Design and Verification of Anonymous Trust Protocols*

Michael Backes[1,2] and Matteo Maffei[1]

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

Abstract. Over the last years, the Web has evolved into the premium forum for freely and anonymously disseminating and collecting information and opinions. However, 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.

In this paper we propose a methodology for designing security protocols that enforce fine-grained trust policies while still ensuring the anonymity of the users. The fundamental idea of this methodology is to incorporate non-interactive zero-knowledge proofs: the trust level of users are certified using digital signatures, and users assert their trust level by proving in zero-knowledge the possession of such certificates. Since the proofs are zero-knowledge, they provably do not reveal any information about the users except for their trust levels; in particular, the proofs hide their identities.

We additionally propose a technique for verifying the security properties of these protocols in a fully automated manner. We specify protocols in the applied pi-calculus, formalize trust policies as authorization policies, and define anonymity properties in terms of observational equivalence relations. The verification of these properties is then conducted using an extension of recently proposed static analysis techniques for reasoning about symbolic abstractions of zero-knowledge proofs.

1 Introduction

Over the last years, the Web has evolved into the premium forum for freely disseminating and collecting information and opinions. In particular, social networks and peer-to-peer (P2P) applications have proven to be particularly salient approaches for this task. However, not all information providers 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.

* Work partially supported by the initiative for excellence of the German federal government, by DFG Emmy Noether program, and by MIUR project "SOFT".

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. Trust management systems have become the most popular technique to determine which resources should be trusted, and to which extent. Except for some notable exceptions [1], trust management systems inherently rely on revealing the identities of the involved parties, and they hence do not live up to nowadays' anonymity demands. Indeed, devising security protocols that simultaneously satisfy conflicting security properties such as trust and anonymity requires significant extensions to the state-of-the-art. In particular, it is crucial to incorporate the most innovative modern cryptographic primitive in the design and the verification of security protocols: zero-knowledge proofs[1] [2]. This primitive goes beyond the traditional understanding of cryptography that only ensures secrecy and authenticity of a communication. The unique security features of zero-knowledge proofs, combined with the recent advent of efficient cryptographic implementations of this primitive for special classes of problems, have paved the way for their deployment in modern applications, such as anonymity protocols [3,1] and electronic voting protocols [4,5].

In this paper we propose a methodology based on zero-knowledge proofs for *designing* security protocols that enforce fine-grained trust policies while still ensuring the anonymity of the user. The fundamental idea of this methodology is to exploit digital signatures for certifying the trust level of users, and to let users prove the possession of such certificates in order to prove their trust level. Since the proof is zero-knowledge, it provably does not reveal any information about the users except for their trust levels; in particular, the proof hides their identities.

We additionally propose a technique for *verifying* the security properties of these protocols in a fully automated manner. We specify protocols in the applied pi-calculus [6], formalize trust policies as authorization policies, and define anonymity properties in terms of observational equivalence relations. The verification of these properties is then conducted using an extension of recently proposed static analysis techniques for reasoning about symbolic abstractions of zero-knowledge proofs [7,8].

2 Anonymous Proofs of Trust

We assume a public-key infrastructure and that users know each other's public key. Whenever user A wants to certify that she trusts user B, A signs B's verification key, thus obtaining the digital signature $\mathsf{sign}(\mathsf{vk}(k_B), k_A)$. Suppose now that B wants to send a message m to A in an authenticated manner, but

[1] 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.

without revealing his identity, i.e., A should solely learn that this message was generated by a trusted user. This requirement clearly prevents B from simply sending a signature on m to A, since this would reveal his identity. In order to guarantee both trust (authentication) and anonymity, B instead runs a zero-knowledge proof showing that B knows a verification key signed by A as well as the corresponding (private) signing key, thus preventing impersonation attacks. Following [8], we represent this zero-knowledge proof as the following term:

$$\mathsf{zk}_{\mathsf{ver}(\alpha_1,\alpha_2,\beta_1)\wedge\alpha_2=\mathsf{vk}(\alpha_3)}(\overbrace{\mathsf{sign}(\mathsf{vk}(k_B),k_A)}^{\alpha_1},\overbrace{\mathsf{vk}(k_B)}^{\alpha_2},\overbrace{k_B}^{\alpha_3};\overbrace{\mathsf{vk}(k_A)}^{\beta_1},m) \qquad (1)$$

We briefly describe the individual parts of this zero-knowledge proof:

– *Involved message components*: The messages $\mathsf{sign}(\mathsf{vk}(k_B),k_A)$, $\mathsf{vk}(k_B)$, and k_B constitute the *private component* of the proof; the semantics of [8] ensures that they are not revealed to the verifier. The messages $\mathsf{vk}(k_A)$ and m constitute the *public component*; they are revealed to the verifier. This proof hence does not reveal any information except for A's verification key and the message m to be authenticated; in particular, A does not learn the identity of B since B's verification key $\mathsf{vk}(k_B)$ is kept secret.
– *Proven Statement*: The statement to be proven is expressed as a Boolean formula $\mathsf{ver}(\alpha_1,\alpha_2,\beta_1)\wedge\alpha_2=\mathsf{vk}(\alpha_3)$ over cryptographic operations. Here α_i and β_j constitute placeholders for the i-th element in the private component and the j-th element in the public component, respectively. We moreover write $\mathsf{ver}(\alpha_1,\alpha_2,\beta_1)$ as an abbreviation of $\exists k\colon (\alpha_1=\mathsf{sign}(\alpha_2,k)\wedge\beta_1=\mathsf{vk}(k))$, i.e., to denote that the A's signature on B's verification key is valid. In words, the statement hence says "B knows a signature α_1 of a message α_2 that can be checked using A's verification key $\mathsf{vk}(k_A)$, as well as the private key α_3 corresponding to the verification key α_2". Proving the knowledge of a certificate for B's verification key (without revealing the key and hence the identity of B) and the knowledge of the corresponding private signing key, however, ensures A that the message m in the public component comes from a trusted user[2].

Using zero-knowledge proofs in this manner constitutes a general approach for asserting trust, and it in particular allows us to implement fine-grained trust policies. For instance, B might be interested in proving that he is considered trusted by C_1 or C_2, without revealing which user C_i it is trusted by. Assume further that both C_1 and C_2 are trusted by A. Such disjunctive proofs aree viable tools for enhancing anonymity, e.g., to deal with the case that A asks C_1 and C_2 for a list of trusted users. This proof can be realized by the following zero-knowledge proof, assuming that A has been certified by C_1:

[2] Technically, we consider (an abstraction of) non-malleable zero-knowledge proofs, i.e., proofs that the adversary cannot modify without knowing the secret witnesses.

$$\mathsf{zk}_{\mathsf{ver}(\alpha_1,\alpha_2,\alpha_4)\wedge\alpha_2=\mathsf{vk}(\alpha_3)\wedge\alpha_4\in\{\beta_1,\beta_2\}}\left(\begin{array}{c}\overbrace{\mathsf{sign}(\mathsf{vk}(k_B),k_{C_1})}^{\alpha_1},\overbrace{\mathsf{vk}(k_B)}^{\alpha_2},\overbrace{k_B}^{\alpha_3},\overbrace{\mathsf{vk}(k_{C_1})}^{\alpha_4};\\ \overbrace{\mathsf{vk}(k_{C_1})}^{\beta_1},\overbrace{\mathsf{vk}(k_{C_2})}^{\beta_2},\overbrace{m}^{\beta_3}\end{array}\right)$$

Another interesting case is when B wants to prove that he has been certified by both C_1 and C_2. Such conjunctive proofs are useful when trust policies take into account multiple trust certifications to upgrade the trust level of a user. This can be realized by the following zero-knowledge proof, which guarantees that the same (secret) verification key α_2 is used in the two (secret) certificates α_1 and α_4:

$$\mathsf{zk}_{\substack{\mathsf{ver}(\alpha_1,\alpha_2,\beta_1)\wedge\alpha_2=\mathsf{vk}(\alpha_3)\\ \wedge\,\mathsf{ver}(\alpha_4,\alpha_2,\beta_2)}}\left(\begin{array}{c}\overbrace{\mathsf{sign}(\mathsf{vk}(k_B),k_{C_1})}^{\alpha_1},\overbrace{\mathsf{vk}(k_B)}^{\alpha_2},\overbrace{k_B}^{\alpha_3},\overbrace{\mathsf{sign}(\mathsf{vk}(k_B),k_{C_2})}^{\alpha_4};\\ \overbrace{\mathsf{vk}(k_{C_1})}^{\beta_1},\overbrace{\mathsf{vk}(k_{C_2})}^{\beta_2},\overbrace{m}^{\beta_3})\end{array}\right)$$

3 Automated Verification of Proofs of Trust

Trust policies can be naturally formalized as authorization policies. For instance, consider the protocol described before, where A sends to B a certificate and B authenticates the message m with the zero-knowledge proof (1):

$$\begin{tikzcd}
A & & B \\
\mathsf{assume}\ \mathrm{Trust}(A,B) & & \\
{} \arrow[rr, "\mathsf{sign}(\mathsf{vk}(k_B),k_A)"] & & {} \\
& & \mathsf{assume}\ \mathrm{Send}(B,A,m) \\
{} & & {} \arrow[ll, "ZK"'] \\
\mathsf{assert}\ \mathrm{Authenticate}(A,m) & &
\end{tikzcd}$$

$$\text{where } ZK=\mathsf{zk}_{\mathsf{ver}(\alpha_1,\alpha_2,\beta_1)\wedge\alpha_2=\mathsf{vk}(\alpha_3)}\left(\begin{array}{c}\overbrace{\mathsf{sign}(\mathsf{vk}(k_B),k_A)}^{\alpha_1},\overbrace{\mathsf{vk}(k_B)}^{\alpha_2},\overbrace{k_B}^{\alpha_3};\\ \overbrace{\mathsf{vk}(k_A)}^{\beta_1},m\end{array}\right)$$

We decorate security-related protocol events with *assumptions* and *assertions*. In the example, A assumes $\mathrm{Trust}(A,B)$ before certifying B. Moreover, B assumes $\mathrm{Send}(B,A,m)$ before sending the zero-knowledge proof to A. After verifying the zero-knowledge proof, A finally asserts $\mathrm{Authenticate}(A,m)$. We say that a protocol is safe if and only if in all protocol executions, even in the presence of an active attacker, every assertion is entailed by the previous assumptions and by the authorization policy. Formally, this is captured by the following authorization policy:

$$\forall A, B, m.\text{Trust}(A, B) \wedge \text{Send}(B, A, m) \Rightarrow \text{Authenticate}(A, m).$$

This policy asserts that A can authenticate message m (assertion Authenticate(A, m)) provided that this message has been sent by a user B (assumption Send(B, A, m)) that is trusted by A (assumption Trust(A, B)). We have specified this protocol as a process in the applied pi-calculus [9] and checked with our type system [7] that this protocol is safe with respect to the given authorization policy.

In order to define the anonymity property of the protocol, we consider a system with two users B_1 and B_2 trusted by A. Both of them receive a certificate from A and afterwards one of them authenticates a message with A. Intuitively, this protocol guarantees the anonymity of the sender if A cannot distinguish the process $S[B_1, B_2]$, in which the message is sent by B_1, from the process $S[B_2, B_1]$, in which the message is sent by B_2. This is formalized by requiring

$$S[B_1, B_2] \approx S[B_2, B_1],$$

where $\approx$ denotes the observational equivalence relation in the applied pi-calculus. We automatically checked this equivalence using ProVerif [10].

4 Open Challenges

In this short paper, we conclude by outlining a series of important challenges that have to be tackled for applying the proposed methodology to realistic scenarios.

First, the methodology for asserting trust using zero-knowledge proofs should be comprehensive enough to model popular trust models such as [11,12,13].

Second, currently used zero-knowledge protocols are still notoriously inefficient for many important classes of statements. However, recent results [14] show that it is possible to automatically devise efficient implementations of zero-knowledge proofs such as the one depicted in Equation (1). Enforcing anonymity in complex trust models, however, calls for efficient implementations of even wider ranges of zero-knowledge proofs.

Third, the advent of social network applications resulted in the demand for novel, more comprehensive requirements on both trust and anonymity. So far, many of these requirements lack a formalization and hence corresponding analysis techniques.

References

1. Lu, L., Han, J., Hu, L., Huai, J., Liu, Y., Ni, L.M.: Pseudo trust: Zero-knowledge based authentication in anonymous peer-to-peer protocols. In: Proc. 2007 IEEE International Parallel and Distributed Processing Symposium, p. 94. IEEE Computer Society Press (2007)

2. Goldreich, O., Micali, S., Wigderson, A.: 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), http://www.wisdom.weizmann.ac.il/~oded/X/gmw1j.pdf
3. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proc. 11th ACM Conference on Computer and Communications Security, pp. 132–145. ACM Press (2004)
4. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proc. 4th ACM Workshop on Privacy in the Electronic Society, WPES, pp. 61–70. ACM Press (2005)
5. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: A secure voting system. In: Proc. 29th IEEE Symposium on Security and Privacy, pp. 354–368. IEEE Computer Society Press (2008)
6. Abadi, M., Blanchet, B.: Secrecy Types for Asymmetric Communication. In: Honsell, F., Miculan, M. (eds.) FOSSACS 2001. LNCS, vol. 2030, pp. 25–41. Springer, Heidelberg (2001)
7. Backes, M., Hriţcu, C., Maffei, M.: Type-checking zero-knowledge. In: 15th ACM Conference on Computer and Communications Security, CCS 2008, pp. 357–370. ACM Press (2008), Implementation available at http://www.infsec.cs.uni-sb.de/projects/zk-typechecker/
8. Backes, M., Maffei, M., Unruh, D.: Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In: Proc. 29th IEEE Symposium on Security and Privacy, pp. 202–215. IEEE Computer Society Press (2008)
9. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Proc. 28th Symposium on Principles of Programming Languages, POPL, pp. 104–115. ACM Press (2001)
10. Abadi, M., Blanchet, B., Fournet, C.: Automated verification of selected equivalences for security protocols. In: Proc. 20th Annual IEEE Symposium on Logic in Computer Science, LICS, pp. 331–340. IEEE Computer Society Press (2005)
11. Jøsang, A.: An algebra for assessing trust in certification chains. In: Proceedings of the Network and Distributed Systems Security Symposium, NDSS 1999. The Internet Society (1999)
12. Xiong, L., Ling, L.: A reputation-based trust model for peer-to-peer ecommerce communities (extended abstract). In: Proceedings of the 4th ACM Conference on Electronic Commerce, EC 2003, pp. 228–229. ACM Press (2003)
13. Carbone, M., Nielsen, M., Sassone, V.: A formal model for trust in dynamic networks. In: International Conference on Software Engineering and Formal Methods, SEFM 2003, pp. 54–64 (2003)
14. Bangerter, E., Camenisch, J., Krenn, S., Sadeghi, A., Schneider, T.: Automatic generation of sound zero-knowledge protocols. IACR Cryptology ePrint Archive: Report 2008/471 (2008), http://eprint.iacr.org/