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 [13], 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 [14] 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, and the public key pk_3 of the intended recipient. We provide a prototypical

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 [18].

[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 [29]. A non-interactive zero-knowledge proof is a zero-knowledge protocol consisting of one message sent by the prover to the verifier.

implementation of our protocol as an extension of the OpenPGP standard. The tool is freely available at [6].
- 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 [3], 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 [11], 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 [9]. 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 [19,37,5,10] 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 [27,12], where anonymity could be obtained by replacing user identifiers with generic anonymous attributes.

Ring signature schemes [34,31,32] 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 [25]. Several techniques have been proposed to keep the social network graph private while enforcing access control policies based on trust degrees [22,21,41]. In contrast to our approach, the proposed protocols are interactive, similar to the delegatable anonymous credential scheme [9]. In other works, trust relationships are instead assumed to be public, e.g., [35,4,17]. 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 [38], public keys and attached certificates are uploaded on key servers and are thus publicly available. Finally, the recently proposed Lockr protocol [40] 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 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. The full-version of this paper is available at [6].

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 pk_u and a secret key sk_u. Trust is distributed via certificates. User u expresses her belief that a given public key pk_v actually belongs to user v by signing 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 $(\mathsf{pk}_\mathsf{A}, \mathcal{A}_\mathsf{A})$ to $(\mathsf{pk}_\mathsf{B}, \mathcal{A}_\mathsf{B})$, thus establishing a trust relation between those keys.

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*

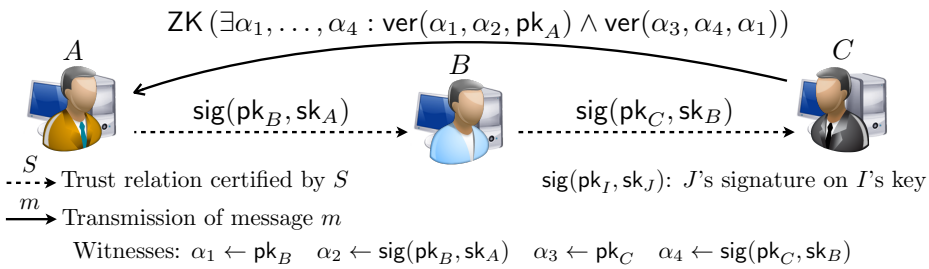

Fig. 1. Protocol for anonymous proof of a certificate chain of length 2

$\ell - 1$, *where* $\mathsf{C}_i = \mathit{sig}((\mathsf{pk}_{i+1}, \mathcal{A}_{i+1}), \mathsf{sk}_i)$ *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* $\mathit{sig}((\mathcal{A}_1, \mathsf{pk}_1), \mathsf{sk}_1)$.

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]

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

[6] Here and throughout the paper, we use the convention introduced in [16] that Greek letters denote those values that are kept secret by the proof.

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.

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 [14] while, for proving statements about certificate chains, we propose a novel non-interactive zero-knowledge proof of knowledge based on Σ-protocols [30]. We first review the basic building blocks and subsequently describe the construction of our zero-knowledge proof scheme.

3.1 Camenisch-Lysyanskaya Signature

This signature scheme was introduced in [14] 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 and a, b, c are random elements from a large subgroup of $\mathbb{Z}_n^*$. 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.

3.2 Zero-Knowledge Proofs and Σ-Protocols

Zero-knowledge proofs were first introduced in [30] 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 [28,20], 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.

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*, *challenge*, and *response*, 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 [28,20].

Special soundness is a strong form of *proof of knowledge* and guarantees that a prover is in possession of a witness. 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 [20], Σ-protocols can be combined together to prove logical conjunctions and disjunctions of their respective statements.

[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 [24], which lets the prover herself choose the challenge by using the random oracle, without interacting with the verifier.

Lemma 1 (Logical Combination of Σ-protocols [20]). *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.

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 [33]. In the following, we let $[\![c]\!]$ denote the value committed to in c.

Range proofs. We use the range proofs proposed in [26]. 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 [15] 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

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

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 $\alpha_{\mathsf{pk}} = (\alpha, \beta, \gamma, \eta)$.

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

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 [24] to make our protocol non-interactive.

3.4 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 [36]. The implementation is in Java and comprises roughly 6000 lines of code. A prototypical implementation is freely available at [6].

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. 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 $\{\mathsf{PK}(\epsilon) : [\![c_e]\!] = \epsilon \wedge \mathit{current\ date} < \epsilon < \mathit{ub}\}$ given a commitment c_e on the expiration date attribute ϵ and a suitable upper bound ub for all possible key expiration dates.

Notice that the OpenPGP standard [13] incorporates a key revocation mechanism, which is implemented by a special signature (also called revocation sig-

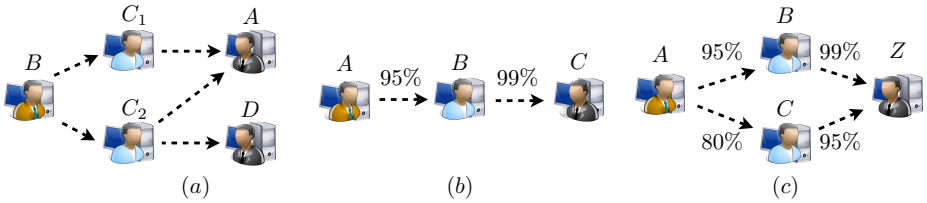

Fig. 2. Webs of trust

nature) 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 2 (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 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 [18]. The examples in this section are intentionally borrowed from [18] in order to show the applicability of our framework to existing trust models. Consider the web of trust in Figure 2 (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 [18] 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 2 (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, 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 2 (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 [1], formalizing the trust property as an authorization policy and the anonymity property as an observational equivalence relation, and verifying our model with ProVerif [11,2], 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 [7], for which computational soundness results

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

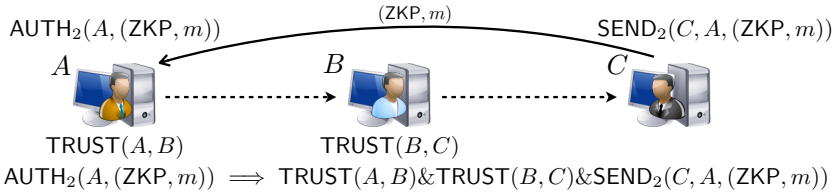

Fig. 3. Trust policy

exist [8]. For easing the presentation, in this section we focus on certificate chains without attributes.

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, 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 3). 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:

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

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

Fig. 4. Anonymity game

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 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 4) 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

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

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 [23]), 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 [23]), 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. M. Abadi and B. Blanchet. Secrecy types for asymmetric communication. In *FOSSACS'01*, volume 2030 of *LNCS*, pages 25–41. Springer, 2001.
2. M. Abadi, B. Blanchet, and C. Fournet. Automated verification of selected equivalences for security protocols. In *LICS'05*, pages 331–340. IEEE, 2005.
3. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01*, pages 104–115. ACM, 2001.
4. R. Ashri, S. D. Ramchurn, J. Sabater, M. Luck, and N. R. Jennings. Trust evaluation through relationship analysis. In *AAMAS'05*, pages 1005–1011. ACM, 2005.
5. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO'00*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.
6. M. Backes, S. Lorenz, M. Maffei, and K. Pecina. Anonymous webs of trust (tool and long version), 2010. Available at `http://www.lbs.cs.uni-sb.de/awot/`.
7. M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *SSP'08*, pages 202–215. IEEE, 2008.
8. M. Backes and D. Unruh. Computational soundness of symbolic zero-knowledge proofs against active attackers. In *CSF'08*, pages 255–269. IEEE, 2008.
9. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO'09*, volume 5677 of *LNCS*, pages 108–125. Springer, 2009.
10. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA'05*, volume 3376 of *LNCS*. Springer, 2005.
11. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *CSFW'01*, pages 82–96. IEEE, 2001.
12. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.
13. J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP message format. In *Request for Comments*, volume 4880. IETF, 2007.
14. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN'02*, volume 2576 of *LNCS*, pages 268–289. Springer, 2002.
15. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *EUROCRYPT'98*, volume 1592 of *LNCS*, pages 107–122. Springer, 1998.
16. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer, 1997.
17. B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *OTM'06*, volume 4278 of *LNCS*, pages 1734–1744. Springer, 2006.
18. G. Caronni. Walking the web of trust. In *WETICE'00*, pages 153–158. IEEE, 2000.
19. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

20. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.
21. J. Domingo-Ferrer, A. Viejo, F. Sebé, and U. González-Nicolás. Privacy homomorphisms for social networks with private relationships. *Computer Networks*, 52(15):3007–3016, 2008.
22. J. Domingo-Ferror. A public-key protocol for social networks with private relationships. In *MDAI'07*, volume 4617 of *LNCS*, pages 373–379. Springer, 2007.
23. facebook. `http://www.facebook.com/`.
24. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'87*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.
25. K. Frikken and P. Srinivas. Key allocation schemes for private social networks. In *WPES'09*, pages 11–20. ACM, 2009.
26. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, 1997.
27. C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT'02*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.
28. O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
29. O. Goldreich, S. Micali, and A. 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.
30. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
31. J. Herranz. Identity-based ring signatures from rsa. *Theoretical Computer Science*, 389(1-2):100–117, 2007.
32. 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/`.
33. T. 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 *LNCS*, pages 129–140. Springer, 1991.
34. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. *Communications of the ACM*, 22(22):612–613, 2001.
35. J. Sabater-Mir. Towards the next generation of computational trust and reputation models. In *MDAI'06*, volume 3885 of *LNCS*, pages 19–21. Springer, 2006.
36. M. Scott. Multiprecision Integer and Rational Arithmetic C/C++ Library. `http://www.shamus.ie/`.
37. D. X. Song. Practical forward secure group signature schemes. In *CCS'01*, pages 225–234. ACM, 2001.
38. The GNU Privacy Guard Team. GnuPG. `http://www.gnupg.org/`.
39. The GNU Privacy Guard Team. The GNU Privacy Handbook. `http://www.gnupg.org/gph/en/manual.pdf`.
40. A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: better privacy for social networks. In *CoNEXT'09*, pages 169–180. ACM, 2009.
41. D.-W. Wang, C.-J. Liau, and T.-S. Hsu. Privacy protection in social network data disclosure based on granular computing. In *Fuzzy'06*, pages 997 – 1003. IEEE, 2006.

Fig. 5. Trust signature chain

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 [39]: *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 [13], 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 5 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 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.