk})$ the successful verification of certificate C on message m with public key pk, the statement that the owner of pk_ℓ has to prove can be formalized by the following logical formula:

$$\mathsf{ver}(\mathsf{pk}_2, \mathsf{C}_1, \mathsf{pk}_1) \wedge \left[\bigwedge_{i=2}^{\ell-1} \mathsf{ver}(\mathsf{pk}_{i+1}, \mathsf{C}_i, \mathsf{pk}_i)\right] \wedge \mathsf{ver}(\mathsf{hash}(\mathsf{m}), \mathsf{sig}(\mathsf{hash}(\mathsf{m}), \mathsf{sk}_\ell), \mathsf{pk}_\ell) \quad (1)$$

which can be read as "the verification of signature C_1 on message pk_2 with verification key pk_1 succeeds and for all i from 2 to $\ell - 1$, the verifications of C_i on pk_{i+1} with pk_i succeed and the verification of the signature on the hash of m with pk_ℓ succeeds." For efficiency reasons, the sender signs the hash of the message she is willing to authenticate. Since the proof should not reveal the user identities, we weaken this statement by existentially quantifying over all secret witnesses:[6]

$$\begin{aligned}&\exists\ \alpha_1,\ldots,\alpha_{2\ell-1}:\\&\mathsf{ver}(\alpha_1,\alpha_2,\mathsf{pk}_1)\wedge\left[\bigwedge_{i=2}^{\ell-1}\mathsf{ver}(\alpha_{2i-1},\alpha_{2i},\alpha_{2i-3})\right]\wedge\mathsf{ver}(\mathsf{hash}(\mathsf{m}),\alpha_{2\ell-1},\alpha_{2\ell-3})\end{aligned}\tag{2}$$

This statement only reveals the public key pk_1 of the intended recipient, the hash of the authenticated message m, and the length of the chain (i.e., the trust level of the sender). The zero-knowledge proof of this statement is sent to the verifier, who, after successful verification, will authenticate message m as coming from a principal of level $\ell-1$. Figure 1 schematically shows our protocol for a certificate chain of length 2. To execute this algorithm, we solely assume that the prover can efficiently retrieve the certificates composing the chain. In an established web of trust, public keys and attached certificates are usually uploaded on key servers and are thus publicly available. Our approach, however, is general and does not put any constraints on the way certificates are distributed (for instance, they could be exchanged by private communication). We just require that the prover has access to the certificate chain linking her key to the verifier's one.

Input: A chain $\mathcal{C}=(\mathsf{C}_1,\ldots,\mathsf{C}_{\ell-1})$ from pk_1 to pk_ℓ, signing key sk_ℓ, recipient U owner of pk_1, and message m.

```
1. Set $\mathsf{sig_m} \leftarrow \mathsf{sig}(\mathsf{hash}(\mathsf{m}),\mathsf{sk}_\ell)$.
2. Set formula
   $f \leftarrow \exists\ \alpha_1,\ldots,\alpha_{2\ell-1} : \mathsf{ver}(\alpha_1,\alpha_2,\mathsf{pk}_1) \wedge [\bigwedge_{i=2}^{l-1}\mathsf{ver}(\alpha_{2i-1},\alpha_{2i},\alpha_{2i-3})] \wedge \mathsf{ver}(\mathsf{hash}(\mathsf{m}),\alpha_{2\ell-1},\alpha_{2\ell-3})$.
3. Set witness $\widetilde{w} \leftarrow (\mathsf{pk}_2,\mathsf{C}_1,\ldots,\mathsf{pk}_\ell,\mathsf{C}_{\ell-1},\mathsf{sig_m})$.
4. Generate non-interactive zero-knowledge proof $\mathsf{ZK}(f)$ for statement $f$ using
   witnesses $\widetilde{w}$.
5. Send $\mathsf{m},\mathsf{ZK}(f),\mathsf{pk}_1,\mathsf{hash}(\mathsf{m}))$ to $\mathsf{U}$.
```

Fig. 2. Anonymous message exchange protocol

3 Cryptographic Protocol

For implementing the ideas described in the previous sections, we need (*i*) a digital signature scheme that allows for efficient zero-knowledge proofs and (*ii*) an expressive set of zero-knowledge proofs that can be combined together in conjunctive and disjunctive forms. For signing messages, we rely on the Camenisch-Lysyanskaya signature scheme [16] while, for proving statements about certificate chains, we propose a novel non-interactive zero-knowledge proof of knowledge based on Σ-protocols [26]. We first review the basic building blocks and subsequently describe the construction of our zero-knowledge proof scheme.

[6] Here and throughout the paper, we use the convention introduced in [18] that Greek letters denote those values that are kept secret by the proof.

3.1 Camenisch-Lysyanskaya Signature

This signature scheme was introduced in [16] together with some zero-knowledge proofs. None of them, however, deals with situations in which every value involved in the verification (and, in particular, the verification key) must be kept secret, as required by the statements considered in this paper. This circumstance required us to develop a novel zero-knowledge proof.

We will now give a short overview of this signature scheme. A public key is a tuple $\mathsf{pk} = (a, b, c, n)$ where $n = p \cdot q$ is a special RSA modulus with $p = 2 \cdot p' + 1$, $q = 2 \cdot q' + 1$, and p, p', q, q' are primes. The numbers a, b, and c are uniformly random elements of $\mathsf{QR}(n)$, the group of quadratic residues modulo n. The corresponding secret key is $\mathsf{sk} = p$. Since factorizing n is assumed to be hard, the attacker cannot efficiently compute sk. To sign a given message $m \in [0, ..., 2^{\ell_m})$, one chooses a random prime e of length $\ell_e \geq \ell_m + 2$ and a random number $s \in [0, ..., 2^{\ell_m + \ell_n + \ell})$ where ℓ_n is the bit-length of n and ℓ is a security parameter. In practice, $\ell = 160$ is considered secure. Finally, one computes v such that:

$$v \equiv_n (a^m \cdot b^s \cdot c)^{1/e} \tag{3}$$

Here and throughout this paper, we write $v \equiv_n u$ to say that u is equivalent to v modulo n. Notice that the factorization of n is used to efficiently compute $1/e$. The signature on message m is the tuple $\mathsf{sig}_m = (e, s, v)$. Given $\mathsf{pk} = (a, b, c, n)$, m, and $\mathsf{sig}_m = (e, s, v)$, the verification of the signature sig_m is performed by checking that $2^{\ell_e - 1} < e < 2^{\ell_e}$ along with the following equivalence:

$$v^e \equiv_n (a^m \cdot b^s \cdot c) \tag{4}$$

This equation constitutes the cryptographic instantiation of the symbolic predicate $\mathsf{ver}(m, \mathsf{sig}_m, \mathsf{pk})$ discussed in Section 2. Under the strong RSA assumption, the Camenisch-Lysyanskaya signature scheme is secure against existential forgery attacks. Security against existential forgery is the standard notion of security when dealing with signature schemes.

Definition 2 (Strong RSA Assumption). *The strong RSA assumption states that it is hard, on input an RSA modulus n and an element $u \in \mathbb{Z}_n^*$, to compute values $e > 1$ and v such that $v^e \equiv u \mod n$. More formally, we assume that for all polynomial-time circuit families $\{\mathcal{A}_k\}$, there exists a negligible function $\mu(k)$ such that*

$$\Pr\left[e > 1 \wedge v^e \equiv_n u : n \leftarrow \mathsf{RSAmodulus}(1^k); u \leftarrow QR_n; (v, e) \leftarrow \mathcal{A}_k(n, u)\right] = \mu(k)$$

3.2 Zero-Knowledge Proofs and Σ-Protocols

Zero-knowledge proofs were first introduced in [37] and have since then become a key element of many cryptographic protocols. A zero-knowledge proof is an interactive proof system (P, V) between two parties: The prover P and the verifier V. Both parties obtain the statement to be proven as input, the prover additionally receives a witness to the given statement. Besides the usual completeness and soundness properties, the zero-knowledge property ensures that even a malicious verifier cannot learn any information on the prover's witness.[7] Our zero-knowledge scheme builds on a class of zero-knowledge protocols, called Σ-protocols [35,26], which allow one to prove certain properties of committed values without opening the commitments. We briefly review below the basic building blocks of our scheme. A detailed description of their cryptographic realization is given in Appendix C.

Σ-protocols and their properties. The proofs outlined below belong to the class of Σ-protocols, i.e., protocols composed of three message exchanges: *commitment* (*com*), *challenge* (*ch*), and *response* (*resp*), sent by the prover, the verifier, and the prover respectively. These protocols enjoy the *special soundness* and the *special honest verifier statistical zero-knowledge (SHVSZK)* properties [35,26].

Special soundness is a strong form of *proof of knowledge* and guarantees that a prover is in possession of a witness. This property says that given two protocol transcripts with the same commitment but different challenges, one can extract a witness to the proven statement. Honest verifier zero-knowledge is a variant of the zero-knowledge property where the verifier chooses the challenge uniformly at random from the according challenge space and, in particular, independently of the commitment sent by the prover.[8] We write $\{\mathsf{PK}(\widetilde{\alpha}) :\ S\}$ to denote a proof of knowledge of witnesses $\widetilde{\alpha}$ for statement S.

As shown in [26], Σ-protocols can be combined together to prove logical conjunctions and disjunctions of their respective statements.

Lemma 1 (Logical Combination of Σ-protocols [26]). *Assume that (P_1, V_1) and (P_2, V_2) are SHVSZK and have special soundness and overwhelming completeness for relations R_1 and R_2 respectively. Assume that $M_1 \supseteq L_{R_1}$ and $M_2 \supseteq L_{R_2}$ where $L_R := \{(x, y) \mid xRy\}$. Assume that for both schemes, the verifier accepts the output of the simulator with overwhelming probability.*

Then there exist SHVSZK proof schemes for the relations $R_\wedge := R_1 \wedge_{M_1,M_2} R_2$ and $R_\vee := R_1 \vee_{M_1,M_2} R_2$.

Intuitively, the M_i represent well-formed inputs and are needed for completeness reasons. The construction is given in Appendix C.

[7] The zero-knowledge property is formalized using a simulator that, without having access to the witness to a given statement, creates simulated proof transcripts that are indistinguishable from actual protocol transcripts. Intuitively, this guarantees that the proof cannot be used to gain any information on the witness.

[8] In general, zero-knowledge implies honest-verifier zero-knowledge but the converse does not necessarily hold. In our setting, however, focusing on honest verifiers does not restrict the power of the attacker since the proof will be eventually made non-interactive using the Fiat-Shamir heuristic [30], which lets the prover herself choose the challenge by using the random oracle, without interacting with the verifier.

Commitments. A commitment scheme consists of the commit phase and the open phase. Intuitively, it is not possible to look inside a commitment until it is opened (hiding property) and the committing principal cannot change the content while opening (binding property). We use the integer commitment scheme described in [44]. In the following, we let $[\![c]\!]$ denote the value committed to in c.

Range proofs. We use the range proofs proposed in [32]. A range proof guarantees that a certain committed value lies in the interval (A, B), where A and B are integers. This proof will be denoted by $\{\mathsf{PK}(\alpha) : [\![c]\!] = \alpha \wedge A < \alpha < B\}$ Notice that this proof does not reveal α, just the commitment c and the bounds A and B of the interval.

Proofs of arithmetic operations. Our protocol also uses some of the protocols presented in [17] for proving sums, multiplications, and exponentiations of committed values in zero-knowledge (i.e., without opening the commitments and revealing the witnesses). These proofs will be denoted by

$$\begin{array}{l}
\{\mathsf{PK}(\alpha,\beta,\delta,\nu): \ [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge [\![c_d]\!] = \delta \wedge [\![c_n]\!] = \nu \wedge \alpha + \beta \equiv_\nu \delta\} \\
\{\mathsf{PK}(\alpha,\beta,\delta,\nu): \ [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge [\![c_d]\!] = \delta \wedge [\![c_n]\!] = \nu \wedge \ \alpha \cdot \beta \equiv_\nu \delta\} \\
\{\mathsf{PK}(\alpha,\beta,\delta,\nu): \ [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge [\![c_d]\!] = \delta \wedge [\![c_n]\!] = \nu \wedge \quad \alpha^\beta \equiv_\nu \delta\}
\end{array}$$

3.3 Our Protocol

Our goal is to compute the verification equation (4) in zero-knowledge. This is achieved by the zero-knowledge protocol (5). We first recompute the exponentiations in the signature verification equation, i.e., $\tau_1 \triangleq a^m$, $\tau_2 \triangleq b^s$, $\tau_4 \triangleq a^m b^s$, and $\tau_3 \triangleq v^e$, and check if $v^e \equiv_n a^m b^s c$ (cf. line (a)). We then test whether the signed message and the verification prime number are in the appropriate ranges (cf. line (b)). This protocol constitutes the cryptographic instantiation of the symbolic proof for the statement $\exists\ \alpha_{\mathsf{m}}, \alpha_{\mathsf{sig}}, \alpha_{\mathsf{pk}} : \mathsf{ver}(\alpha_m, \alpha_{\mathsf{sig}}, \alpha_{\mathsf{pk}})$ discussed in Section 2 with $\alpha_m = \mu$, $\alpha_{\mathsf{sig}} = (\nu, \sigma, \epsilon)$, and $\alpha_{\mathsf{pk}} = (\alpha, \beta, \gamma, \eta)$.

$$\left\{\begin{array}{ll}
\mathsf{PK}(\alpha,\beta,\gamma,\epsilon,\eta,\mu,\nu,\sigma,\tau_1,\tau_2,\tau_3,\tau_4) : [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge & \\
[\![c_c]\!] = \gamma \wedge [\![c_n]\!] = \eta \wedge [\![c_m]\!] = \mu \wedge [\![c_v]\!] = \nu \wedge [\![c_s]\!] = \sigma \wedge [\![c_e]\!] = \epsilon & \\
\wedge [\![c_{(a^m)}]\!] = \tau_1 \wedge [\![c_{(b^s)}]\!] = \tau_2 \wedge [\![c_{(v^e)}]\!] = \tau_3 \wedge [\![c_{(a^m b^s)}]\!] = \tau_4 & \\
\tau_1 \equiv_\eta \alpha^\mu \wedge \tau_2 \equiv_\eta \beta^\sigma \wedge \tau_3 \equiv_\eta \nu^\epsilon \wedge \tau_4 \equiv_\eta \tau_1 \cdot \tau_2 \wedge \tau_3 \equiv_\eta \tau_4 \cdot \gamma & (a) \\
\wedge\ 0 \le \mu < 2^{\ell_m} \ \wedge \ 2^{\ell_m+1} < \epsilon < 2^{\ell_m+2} & (b)
\end{array}\right\} \qquad (5)$$

Zero-knowledge proofs for single chain elements are combined together in conjunctive form to prove the existence of a valid certificate chain, as formalized in equation (2). In particular, every occurrence of value u is instantiated with the same commitment c_u. This ensures the equality of the values appearing in different chain element proofs. We reveal the public key of the verifier and the hash of the signed message by opening the corresponding commitments.

Theorem 1. *Let c_a, c_b, c_c, c_m, c_s, c_v, c_e, and c_n be integer commitments and let $c_{(a^m)}$, $c_{(b^s)}$, $c_{(v^e)}$, and $c_{(a^m b^s)}$ be auxiliary commitments. Then, the protocol from equation (5) is a special honest verifier statistical zero-knowledge proof with* *special soundness that the values committed to in c_a, c_b, c_c, c_m, c_s, c_v, c_e, and c_n fulfill the Camenisch-Lysyanskaya signature scheme verification equation.*

Proof. The completeness follows from inspection of the protocol and the verification equation of the signature scheme. Special soundness and SHVSZK follow from the special soundness and the SHVSZK property of the individual proofs by applying Lemma 1.

Finally, we apply the Fiat-Shamir heuristic [30] to make our protocol non-interactive.

3.4 Complexity Analysis

We now analyze the communication complexity in terms of the various security parameters of the different zero-knowledge proofs:

ℓ	determines the maximum bit length of a committed value.
ϵ	> 1 is a security parameter.
ℓ_b	denotes the maximum bit length of the exponents used in the exponentiation proof. Typically, $\ell_b = \ell$.
C	describes the challenge space $CS := \{0, ..., 2^C - 1\}$. Our protocol will have soundness error 2^{-C}, i.e., a cheating prover will convince an honest verifier with probability of at most 2^{-C}.

As $p = 2 \cdot q + 1$ and $q > 2^{2\epsilon\ell+5}$, we have that all numbers computed modulo p or modulo q require a communication complexity of $\Theta(\epsilon \cdot \ell)$. In the following, we assume that we already applied the Fiat-Shamir heuristic and hence do not consider the challenge as it can be computed by the verifier.

– Proofs of knowledge of a representation use one message modulo p as commitment and n messages modulo q as response. As $n \leq 3$ for our proofs, we get that a representation proof is in $\Theta(\epsilon \cdot \ell)$.
– Multiplication and addition proofs are in $\Theta(\epsilon \cdot \ell)$ as both are representation proofs.
– In a range proof, the commitment is computed modulo p and the response s is a value in $\mathbb{Z}$. Since s is at most the sum of two numbers smaller than q, the whole proof is in $\Theta(\epsilon \cdot \ell)$. However, we need to run the protocol C times to achieve a soundness equal to the soundness of the other proofs. Hence, a range proof protocol is actually in $\Theta(\epsilon \cdot \ell \cdot C)$.
– The exponentiation proof is a combination of many knowledge of representation, knowledge of discrete logarithm, and range proofs. A careful analysis shows that the dominating parts are the range proofs for all the intermediate results used in the protocol; an exponentiation proof is in $\Theta(\epsilon \cdot \ell \cdot \ell_b \cdot C)$.

Since exponentiation proofs are the dominant factor, our zero-knowledge protocol for the Camenisch-Lysyanskaya signature scheme is in $\Theta(\epsilon \cdot \ell \cdot \ell_b \cdot C)$.

Once all aforementioned parameters are fixed, the communication complexity is linear in the length of the certificate chain as each chain element requires exactly one signature verification proof.

3.5 Implementation

We implemented our protocol as an extension of the OpenPGP standard. Our system relies on key servers that provide standard OpenPGP functionality and additionally maintain the certificates from the anonymous web of trust. The authenticity of anonymous web of trust keys is established by OpenPGP certificates. Arithmetic operations are performed by using MIRACL [48]. The implementation is in Java and comprises roughly 6000 lines of code. A prototypical implementation is freely available at [7].

4 Partial Disclosure: Beyond the All-or-Nothing Barrier

The cryptographic protocol described so far allows the prover to show the existence of a certificate chain without revealing anything other than the length of the chain. In some situations, however, the length of the chain might reveal too much about the prover's identity while in some other scenarios, users might desire more precise trust measures, even at the price of sacrificing a little their anonymity. There is indeed an inherent trade-off between anonymity and trust. In this section we develop extensions of our protocol that allow users to fine-tune the degree of anonymity and trust.

Hiding the chain length. The length of the chain might actually reveal some information about the sender, depending on the topology of the web of trust. For instance, in the extreme scenario where the intended recipient has certified just one key and the length of the chain is 1, the intended recipient knows exactly the identity of the sender. In this case, the prover can arbitrarily increase the length of the chain proven in zero-knowledge by attaching self-generated certificates.

```
Input: A chain $\mathcal{C}$ from $\mathsf{pk}_1$ to $\mathsf{pk}_\ell$ and signing key $\mathsf{sk}_\ell$
  1. Create a new key-pair $(\mathsf{pk}', \mathsf{sk}')$
  2. Set $\mathcal{C} \leftarrow \mathcal{C}, \mathsf{sig}(\mathsf{pk}', \mathsf{sk}_\ell)$
  3. Return $\mathcal{C}$ and $\mathsf{sk}'$
```

Note that the keys used in these certificates need not be uploaded onto a server as the verifier does not need them to check the proof and, after the proof is generated, these keys can be discarded. Indeed, a proof for a certificate chain of length n does not guarantee that the prover is n hops away from the verifier, but that she is *at most* n hops away.

Partial release of secrets. To achieve fine-grained trust properties, we now consider certificate attributes, such as user name and key expiration date, and show how to reveal some of them while keeping the others secret. For instance, we might want to reveal the key expiration date while hiding confidential information such as the user name. We recall that participants in a web of trust place the signature on the concatenation of a public key and a set of attributes. Intuitively, instead of proving $\exists\ \alpha_{\mathsf{m}}, \alpha_{\mathsf{sig}}, \alpha_{\mathsf{pk}} : \mathsf{ver}(\alpha_{\mathsf{m}}, \alpha_{\mathsf{sig}}, \alpha_{\mathsf{pk}})$, we would like to prove a statement of the form $\exists\ \alpha_{\mathsf{S}}, \alpha_{\mathsf{sig}}, \alpha_{\mathsf{pk}}, \alpha_{\mathsf{K}}, \alpha_{\mathsf{A}}.\mathsf{ver}(\alpha_{\mathsf{S}}, \alpha_{\mathsf{sig}}, \alpha_{\mathsf{pk}}) \wedge \alpha_{\mathsf{S}} = (\alpha_{\mathsf{K}}, \alpha_{\mathsf{A}})$ and then reveal (part of) the attributes α_{A}. The concatenation of the public key and the attributes is implemented as $b = k \cdot 2^{\ell} + A$ where ℓ is an a priori fixed upper bound on the length of the attribute set. The idea is to split b in zero-knowledge and to reveal some of the components to the verifier. Given commitment c_{kA} on public key k and attributes A, commitment c_k on k, and commitment c_A on A, we execute the following zero-knowledge protocol:

$$\left\{\mathsf{PK}(\alpha, \kappa, \tau) :\ [\![c_k]\!] = \kappa \wedge [\![c_A]\!] = \alpha \wedge [\![c_{kA}]\!] = \tau \wedge \tau = \kappa \cdot 2^{\ell} + \alpha \wedge 0 \leq \alpha < 2^{\ell}\right\}$$

We can then open c_A and release all the attributes A to the verifier or apply the protocol again on c_A to select which attributes have to be revealed.

Dynamic trust relationships and key expiration. Since trust relationships may vary over time, it is important to provide users with the possibility to periodically update their certificates. Our system incorporates two distinct key expiration mechanisms.

The first mechanism is based on a global version number that is attached to all public keys as an attribute. Periodically after a fixed interval, all keys have to be generated from scratch, re-signed, and tagged with the updated version number. Proving a key valid translates into showing that it is tagged with the most recent version number. This version number is revealed using our partial secret release protocol. As the interval is globally fixed, revealing the version number does not leak any information about the key.

In order to provide the user with the possibility to independently decide the validity of each certificate, we also support a second mechanism based on a key expiration date. Users can use our partial secret release protocol to selectively reveal the expiration date of a key. Since the exact expiration date might uniquely identify the public key, one can also prove $\{\mathsf{PK}(\epsilon) : [\![c_e]\!] = \epsilon \wedge \textit{current date} < \epsilon < \textit{ub}\}$ given a commitment c_e on the expiration date attribute ϵ and a suitable upper bound $\textit{ub}$ for all possible key expiration dates.

Notice that the OpenPGP standard [15] incorporates a key revocation mechanism, which is implemented by a special signature (also called revocation signature) that is attached to the revoked key by the revoking principal. Although conceptually appealing, such a revocation mechanism is not compatible with our framework since there is no way to prove in zero-knowledge that a certain key has not been revoked. In particular, even if revoked, the key and the according certificates could still be used in our zero-knowledge proof.

Conjunctive and disjunctive statements over certificate chains. Σ-protocols allow us to prove logical conjunction and disjunction of statements. Proving a conjunctive statement over certificate chains strengthens trust at the price of decreasing anonymity guarantees, whereas a disjunctive statement enhances the anonymity guarantees but diminishes trust.

In a way of example, consider Figure 3 (a) where A is trusted by both C_1 and C_2, and D is only trusted by C_2. Assume A is interested in authenticating to a party B trusting both C_1 and C_2 and suppose also that A does not know the

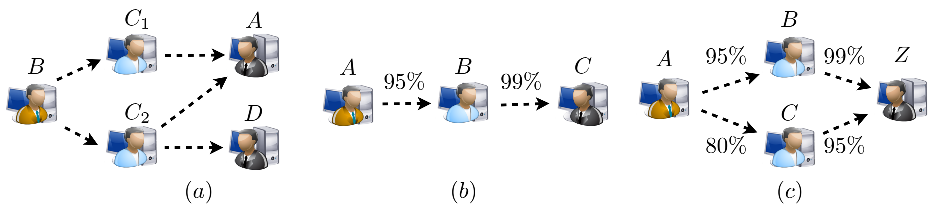


Fig. 3. Webs of trust

public key of B. If A proves that she is trusted by C_1 or C_2, a curious principal will not be able to distinguish whether the message originated from D or A. The trust guarantee provided by the proof, however, may be low if, for instance, the link between C_2 and D is weak (cf. the following discussion on trust measures).

A proof that A is trusted by C_1 and C_2 strengthens the trust guarantee. One can, however, compute the intersection of the principals trusted by C_1 and C_2, potentially reducing the anonymity guarantees. In this example, the intersection uniquely identifies A as the prover. This example shows that there is often an inherent trade-off between trust and anonymity. The expressiveness of our zero-knowledge proof scheme is crucial to fine tune the security requirements according to the application scenario.

Trust measures. In the following, we extend our approach to trust measures. We will focus in particular on the trust model from [21]. The examples in this section are intentionally borrowed from [21] in order to show the applicability of our framework to existing trust models. Consider the web of trust in Figure 3 (b). As shown by the weight of the two links, the trust of B in C is higher than the trust of A in B. The trust measure proposed in [21] is based on the multiplication of the trust values of the individual links. Therefore the trust degree provided by the chain between A and C is $95\% \cdot 99\% = 94.05\%$.

We devise a proof that reveals the trust degree provided by a given chain, without disclosing the weight of individual links, since this might compromise the anonymity of participants. In case even the exact trust degree is considered too informative on the identity of the parties involved in the chain, we can approximate this value using range proofs (cf. key expiration).

In addition to proving the validity of the certificate chain of Figure 3 (b), the prover executes the following protocol:

$$\{\mathsf{PK}(\alpha,\beta,\gamma): \ [\![c_t]\!] = \alpha \wedge [\![c_{t_1}]\!] = \beta \wedge [\![c_{t_2}]\!] = \gamma \wedge \alpha \equiv_P \beta \cdot \gamma\}$$

where c_{t_1} and c_{t_2} are the commitments to the certificate attributes 95 and 99, P is a large publicly known prime (cf. Proving that two committed numbers are not equal in Appendix B), and c_t is a commitment to 9405, which is opened by the prover. Since we cannot reason on rational numbers and consequently on divisions,[9] the verifier has to perform the remaining computation on the value $[\![c_t]\!] = 9405$, namely, $1 - (1 - 9405/10000) = 94.05\%$.

We now show how our protocol can be extended to deal with even more complex scenarios. Consider the graph in Figure 3 (c): Z has to show that there exist two distinct paths from A to Z. The total trust degree is computed as $1 - (1 - 95\% \cdot 99\%) \cdot (1 - 80\% \cdot 95\%) \approx 98.6\%$.

The corresponding zero-knowledge proof is computed as follows. Given the commitments c_{s_1}, c_{s_2}, c_{s_3}, and c_{s_4} on the certificates $cert_{AB}$, $cert_{AC}$, $cert_{CZ}$, and $cert_{BZ}$, where $cert_{IJ}$ denotes the certificate issued by I on J's public key, and the commitments c_{t_1}, c_{t_2}, c_{t_3}, and c_{t_4} on the corresponding trust values, in addition to showing that both chains are valid we run the following protocol:

$$\left\{\begin{array}{c} \mathsf{PK}(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2) : \ [\![c_{t_1}]\!] = \alpha_1 \wedge [\![c_{t_2}]\!] = \alpha_2 \wedge [\![c_{t_3}]\!] = \alpha_3 \wedge [\![c_{t_4}]\!] = \alpha_4 \wedge \\ [\![c_{s_1}]\!] = \beta_1 \wedge [\![c_{s_2}]\!] = \beta_2 \wedge \beta_1 \neq \beta_2 \wedge [\![c_r]\!] \equiv_P ([\![c_{10000}]\!] - \alpha_1 \cdot \alpha_3) \cdot ([\![c_{10000}]\!] - \alpha_2 \cdot \alpha_4) \end{array}\right\}$$

Proving $[\![c_{s_1}]\!] \neq [\![c_{s_2}]\!]$ ensures that the first two signatures, and therefore the two chains, are different. The rest of the proof computes in zero-knowledge the total trust value as follows: $[\![c_r]\!] = (10000 - 95 \cdot 99) \cdot (10000 - 80 \cdot 95) = 1428000$ (c_{10000} is a commitment to 10000). The verifier then computes $(10^8 - [\![c_r]\!])/10^8 \approx 98.6\%$. Although the numbers grow quickly with the chain length and the number of parallel paths, $P \gg 10^{100}$ is large enough for any reasonably sized chain.

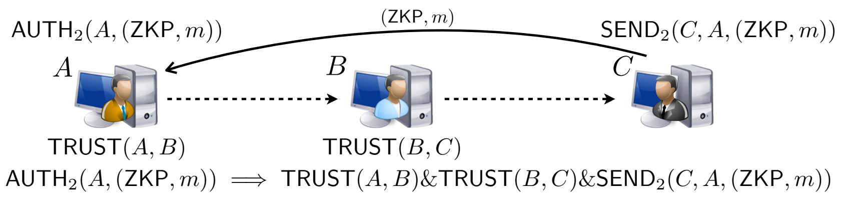
5 Formal Verification

The cryptographic proof from Section 3 ensures that our scheme enjoys the special soundness and honest verifier statistical zero-knowledge properties. It is important to verify, however, that the protocol as a whole guarantees the intended trust and anonymity properties. We conducted a formal security analysis by modeling our protocol in the applied pi-calculus [2], formalizing the trust property as an authorization policy and the anonymity property as an observational equivalence relation, and verifying our model with ProVerif [12,1], a state-of-the-art automated theorem prover that provides security proofs for an unbounded number of protocol sessions. We model zero-knowledge proofs following the approach proposed in [8], for which computational soundness results exist [9]. For easing the presentation, in this section we focus on certificate chains without attributes. The ProVerif scripts used in the analysis are reported in Appendix D.

Attacker model. In our analysis, we consider a standard symbolic Dolev-Yao active attacker who dictates the certificates released by each party (i.e., the attacker controls the web of trust), the certificate chains proven in zero-knowledge, and the proofs received by each verifier.

Verification of trust. We partition the set of parties into honest and compromised. Honest parties generate a fresh key-pair, publish the public component,

[9] Computing $1/m$ for a given m results in a number u such that $m \cdot u = 1 \mod q$, e.g., $1/4 = 5 \mod 19$.



$\mathsf{AUTH}_2(A,(\mathsf{ZKP},m))$
(ZKP,m)
$\mathsf{SEND}_2(C,A,(\mathsf{ZKP},m))$
A
B
C
$\mathsf{TRUST}(A,B)$
$\mathsf{TRUST}(B,C)$
$\mathsf{AUTH}_2(A,(\mathsf{ZKP},m)) \implies \mathsf{TRUST}(A,B)\&\mathsf{TRUST}(B,C)\&\mathsf{SEND}_2(C,A,(\mathsf{ZKP},m))$

Fig. 4. Trust policy

and engage in three distinct activities: Certificate generation, proof generation, and proof verification.

We decorate security-related protocol events with logical predicates, which constitute the building blocks of the authorization policy formalizing the trust property (cf. Figure 4). The event $\mathsf{TRUST}(x,y)$ describes the point in the protocol where the honest party associated with public key x releases a certificate for public key y. The event $\mathsf{COMPR}(x)$ tracks the compromise of the party associated with public key x (i.e., this party is under the control of the attacker, which also knows the corresponding private key). The event $\mathsf{SEND}_i(x,y,z)$ describes the point in the protocol where the party associated with public key x sends a zero-knowledge proof for a certificate chain of length i to the party associated with public key y to authenticate message z. Finally, the event $\mathsf{AUTH}_i(x,y)$ describes the point in the protocol where the party associated with public key x authenticates message y as coming from a party of trust level i. The trust property is formalized as the following authorization policy:

$$
\begin{aligned}
\mathsf{AUTH}_2(id2,x) \Rightarrow\ & \mathsf{SEND}_2(id1,id2,x)\ \&\ \mathsf{TRUST}(id2,id3)\ \&\ \mathsf{TRUST}(id3,id1)) && (1)\\
& \mid (\mathsf{TRUST}(id2,id3)\ \&\ \mathsf{TRUST}(id3,id1)\ \&\ \mathsf{COMPR}(id1)) && (2)\\
& \mid (\mathsf{TRUST}(id2,id3)\ \&\ \mathsf{COMPR}(id3)). && (3)
\end{aligned}
$$

For the sake of simplicity, we focus on certificate chains of length 2: The extension to arbitrary chain lengths is straightforward. This policy says that in all execution traces, the event $\mathsf{AUTH}_2(id2,x)$ has to be preceded by either (1) $\mathsf{SEND}_2(id1,id2,x)$ and $\mathsf{TRUST}(id2,id3)$ and $\mathsf{TRUST}(id3,id1)$ (i.e., all parties are honest), or (2) $\mathsf{TRUST}(id2,id3)$ and $\mathsf{TRUST}(id3,id1)$ and $\mathsf{COMPR}(id1)$ (i.e., all parties except for the prover are honest), or (3) $\mathsf{TRUST}(id2,id3)$ and $\mathsf{COMPR}(id3)$ (i.e., the party trusted by the verifier is compromised and the attacker has chosen to lengthen the certificate chain by an additional, possibly fake, certificate). In other words, this policy says that whenever the verifier authenticates a message as coming from a party of trust level i, then indeed a party of trust level i or less has started a protocol session with the verifier to authenticate that message.

This authorization policy is successfully verified by ProVerif and the analysis terminates in 3 seconds. The formal analysis highlighted a couple of important requirements for the safety of our protocol. First, the verifier has to check that the authenticated message is not a public key,[10] otherwise the following attack would

[10] We recall that parties sign the hash of messages and these are shorter than keys.

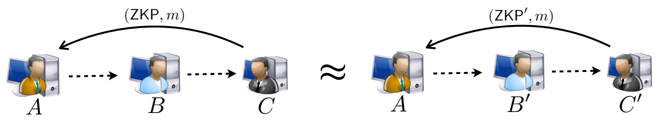


Fig. 5. Anonymity game

be possible: The attacker gathers a certificate chain of length $i + 1$ and builds a zero-knowledge proof for a certificate chain of length i, authenticating the public key signed in the $i + 1$-th certificate as coming from the party associated with the public key signed in the i-th certificate. For a similar reason, signatures on messages other than public keys cannot be sent in plain or must be tagged differently from the signatures proven in zero-knowledge.

Verification of anonymity. Intuitively, we formalize the anonymity property as a cryptographic game where two principals act in a web of trust set up by the attacker and one of them authenticates by proving in zero-knowledge a certificate chain chosen by the attacker. If the attacker cannot guess which of the two principals generated this zero-knowledge proof, then the protocol guarantees anonymity. Our model includes an arbitrary number of honest and compromised parties as well as the two (honest) principals engaging in the anonymity game.

The anonymity game is defined by two distinct processes that are replicated (i.e., spawned an unbounded number of times) and in parallel composition (i.e., concurrently executed). In the first process, each of the two principals releases certificates as dictated by the attacker. Since the attacker controls also the certificates released by the other parties in the system, both honest and compromised ones, the attacker controls the topology of the whole web of trust. In the second process, the two principals receive two (possibly different) certificate chains from the attacker. If both certificate chains are valid and of the same length, we non-deterministically choose one of the two principals and we let it output the corresponding zero-knowledge proof. The observational equivalence relation $\approx$ (cf. Figure 5) says that the attacker should not be able to determine which of the two principals output the zero-knowledge proof.

ProVerif successfully verifies this observational equivalence relation. This implies that our protocol guarantees the anonymity of users even against our strong adversarial model. Since processes are replicated and the two principals may output an unbounded number of zero-knowledge proofs, our protocol additionally provides unlinkability, that is, the attacker is not able to tell if two zero-knowledge proofs come from the same principal or not.

6 Conclusion

We have proposed a cryptographic protocol for anonymous communication in webs of trust. We reconcile trust and anonymity, two seemingly conflicting requirements, using a novel zero-knowledge proof that allows the sender to prove the existence of a certificate chain without revealing her identity and the receiver to verify the trust level of the sender. The zero-knowledge proof scheme is general and accommodates different aspects of webs of trust, such as key expiration, trust measures, and existence of multiple certificate chains. We conducted a formal security analysis of our protocol, showing that trust and anonymity are guaranteed even in a strong adversarial setting.

Our approach inherently requires that the certificates comprising the certificate chain are accessible to the prover, since they have to be proven in zero-knowledge. While public relationships are not a problem in a company (e.g., boss, employee, trainee, etc.), there might be privacy issues in other settings, e.g., in the context of social networks where users may want to keep their social relationships secret. We stress that our approach does *not* require the whole relationship graph to be public; only the certificates used in the proof need to be accessible to the prover.

In a distributed social network, for instance, we envision the following *local* certificate distribution mechanism: A expresses her friendship with B by signing B's public key and sending the corresponding certificate C_{AB} to him. If A wants her profile to be available only to her friends (this corresponds to a "friends only" policy in Facebook [29]), then B is expected to keep C_{AB} to himself. Should A instead opt for a "friends of friends" policy (which is also available in Facebook [29]), then A authorizes B to release C_{AB} to his friends in order to let them anonymously authenticate with A (with a zero-knowledge proof of length 2). B's friends might express interest in authenticating with A, after looking at a preview of A's profile, which could be made available by B.

In general, there is an inherent trade-off between the privacy of the relationship graph and the anonymity guarantees of our scheme. On the one hand, if the relationship graph is fully private, then the prover does not know how many other principals have her own trust level. Hence, in the extreme scenario in which the verifier and all the principals in the chain have issued just one certificate, the prover is just anonymous in the set of principals occurring in the chain (due to the chain enlargement technique discussed in Section 4). On the other hand, if the relationship graph is public, as in GnuPG, the prover can be certain of her anonymity guarantees. As a future work, it would be interesting to investigate techniques to solve this tension, e.g., by selectively disclosing parts of the relationship graph in order to ensure meaningful anonymity properties.

Acknowledgments This work was partially supported by the initiative for excellence and the Emmy Noether program of the German federal government and by Miur Project SOFT (*Security Oriented Formal Techniques*).

References

1. Martín Abadi, Bruno Blanchet, and Cédric Fournet. Automated verification of selected equivalences for security protocols. In *Proc. 20th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 331–340. IEEE Computer Society Press, 2005.

2. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL)*, pages 104–115. ACM Press, 2001.
3. Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proc. 1997 workshop on New Security Paradigms (NSPW)*, pages 48–60. ACM Press, 1997.
4. Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, 2007.
5. Ronald Ashri, Sarvapali D. Ramchurn, Jordi Sabater, Michael Luck, and Nicholas R. Jennings. Trust evaluation through relationship analysis. In *Proc. 4th international joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 1005–1011. ACM Press, 2005.
6. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, 2000.
7. Michael Backes, Stefan Lorenz, Matteo Maffei, and Kim Pecina. Anonymous webs of trust (tool and long version), 2010. Available at `http://www.lbs.cs.uni-sb.de/awot/`.
8. Michael Backes, Matteo Maffei, and Dominique Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Proc. 29th IEEE Symposium on Security & Privacy*, pages 202–215. IEEE Computer Society Press, 2008.
9. Michael Backes and Dominique Unruh. Computational soundness of symbolic zero-knowledge proofs against active attackers. In *Proc. 21th IEEE Symposium on Computer Security Foundations (CSF)*, pages 255–269. IEEE Computer Society Press, 2008.
10. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer-Verlag, 2009.
11. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology - CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
12. Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE Computer Society Press, 2001.
13. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer-Verlag, 2005.
14. Stefan Brands. Electronic cash systems based on the representation problem in groups of prime order. In *Advances in Cryptology - CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 26.1–26.15. Springer-Verlag, 1993.
15. J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP message format. In *Request for Comments*, volume 4880. Internet Engineering Task Force, 2007.
16. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Proc. 3rd International Conference on Security in Communication Net-*

works (SCN), volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer-Verlag, 2002.

17. Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology - EUROCRYPT 1998*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer-Verlag, 1998.
18. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology - CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.
19. Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A formal model for trust in dynamic networks. In *International Conference on Software Engineering and Formal Methods (SEFM '03)*, pages 54–64. IEEE Computer Society Press, 2003.
20. Barbara Carminati, Elena Ferrari, and Andrea Perego. Rule-based access control for social networks. In *Proc. On the Move to Meaningful Internet Systems 2006 (OTM)*, volume 4278 of *Lecture Notes in Computer Science*, pages 1734–1744. Springer-Verlag, 2006.
21. Germano Caronni. Walking the web of trust. In *Proc. 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 153–158. IEEE Computer Society Press, 2000.
22. Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come - easy go divisible cash. In *Advances in Cryptology - EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer-Verlag, 1998.
23. David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology - EUROCRYPT 1987*, volume 304 of *Lecture Notes in Computer Science*, pages 127–141. Springer-Verlag, 1987.
24. David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology - CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 89 – 105. Springer-Verlag, 1992.
25. David Chaum and Eugene van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.
26. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1994.
27. Josep Domingo-Ferrer, Alexandre Viejo, Francesc Sebé, and Úrsula González-Nicolás. Privacy homomorphisms for social networks with private relationships. *Computer Networks*, 52(15):3007–3016, 2008.
28. Joseph Domingo-Ferror. A public-key protocol for social networks with private relationships. In *Proc. 4th International Conference on Modeling Decisions for Artificial Intelligence (MDAI'07)*, volume 4617 of *Lecture Notes in Computer Science*, pages 373–379. Springer-Verlag, 2007.
29. facebook. `http://www.facebook.com/`.
30. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO 1987*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.
31. Keith Frikken and Preethi Srinivas. Key allocation schemes for private social networks, 2009. To appear in Proc. ACM Workshop on Privacy in the Electronic Society (WPES).

32. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology - CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 1997.
33. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology - CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 1997.
34. Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Proc. 8th International Conference on the Theory and Application of Cryptology and Information Security: ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer-Verlag, 2002.
35. Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
36. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991.
37. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
38. Javier Herranz. Identity-based ring signatures from rsa. *Theoretical Computer Science*, 389(1-2):100–117, 2007.
39. Jingwei Huang and David Nicol. A calculus of trust and its application to pki and identity management. In *Proc. 8th Symposium on Identity and Trust on the Internet*, pages 23–37. ACM Press, 2009.
40. Audun Jøsang. An algebra for assessing trust in certification chains. In *Proc. Network and Distributed System Security Symposium (NDSS)*. Internet Society, 1999.
41. Lance Cottrell, Pr0duct Cypher, Hal Finney, Ian Goldberg, Ben Laurie, Colin Plumb, or Eric Young. Signing as one member of a set of keys. `http://www.abditum.com/ringsig/`.
42. J. Linn. Trust models and management in public key infrastructures, 2000. RSA Laboratories.
43. Ueli Maurer. Modelling a public-key infrastructure. In *Proc. 4th European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *Lecture Notes in Computer Science*, pages 325–350. Springer-Verlag, 1996.
44. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1991.
45. Ronald Linn Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. *Communications of the ACM*, 22(22):612–613, 2001.
46. Jordi Sabater-Mir. Towards the next generation of computational trust and reputation models. In *Proc. 3th International Conference on Modeling Decisions for Artificial Intelligence (MDAI'06)*, volume 3885 of *Lecture Notes in Computer Science*, pages 19–21. Springer-Verlag, 2006.
47. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
48. Michael Scott. Multiprecision Integer and Rational Arithmetic C/C++ Library. `http://www.shamus.ie/`.

49. Dawn Xiaodon Song. Practical forward secure group signature schemes. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 225–234. ACM Press, 2001.
50. The GNU Privacy Guard Team. GnuPG. `http://www.gnupg.org/`.
51. The GNU Privacy Guard Team. The GNU Privacy Handbook. `http://www.gnupg.org/gph/en/manual.pdf`.
52. Amin Tootoonchian, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: better privacy for social networks. In *Proc. 5th International Conference on emerging Network Experiments and Technologies (CoNEXT)*, pages 169–180. ACM Press, 2009.
53. Da-Wei Wang, Churn-Jung Liau, and Tsan sheng Hsu. Privacy protection in social network data disclosure based on granular computing. In *International Conference on Fuzzy Systems 2006*, pages 997 – 1003. IEEE Computer Society Press, 2006.
54. L. Xiong and L. Ling. A reputation-based trust model for peer-to-peer ecommerce communities [extended abstract]. In *Proc. 4th ACM conference on Electronic commerce (EC'03)*, pages 228–229. ACM Press, 2003.

A Trust model

One of the core motivations behind webs of trust as public key infrastructures is the fact that there is no central authority one has to trust. Every participant bases trust decisions on her own policy.

However, this poses problems: Consider a simple web of trust where Alice signed Bob's key and Bob signed Charlie's key. Alice trusts Bob only marginally, i.e., she is not convinced that her signing policy is fully compatible with Bob's policy. What does this say about Charlie's key? Probably Alice should not accept it as a valid key if it is only signed by Bob. In the following, we use trust and validity on the basis of the GnuPG Handbook [51]: *Trust* denotes the belief that the owner of a key acts in accordance with our signing policy and *validity* denotes our belief that a key actually belongs to the designated owner.

Our work is based on the OpenPGP standard [15], which stipulates a method for conveying and expressing trust, namely, trust signatures. Such signatures allow the signer to assert a *transitivity level* and a *trust level*. The former rules the transitivity of trust relationships while the latter allows one to publicly state the amount of trust set in the owner of a key. (Typical trust values are unknown, no trust, marginal, and full.) For instance, a level one trust signature on key k means that k can be used to sign another key k', which will inherit the same trust level as k. Key k', however, is not trusted to sign further keys. In general, a level n trust signature asserts that the owner of a key is trusted to issue level $n-1$ trust signatures. Figure 6 depicts a trust signature chain with a constant trust level. Note that the OpenPGP standard does not require the trust level to remain constant throughout a chain. In practice, common transitivity levels are 0 (direct friendship relation) and 1 (friend of a friend relation). A level zero signature is equivalent to a standard signature in the web of trust. Higher transitivity levels may be useful in certain applications where they have a clear and meaningful interpretation (e.g., reflecting the hierarchical structure

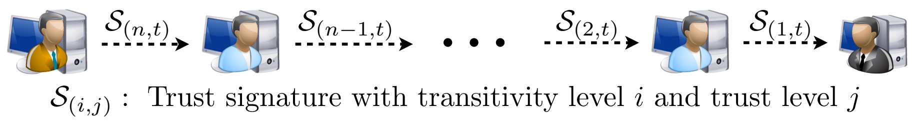

$\mathcal{S}_{(n,t)}$
$\mathcal{S}_{(n-1,t)}$
$\mathcal{S}_{(2,t)}$
$\mathcal{S}_{(1,t)}$
$\mathcal{S}_{(i,j)}$: Trust signature with transitivity level i and trust level j

Fig. 6. Trust signature chain

of a company). PGP, since version 5, as well as GnuPG, depending on user preferences, use transitivity levels and trust levels to calculate the validity of keys. The specific details of these computations are implementation dependent.

Our approach is compatible with the trust signature mechanism and a variety of validity calculation algorithms. In fact, we can selectively reveal both transitivity levels and trust levels in our zero-knowledge proofs as well as compute in zero-knowledge the validity of keys as described in Section 4.

In a way of example, consider the web of trust depicted in Figure 7. This example is intentionally borrowed from the GnuPG Handbook [51] to show the applicability of our framework to existing implementations. A labeled edge denotes a trust signature issued on the target's public key. The first number represents the transitivity level and the second number represents the trust level. Here we assume two trust levels (namely, marginal and full), which are denoted by 1 and 2.[11] We further assume that in the validity calculation algorithm, two signatures of marginal trusted principals are needed to deduce full validity.

The web is rooted at Alice who assigns a signature with level 1 and marginal trust to Dharma and Blake. Thus the keys of Dharma and Blake become automatically fully valid. Since both Dharma and Blake may issue level 0 signatures with marginal trust, Chloe and Francis are marginally trusted by Alice. Chloe's key becomes fully valid as it is signed by two trusted principals. The same holds for Francis' key, which is signed by Chloe and Dharma. Elena's key is considered partially valid as it is signed by only one marginally trusted party. Geoff's key can not be validated.

If Chloe wants to authenticate a message with Alice while remaining anonymous, she should send a zero-knowledge proof that shows the existence of two distinct paths of length two and additionally reveals the transitivity level and the trust level of the incorporated chain elements,[12] as discussed in Section 4.

B Extensions of our zero-knowledge proof system

Proof that two committed numbers are not equal. For proving relations about independent chains, it is necessary to prove that two given commitments do not contain the same number. This will be particularly useful when we show

[11] The OpenPGP standard suggests to use 60 for partial trust and 120 for full trust.

[12] This proof only preserves anonymity if there is at least one more principle with similar chains to Alice.

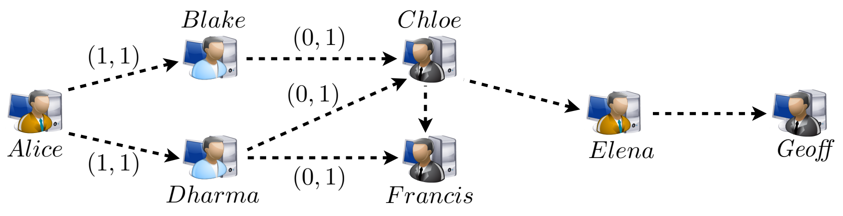


Fig. 7. Hypothetical Web of Trust

how we can increase trust by demonstrating knowledge of multiple paths from the recipient to the sender of a message. Proving that two commitments do not contain the same number allows us to guarantee that multiple paths are distinct, without revealing the nodes occurring therein.

The following protocol proves that c_a and c_b are commitments to different numbers. The intuition is that if $a = b$, then multiplying $a - b$ by any value yields zero modulo P. If $a \neq b$, then multiplying $a - b$ by a random value yields a (non-zero) uniformly random number in the group, thus hiding a and b.

$$\{\mathsf{PK}(\alpha, \beta, \rho) : [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge [\![c_d]\!] = d \wedge [\![c_r]\!] = \rho \wedge \\ \wedge d \equiv_P (\alpha - \beta) \cdot \rho \ \wedge \ d \neq 0 \ \wedge \ 0 < \rho < P\}$$

where c_r is a commitment to a freshly generated random value r, $P \approx \sqrt{q}$ is a publicly known prime, and d is the result of $(a - b) \cdot \rho \bmod P$. Notice that the proof reveals the opening information for the commitment $[\![c_d]\!]$. If d is different from zero, then the verifier knows that the values a and b committed to in c_a and c_b are different.

C Cryptographic protocols

In this section, we review the cryptographic implementation of the zero-knowledge proofs used in our protocol. Unless stated differently, *ch* will denote the random verifier challenge in the Σ-protocol chosen from the challenge space $CS := \{0, \ldots 2^C - 1\}$.

We assume a correctly initialized commitment scheme used by every participant that the discrete logarithm between the two generators is not known.

Commitment. In order to commit to a value m, one first chooses a uniformly random value $r \in \{0, ..., q - 1\}$ and computes $c_m = G^m \cdot H^r \mod p$. In the opening phase, the prover sends m and r to the verifier, who then reconstructs the commitment and compares it to c_m. Here p is a safe prime of the form $p = 2 \cdot q + 1$ such that p and q are primes; G and H are uniformly random generators of the subgroup $\mathcal{G}_q \subset \mathbb{Z}_p^\times$ of prime order q. It is important that the discrete logarithm between the generators G and H is *not known* to the prover as the commitment scheme loses its binding property otherwise. We require that the bit length $|q| > 2^{2\epsilon\ell+5}$ where ℓ determines the maximum bit length of committed values and $\epsilon > 1$ is a security parameter. In the following, we let $\ddot{\ell} = \epsilon\ell + 2$.

We assume a correctly initialized commitment scheme used by every participant. In particular, we assume that p, q, G, and H are constructed as described above such that the discrete logarithm between the two generators is not known.

Representation proofs. Such proofs are used to show knowledge of a representation of an element y with respect to base elements $(g_1, ..., g_m)$ [14,23,47]. This proof is denoted by $\{\mathsf{PK}((\alpha_i)_{i=1,...,m}) : \; y = \prod_{i=1}^{m} g_i^{\alpha_i}\}$. If we use the base elements from the commitment scheme, we often write $\{\mathsf{PK}(\alpha) : \; [\![c]\!] = \alpha\}$ to denote $\{\mathsf{PK}(\alpha, \rho) : \; c = G^{\alpha} H^{\rho}\}$.

The prover chooses $r_1, ..., r_m \in_{\mathrm{R}} \mathbb{Z}_q$ and sends $t := \prod_{i=1}^{m} g_i^{r_i} \mod p$ to the verifier. The prover responds with $s_i := r_i - ch \cdot x_i \mod q$, $i = 1, ..., m$. The verifier accepts iff $t = y^{ch} \prod_{i=1}^{m} g_i^{s_i}$.

Such proofs also enable us to elegantly show that a commitment contains 0 or 1. Given commitment $c = G^m \cdot H^r$, if we are able to prove that $\{\mathsf{PK}(\rho) : \; c = H^{\rho}\}$ or $\{\mathsf{PK}(\rho) : \; c/G = H^{\rho}\}$, we convince the verifier that c is indeed a commitment to 0 or 1; were we able to successfully run this protocol with a commitment c not containing 0 or 1 respectively, we could compute the discrete logarithm between G and H which we assumed to be unknown.

Combining this technique with logical disjunction, we can prove a case distinction on whether a bit is 0 or 1 without revealing the actual value.

Range proofs. In known-order groups, we need to show that a committed number lies in a certain interval [22,33]. This prevents a malicious party from exploiting the computation modulo group order in exponents when dealing with commitments. A range proof is denoted by $\{\mathsf{PK}(\alpha_i) : \; c = g^{\alpha} \prod_{i=2}^{m} g_i^{\alpha_i} \wedge A < \alpha < B\}$ for integers A, B. For readability, the range and the representation part of the proof can be separated.

Given a representation $y = g^x \prod_{i=2}^{m} g_i^{x_i} \mod p$, ℓ_1, and ℓ_2, the following protocol convinces the verifier that $2^{\ell_1} - 2^{\epsilon\ell_2+2} \leq x \leq 2^{\ell_1} + 2^{\epsilon\ell_2+2}$. The prover chooses $z \in_{\mathrm{R}} \{-2^{\epsilon\ell_2}, \ldots, 2^{\epsilon\ell_2}\}$, $z_i \in_{\mathrm{R}} \mathbb{Z}_q$ and sends $t := g^z \prod_{i=2}^{m} g_i^{z_i} \mod p$ to the verifier. The verifier chooses $ch \in_{\mathrm{R}} \{0, 1\}$ as challenge. The prover answers $s := z - ch \cdot (x - 2^{\ell_1})$ in $\mathbb{Z}$ and $s_i := z_i - ch \cdot (x_i - 2^{\ell_1}) \mod q$. The verifier checks $-2^{\epsilon\ell_2+1} \overset{?}{<} s \overset{?}{<} 2^{\epsilon\ell_2+1}$ and $t \overset{?}{=} g^{s-c2^{\ell_1}} \cdot \prod_{i=2}^{m} g_i^{s_i - c2^{\ell_1}} y^c \mod p$.

The relation $\epsilon\ell_2 + 2 < \log q$ should hold as otherwise, every value will lie within the range. $\ell_1 < \log q$ is an offset and can also be zero.

This proof is repeated ℓ-times to achieve a soundness error of $2^{-\ell}$.

Equality of discrete logarithms. The protocols so far only dealt with linear relations. However, proving, for instance, multiplication requires us to cope with non-linear relations of secrets. Proofs of equality of discrete logarithms [24] are denoted $\left\{\mathsf{PK}((\alpha_i)_{i=1,...,m}, (\beta_j)_{j=1,...,n}) : \; \alpha_1 = \beta_1 \wedge y_1 = \prod_{i=1}^{m} g_i^{\alpha_i} \wedge y_2 = \prod_{j=1}^{n} h_j^{\beta_j}\right\}$.

Equality is shown with respect to the bases g_1 and h_1 in the representation $(x_1, ..., x_m)$ and $(z_1, ..., z_n)$ of elements y_1 and y_2 to the bases $(g_1, \ldots, g_m)$ and

$(h_1, \ldots, h_n)$ respectively. The prover chooses $r_1, \ldots, r_m, u_2, \ldots, u_n \in_{\mathrm{R}} \mathbb{Z}_q$ and sends $t_1 := \prod_{i=1}^{m} g_1^{r_1}$ and $t_2 := h_1^{r_1} \cdot \prod_{j=2}^{n} h_j^{u_j}$ to the verifier. The prover answers with $s_i := r_i - ch \cdot x_i \mod q$ and $v_j := u_j - ch \cdot z_j \mod q$, $i = 1, \ldots, m$ and $j = 2, \ldots, n$. The verifier accepts iff $t_1 = y_1^{ch} \prod_{i=1}^{m} g_i^{s_i}$ and $t_2 = y_2^{ch} h_1^{s_1} \prod_{j=2}^{n} h_j^{v_j}$.

Arithmetic operations. We compute certain arithmetic operations in zero-knowledge, i.e., hiding some of the witnesses. In particular, given commitments $c_a = G^a \cdot H^{r_a}$, $c_b = G^b \cdot H^{r_b}$, $c_d = G^d \cdot H^{r_d}$, and $c_n = G^n \cdot H^{r_n}$, we can prove that $[\![c_d]\!] \equiv [\![c_a]\!] + [\![c_b]\!] \mod [\![c_n]\!]$, $[\![c_d]\!] \equiv [\![c_a]\!] \cdot [\![c_b]\!] \mod [\![c_n]\!]$, and $[\![c_d]\!] \equiv [\![c_a]\!]^{[\![c_b]\!]} \mod [\![c_n]\!]$. The protocols are defined as follows:

$$
\begin{aligned}
&[\![c_d]\!] \equiv_{[\![c_n]\!]} [\![c_a]\!] + [\![c_b]\!] := \\
&\{\mathsf{PK}(\alpha, \beta, \delta, \eta, \chi, \rho) : \\
&\qquad [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge [\![c_d]\!] = \delta \wedge [\![c_n]\!] = \eta \wedge -2^{\ddot{\ell}} < \alpha, \beta, \delta, \eta < 2^{\ddot{\ell}} \quad (a) \\
&\qquad c_d/(c_a c_b) = c_n^{\chi} h^{\rho} \wedge -2^{\ddot{\ell}} < \chi < 2^{\ddot{\ell}} \; \} \quad (b)
\end{aligned}
$$

$$
\begin{aligned}
&[\![c_d]\!] \equiv_{[\![c_n]\!]} [\![c_a]\!] \cdot [\![c_b]\!] := \\
&\{\mathsf{PK}(\alpha, \alpha', \beta, \delta, \eta, \chi, \rho : \\
&\qquad [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge [\![c_d]\!] = \delta \wedge [\![c_n]\!] = \eta \wedge -2^{\ddot{\ell}} < \alpha, \beta, \delta, \eta < 2^{\ddot{\ell}} \quad (a) \\
&\qquad c_d = c_b^{\alpha'} c_n^{\chi} h^{\rho} \wedge -2^{\ddot{\ell}} < \chi < 2^{\ddot{\ell}} \; \} \wedge \alpha = \alpha' \quad (b)
\end{aligned}
$$

Let us argue why the desired relations should hold: Due to the range proofs in both protocols, using standard techniques, we can extract witnesses $\hat{a}, \hat{r_a}, \hat{b}, \hat{r_b}, \hat{d}, \hat{r_d}, \hat{n}, \hat{r_n}, \hat{x}, \hat{r}$ such that

$$
\begin{aligned}
&c_a = G^{\hat{a}} H^{\hat{r_a}} \wedge c_b = G^{\hat{b}} H^{\hat{r_b}} \wedge c_d = G^{\hat{d}} H^{\hat{r_d}} \wedge c_n = G^{\hat{n}} H^{\hat{r_n}} \\
&\wedge -2^{\ddot{\ell}} < \hat{a}, \hat{b}, \hat{d}, \hat{n}, \hat{x} < 2^{\ddot{\ell}}
\end{aligned}
$$

In the addition proof, we additionally have that

$$c_d/(c_a c_b) = c_n^{\hat{x}} H^{\hat{r}}$$

Since the discrete logarithm between G and H is unknown, we can deduce

$$
\begin{aligned}
&G^{\hat{d}-(\hat{a}+\hat{b})} = G^{\hat{n}\cdot\hat{x}} \mod p \\
\Leftrightarrow\; &\hat{d} - (\hat{a} + \hat{b}) = \hat{x} \cdot \hat{n} \mod q \\
\Leftrightarrow\; &\hat{d} - (\hat{a} + \hat{b}) = \hat{x} \cdot \hat{n}
\end{aligned}
$$

where the last equivalence holds due to the range constraints imposed by the range proofs.

For the multiplication proof, we also have that

$$c_d = c_b^{\hat{a}} c_n^{\hat{x}} h^{\hat{r}}$$

Using the same argument as above, we can deduce that

$$
\begin{aligned}
&G^{\hat{d}} = G^{\hat{a}\cdot\hat{b}+\hat{x}\cdot\hat{n}} \mod p \\
\Leftrightarrow\; &\hat{d} - \hat{a} \cdot \hat{b} = \hat{x} \cdot \hat{n} \mod q \\
\Leftrightarrow\; &\hat{d} - \hat{a} \cdot \hat{b} = \hat{n} \cdot \hat{x}
\end{aligned}
$$

where the last equivalence again holds due to the range constraints.

The exponentiation protocol uses all previously introduced proofs as building blocks. The main idea is to use the square and multiply algorithm to step by step compute the result of the exponentiation where $\ell_b = |b|$ is the bit length of b.

$$
\begin{array}{ll}
\multicolumn{2}{l}{[\![c_d]\!] \equiv_{[\![c_n]\!]} [\![c_a]\!]^{[\![c_b]\!]} :=} \\
\multicolumn{2}{l}{\{\mathsf{PK}(\alpha, \beta, \delta, \eta, \chi, (\rho_i)_{i=1,\ldots,5}) :} \\
\quad [\![c_a]\!] = \alpha \wedge [\![c_b]\!] = \beta \wedge [\![c_d]\!] = \delta \wedge [\![c_n]\!] = \eta \wedge -2^{\ddot{\ell}} < \alpha, \delta, \eta < 2^{\ddot{\ell}} & (a) \\
\quad \left(\prod_{i=0}^{\ell_b - 1} (c_{b_i})^{2^i}\right) / c_b = H^{\rho_5} \wedge & (b) \\
\quad [\![c_{v_1}]\!] \equiv_{[\![c_n]\!]} [\![c_a]\!] \cdot [\![c_a]\!] \wedge \ldots \wedge [\![c_{v_{\ell_b - 1}}]\!] \equiv_{[\![c_n]\!]} [\![c_{v_{\ell_b - 2}}]\!] \cdot [\![c_{v_{\ell_b - 2}}]\!] \wedge & (c) \\
\quad [\![c_{u_1}]\!] = \mu_1 \wedge \ldots \wedge [\![c_{u_{\ell_b - 2}}]\!] = \mu_{\ell_b - 2} \wedge -2^{\ddot{\ell}} < \mu_1, \ldots, \mu_{\ell_b - 2} < 2^{\ddot{\ell}} \wedge & (d) \\
\quad \left((c_{b_0} = H^{\sigma_0} \wedge c_{u_0}/G = H^{\tau_0}) \vee \left(c_{b_0}/G = H^{\vartheta_0} \wedge c_{u_0}/c_a = H^{\psi_0}\right)\right) \wedge & (e) \\
\quad \bigwedge_{i=1}^{\ell_b - 2} \left(\begin{array}{l} \left(c_{b_i} = H^{\vartheta_i} \wedge c_{u_i}/c_{u_i - 1} = H^{\tau_i}\right) \vee \\ \left(c_{b_i}/G = H^{\vartheta_i} \wedge [\![c_{u_i}]\!] \equiv_{[\![c_n]\!]} [\![c_{u_i - 1}]\!] \cdot [\![c_{v_i}]\!]\right) \end{array} \right) \wedge & (f) \\
\quad \left(\begin{array}{l} \left(c_{\ell_b - 1} = H^{\sigma_{\ell_b - 1}} \wedge c_d / c_{u_{\ell_b - 2}} = H^{\tau_{\ell_b - 1}}\right) \vee \\ \left(c_{\ell_b - 1}/G = H^{\sigma_{\ell_b - 1}} \wedge [\![c_d]\!] \equiv_{[\![c_n]\!]} [\![c_{u_{\ell_b - 2}}]\!] \cdot [\![c_{v_{\ell_b - 1}}]\!]\right) \end{array} \right) \} & (g)
\end{array}
$$

(a) shows that we know how to open the commitments and that the committed values are in appropriate ranges. (b) shows that the commitments c_{b_i} are a base for a binary representation of c_b similar to the partial release of secrets protocol. We do not prove that each c_{b_i} is a commitment to either 0 or 1 as this is done in steps $(e - g)$. (c) shows that the c_{v_i} are commitments to powers of $[\![c_a]\!]$, namely, $[\![c_{v_i}]\!] \equiv_{[\![c_n]\!]} [\![c_a]\!]^{2^i}$. We do not state the range proofs explicitly as they are included in the individual multiplication proofs. (d) shows that the intermediate results of the square and multiply algorithm are well-formed. $(e - g)$ prove the actual computation. (e) shows that $[\![c_{b_0}]\!] = 0$ and $[\![c_{u_0}]\!] = 1$ or $[\![c_{b_0}]\!] = 1$ and $[\![c_{u_0}]\!] = [\![c_a]\!]$. (f) is similar to (e) but if $[\![c_{b_i}]\!] = 1$, then current intermediate result $[\![c_{u_i}]\!] \equiv_{[\![c_n]\!]} [\![c_{u_{i-1}}]\!] \cdot [\![c_{v_i}]\!]$ equals the previous intermediate result multiplied with the appropriate power of $[\![c_a]\!]$. Finally, (g) proves that the last computation step results in $[\![c_d]\!]$.

Fiat-Shamir heuristic and random oracles. All the protocols shown so far were interactive *Σ*-protocols. Ultimately, our goal is to generate a non-interactive proof, i.e., a proof comprised of one single message sent from the prover to the verifier. To achieve this goal, we use the Fiat-Shamir heuristic and introduce the random oracle model.

The idea is that cryptographic hash-functions such as SHA-1 return, upon a query, a truly uniformly random string but answer consistently with previous queries. Since the only verifier message is one random string, we can make a *Σ*-protocol non-interactive by computing the hash-value of the statement and the first message of the prover.

We stress that our (interactive) protocol is secure without random oracles; we merely rely on random oracles to make our protocol non-interactive.

And: (P, S, V)
Common input: statements x_1, x_2
Private input: witnesses w_1, w_2

$$\begin{array}{lcl}
P^{(w_1,w_2)} & & V \\
com_i \leftarrow P_i^{w_i} & \xrightarrow{(com_1,com_2)} & \\
 & \xleftarrow{ch} & ch \in_{\mathrm{R}} CS \\
resp_i \leftarrow P_i^{w_i}(ch) & \xrightarrow{(resp_1,resp_2)} & \\
 & & V_i(com_i, ch, resp_i) \overset{?}{=} \mathsf{true},\ i = 1, 2
\end{array}$$

$S(ch)$:
$(com_i, resp_i) \leftarrow S_i(ch)$, return $((com_1, com_2), ch, (resp_1 resp_2))$

Fig. 8. Logical And of two sub proofs

Construction of And and Or proofs. We give the construction of the logical And in Figure 8 and Or in Figure 9 [26] for the case of two sub protocols (P_1, V_1, S_1) and (P_2, V_2, S_2) where S_i denotes the simulator for the according scheme. We assume that the sub protocols are SHVSZK proofs with special soundness.

We require that the prover returns, upon their first call, the commitment message and upon the second call the response to the given challenge. The verifier returns a random challenge and verification requires the commitment, the challenge, and the response as input. The simulator returns, given a challenge, a valid simulation for that challenge, i.e., the simulator's output verifies using the according verifier. We write P^w to denote that P has access to witness w.

As range proofs only have a binary challenge space, a challenge ch is translated into C many runs where the i-th run uses the i-th bit of ch.

Notice that the simulation for And and Or is only possible because we can give the simulator a specific challenge.

Since both schemes for And have special soundness and a change in the challenge is propagated to both schemes, the And also has special soundness.

Before arguing about the special soundness of the Or protocol, we note that once the commitment has been sent, the challenge for the second scheme is fixed (otherwise, P_2 can compute a witness due to the special soundness of the second scheme). Hence, if we change the challenge ch, we also change the challenge ch_1 to for the first scheme. Since this scheme has special soundness, the construction of Or also has special soundness (we only need to compute one witness).

Or: (P, S, V)
Common input: statements x_1, x_2
Private input: w.l.o.g. witnesses w_1

P^{w_1}		V
$ch_2 \in_{\mathrm{R}} CS$		
$(com_2, rep_2) \leftarrow S_2(ch_2)$		
$com_1 \leftarrow P_1^{w_1}$	$\xrightarrow{(com_1, com_2)}$	
	$\xleftarrow{ch}$	$ch \in_{\mathrm{R}} CS$
$ch_1 := ch - ch_2 \mod 2^C$		
$resp_1 \leftarrow P_1^{w_1}(ch_1)$	$\xrightarrow{(resp_1, ch_1, resp_2, ch_2)}$	
		$ch \overset{?}{=} ch_1 + ch_2 \mod 2^C$
		$V_i(com_i, ch_i, resp_i) \overset{?}{=} \mathsf{true}$
		$i = 1, 2$

$S(ch)$:
$ch_1 \in_{\mathrm{R}} CS$
$ch_2 := ch - ch_1 \mod 2^C$
$(com_i, resp_i) \leftarrow S_i(ch_i)$, return $((com_1, com_2), ch, ((resp_1, ch_1), (resp_2, ch_2)))$

Fig. 9. Logical Or of two sub proofs

D Applied pi-calculus model

We report the applied pi-calculus model used for verifying the trust and anonymity properties of our protocol (for certificate chains of length 2). Table 1 contains the model of our protocol and the trust policy, while Table 2 shows the processes capturing the observational equivalence relation verified by ProVerif. Notice that ProVerif is able to analyze biprocesses, i.e., processes sharing the same structure and differing just in their terms. As usual, the biprocess $P(\text{choice}[x, y])$ denotes the observational equivalence relation $P(x) \approx P(y)$.

```
free c. (* communication channel *)
private free d. (* synchronization channel for public keys *)

private reduc pkey(pk(x)) = true.
...
(* standard equations for cryptographic messages and zero-knowledge omitted *)

query ev:AUTH(id2,x) ==>
      (ev:SEND(id1,id2,x) & ev:TRUST(id2,id3) & ev:TRUST(id3,id1)) |
      (ev:TRUST(id2,id3) & ev:TRUST(id3,id1) & ev:COMPROMISE(id1)) |
      (ev:TRUST(id2,id3) & ev:COMPROMISE(id3)).

(* Principals release certificates as dictated by the attacker *)

let auth = in(c,(xsig2,xpk2,xsig3,pk1));
           let z=pkey(pk1) in
           new r1; new m1;
           let sigm1 = sign(hash(m1),sk(k)) in
           let xzk1 = zk(sigm1,pk(k),xsig3,xpk2,xsig2,r1;hash(m1),pk1;proof) in
           event SEND(pk(k),pk1,m1);
           out(c,(xzk1,m1)).


(* The verifier receives and checks the zero-knowledge proof*)

let ver = in(c,(x,y));
          if zkver(6;2;proof;x) = true then
          if public2(x)=pk(k) then
          let z = public1(x) in
          if z=hash(y) then
          AUTH(pk(k),z).


(* Honest principals *)

let peer =
      new k;
      !out(d,pk(k)) | (out(c,pk(k));
      (!in(c, x);
      let xi = pkey(x) in
      in(d,=x);
      event TRUST(pk(k),x);
      out(c, sign(x, sk(k))))) | auth | ver.

(* Compromised principals *)

let keygencompromise =
      new k;
      event COMPROMISE(pk(k));
      !out(d,pk(k)) | out(c,k).


(* Statement of our proof *)

define proof = land(
                  land(
                     check(alpha1,beta1,alpha2), check(alpha3,alpha2,alpha4)
                  ), check(alpha5,alpha4,beta2)
               ) .

process ( !keygencompromise | !peer).
```
Table 1. Applied pi-calculus model with trust policy as described in Section 5

```
free c.

(* Equations for cryptographic messages as in Table 1 *)

(* Phase 1: the attacker generates the Web of Trust *)

let keygen =
 new k;
 out(c,pk(k));
 !in(c, x);
 let xi = pkey(x) in
 out(c, sign(x, sk(k))).

(* Phase2 : Each peer generates a key-pair, publishes the public key,
and sign the public keys chosen by the attacker *)

let signing1 =  out(c,pk(k1));
                !in(c, x);
                let xi = pkey(x) in
                out(c, sign(x, sk(k1))).

let signing2 = out(c,pk(k2));
               !in(c, x);
               let xi = pkey(x) in
               out(c, sign(x, sk(k2))).

(* Phase 3:
The attacker chooses a key-chain and each peer generates the corresponding proof.
If the two proofs are both valid and are addressed to the same verifier,
then the attacker does not know which of the two peers is authenticating *)

let auth =  in(c,(xsig2,xpk2,xsig3));
            in(c,(ysig2,ypk2,ysig3));

            // first peer
            new r1; new m1;
            let sigm1 = sign(hash(m1),sk(k1)) in
            let xzk1 = zk(sigm1,pk(k1),xsig3,xpk2,xsig2,r1;hash(m1),pk1;proof) in

            // second peer
            new r2; new m2;
            let sigm2 = sign(hash(m2),sk(k2)) in
            let yzk2 = zk(sigm2,pk(k2),ysig2,ypk2,ysig2,r2;hash(m2),pk1;proof) in

            // both proofs are valid
            if zkver(6;2;proof;xzk1) = true then
            if zkver(6;2;proof;yzk2) = true then
            out(c,choice[(xzk1,m1),(yzk2,m2)]).

(* Statement of our proof as in Table 1 *)

process ( !keygen | new k1; new k2;(signing1 | signing2 |
               in(c,pk1);let xtemp = pkey(pk1) in auth)).
```

Table 2. Applied pi-calculus model of the anonymity game described in Section 5. P(choice[x,y]) $\triangleq$ P(x) $\approx$ P(y)