Trust-Rated Authentication for Domain-Structured Distributed Systems

Ralph Holz, Heiko Niedermayer, Peter Hauck, Georg Carle
`lastname@informatik.uni-tuebingen.de`

Wilhelm-Schickard-Institut für Informatik
University of Tübingen, Germany

Abstract. We present an authentication scheme and new protocol for domain-based scenarios with inter-domain authentication. Our protocol is primarily intended for domain-structured Peer-to-Peer systems but is applicable for any domain scenario where clients from different domains wish to authenticate to each other. To this end, we make use of Trusted Third Parties in the form of Domain Authentication Servers in each domain. These act on behalf of their clients, resulting in a four-party protocol. If there is a secure channel between the Domain Authentication Servers, our protocol can provide secure authentication. To address the case where domains do not have a secure channel between them, we extend our scheme with the concept of trust-rating. Domain Authentication Servers signal security-relevant information to their clients (pre-existing secure channel or not, trust, . . .). The clients evaluate this information to decide if it fits the security requirements of their application.

Key words: Authentication, Protocols, PKI, Trust-Rating, Multi-Domain, Distributed Systems, Peer-to-Peer

Acknowledgments This work was in part funded by Landesstiftung Baden-Württemberg in the scope of the Spontaneous Virtual Networks (SpoVNet) project.

1 Motivation

Emerging technologies like Peer-to-Peer networks or Identity Federation have revived interest in concepts for distributed authentication. Identity Federation is a use case where authentication has to be conducted across domain boundaries, i.e. members of one domain wishing to authenticate to members of another domain. Peer-to-Peer networks are often formed in an ad-hoc manner and are decentral by definition. Thus, these networks exhibit the need for distributed authentication even more as no *a priori* context between peers may exist.

[⋆] The original source of this publication is `www.springerlink.com`: `http://www.springerlink.com/content/k6786282r5378k42/`

According to the work of Boyd [1], however, authentication protocols need at least one secure channel in order to be safe against a Dolev-Yao intruder [2]. Boyd proved that where *'a principal has, at some state of the system, established a secure channel with another principal, such a channel must already have existed at all previous states'* [1]. This means that principals cannot establish an authenticated session without having established keys previously in a secure manner. This can only be circumvented by introducing a Trusted Third Party (TTP) which can mediate between the principals (e.g. by certifying their keys) – which effectively introduces secure channels by other means.

Where security is a requirement, Peer-to-Peer networks typically rely on a central entity that acts as a TTP, often as a Certification Authority (CA). Skype [3] uses such an approach. However, such a central component ('server') conflicts with the Peer-to-Peer paradigm. While practical for a single network with one administrative domain, it would probably be utopian to assume a single global entity for all Peer-to-Peer networks and, if the networks are to interoperate, their peers. Moreover, Ellison and Schneier point out that CAs might imply more security than they actually achieve [4], in particular if a single CA is responsible for identity verification on a global scale. Given its limited resources, it may not verify identities carefully enough as a consequence.

Another option that Peer-to-Peer networks may rely on are Web of Trust-like certification chains. However, these are susceptible to the Sybil Attack [5]. The concept can be strengthened by reputation mechanisms such as those for PGP [6] or as described in [7]. While arguably suitable for low-risk use cases, this remains unsatisfactory in general.

It is an interesting observation that most of these solutions may fit well for small Peer-to-Peer networks but become unrealistic for large scenarios (do not scale, cannot verify all identities, etc.). We consider a domain-based Peer-to-Peer network to be a good compromise - a large network consisting of easier-to-secure small networks (each being one domain). Note that such domains may also reflect social and trust relations. This is similar to so-called darknets where peers only connect to their human friends, thus avoiding direct contact with attackers.

Yet, similar to Identity Federation, domain-based solutions still need inter-domain authentication. Of course, Boyd's theorem applies to inter-domain communication as well. Most protocols for inter-domain authentication thus assume the existence of a secure channel to provide their service. The drawback is that they do not cover the use case of domains without previous knowledge of each other. This dilemma can only be mediated, but never resolved completely.

Our contribution is an authentication scheme that explicitly addresses this situation. We present a 4-party authentication protocol for domain scenarios, and extend it with a *Trust-Rating Mechanism* for the Peer-to-Peer use case. The novelty in our scheme is that it covers the case of non-secure channels or untrusted systems by supplying the authenticating principals with means to assess the situation and make a well-founded authentication decision. Terminology-wise, our scheme can also be viewed as a PKI for domain-structured systems without *a priori* knowledge of each other. We have implemented a prototype, but put the focus on the concepts in this paper.

This paper is organized as follows. We describe the scheme and the underlying ideas in Section 2. Section 3 contains the formal description of protocol and protocol goals for the scheme. We verify that the goals are achieved in Section 4 and discuss protocol security in Section 5. Section 6 discusses further aspects. Related work is presented in Section 7.

2 Authentication Scheme With Trust-Rating

In the following, we use the term 'domain' generically to indicate a group of principals that share a common context and desire communication with principals in other domains to be authenticated.

We introduce TTPs in each domain, which we call Domain Authentication Servers (DAS). We structure communication in our scheme by distinguishing between intra-domain and inter-domain communication. The latter takes place on one of two channels: either between two DAS or between two clients. An authentication process between two principals B and A includes activity on the part of their DAS, S_B and S_A. If the channel between S_A and S_B is secure, our protocol (see Section 3) will provide secure authentication, with part of the process delegated to the DAS.

This scheme is now extended with the Trust-Rating mechanism, which is somewhat orthogonal to the authentication. The idea is that the DAS, at one point of the protocol, also communicate their knowledge about the inter-domain channel and about the other domain to their clients. This takes the form of a *Trust Token* that the DAS sends to its client.

We explain this by example. Let us assume an authentication process between B in domain D_B and A in D_A. Let S_B be the DAS for B and S_A the DAS for A. The information that S_B can pass to B can be described with the following two categories. First, whether there exists a secure channel with S_A. Second, collected 'knowledge' about S_A, D_A and A itself.

The first is a simple yes/no statement. Where the DAS of two domains have exchanged their keys in a secure manner, the *Trust Token* will signal to the client that there is a secure channel. The protocol flow ensures that secure authentication is possible in this case. Where the *Trust Token* signals no such secure channel (and the DAS have to exchange their keys on an insecure channel), the decision whether to proceed remains with the client principals[1]. For this case, we have information of the second category.

Knowledge about S_A is delivered in the form of a set of properties, for example information about the channel on which the key exchange between S_B and S_A has taken place, or recent information from observations, e.g. whether S_A has acted faithfully so far or whether there are negative reports from other client principals (feedback). The only requirement is that there is a domain-specific

[1] The DAS could theoretically also stop the authentication process, of course.

configuration that defines the set of properties which both B and S_B must be able to interpret. The software may evaluate the *Trust Tokens* automatically or delegate it to a human user (which makes sense in interactive settings). Information about D_A may take a similar form, e.g. whether members of D_A have been known to have conducted fraud etc. The evaluation is again a domain-specific process. If S_A has not been acting faithfully, this will usually mean that the description of D_A will change, too. Information about the responder peer, A, is given in the same style.

An example of a Trust Token may thus look like this:

SecureChannel	no
KeyExchange	SinglePathLookup
OtherDomain	SelfCertifyingID: no
OtherDomain	PriorContacts: yes(10)
OtherDomain	KnownFrauds: no(0)
OtherDomain	HumanFeedback: yes(3)/no(0)
OtherPeer	NoInformation

Upon receiving it, a client will know that the DAS have looked up each other via a standard look-up without further hardening and exchanged their keys during this process. The other DAS can also not prove its identity as it is not bound to a public/private key pair. However, there have been 10 prior contacts with that domain before, and no frauds reported yet by other clients. The DAS received feedback for the other domain from 3 clients, all positive and based on human interaction, e.g. from voice sessions like in Zfone [8] (see Section 7). There is no information about the other peer.

A final remark: Note that even in the case of a secure channel and successful authentication of some entity A, this does not imply that A will necessarily be honest and act faithfully – merely that its identity has been ascertained. Information of the second category can thus also be useful in the case where the Trust Token indicates a *secure* channel between S_A and S_B.

2.1 Strengthening the Insecure Channel

Although not the primary focus of this paper, it is worthwhile to observe that there are measures to address the situation where DAS have no secure channel. The methods we propose are the following.

First, in the Peer-to-Peer case, we propose to strengthen the Peer-to-Peer system against network-related attacks. The other domain is given as an identifier within the system (a name, a number, etc.). The purpose of the Peer-to-Peer system is to look-up the other domain and its DAS. An attacker could set up a man-in-the-middle attack. To strengthen the communication, the DAS may attempt to communicate over multiple paths in the hope that a man-in-the-middle cannot control all of the paths. The multiple path aspects can be divided into two mechanisms: a) storage of location information at multiple places, b) multiple independent paths towards a target. Furthermore, we need to ensure that the paths and the corresponding peers are diverse.

Second, where possible, we suggest self-certifying IDs for the DAS to avoid man-in-the-middle attacks and impersonation. A self-certifying identifier is e.g. $ID = hash(PublicKey)$. With the knowledge of the private key a peer can prove ownership of the ID. This is quite useful and does not need a central authority, but it does not solve all problems. We can bind an ID to an entity with a key, but we cannot bind a name to an ID. If we need to resolve a human-readable or application-specific name, we still have no guarantee to reach the correct domain.

Third, where possible, we suggest to use user feedback to the DAS to report errors in authentication and misbehavior (although this can also be a flaw if users purposefully inject false information). Only user and application are able to judge if the authentication decision of an insecure session was correct or not. Furthermore, this is also true for the behavior of other entities.

None of these methods can solve the underlying dilemma, but the DAS can include information about which methods were used in the *Trust Token* to aid the client to some degree. This does not provide us with a secure channel, but rather acts as means of a risk assessment.

2.2 PKI Aspects

From the perspective of key distribution, our scheme represents a PKI with the following properties. It is distributed over several domains, yet can operate without a single global authority. This makes it more flexible and eliminates the need for manual interaction, yet allows manual setup of secure channels between domains and leverages trust evaluation. We argue that our concept offers better scalability as only domains that need to communicate are involved in the authentication scheme.

3 Authentication Protocol with Trust-Rating

In the following, we give a formal description of the protocol for our authentication scheme. We begin with the notion of authentication and state the protocol goals against this background.

3.1 Defining Authentication

There are several definitions for authentication in academic literature. We use the relatively strong definition by Lowe [9], ‘Injective Agreement’, against which we have defined our protocol. The Model Checker that we used to verify and check our protocol with also operates on this definition. We cite it here in the form of Fresh Injective Agreement:

Definition 1. *We say that a protocol guarantees to an initiator A Agreement with a responder B on a set of data items ds if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as* *responder in his run, and the two agents agreed on the data values corresponding to all the variables in ds, and each such run of A corresponds to a unique run of B. Fresh Agreement means that if any initiator A completes a run of the protocol, apparently with B, using particular values for the nonces, then A can be sure that at some time in the past, B believed that he was acting as responder in a run of the protocol with A, using the same values for the nonces.*

3.2 Protocol Goals

We now state the goals that our protocol must achieve. There are several goals in addition to Agreement.

Goal 1: Authentication as Agreement We specify this as Fresh Injective Agreement. The principals which authenticate each other are A and B. Authentication must be mutual.

Goal 2: Key Establishment As an outcome of the protocol, the principals A and B establish a session key. We insist that the DAS must not be able to obtain or derive this key.

Goal 3: Freshness of the Session Key A and B must be able to verify that the session key is a new one, and not the result of an earlier protocol run.

Goal 4: Mutual Belief in Session Key In the style of Boyd [10], we call a session key K that is only known to A and B (Goal 2) and is fresh (Goal 3) a *good key.* Mutual Belief in K is achieved if and only if B believes K is a good key for use with A and A also believes K to be a good key for use with B.

Optional Goal: Perfect Forward Secrecy We include Perfect Forward Secrecy as an optional goal.

3.3 Notation

We use the same notation as in [10]. The identity of a principal is denoted by a capital letter. An arrow indicates the process of one principal sending a message to another. A symmetric key is denoted by the letter K with an index indicating between which principals the key is shared. We denote a public key as PK_X, where X indicates the principal to which the key belongs. Encryption with a symmetric key is written as $\{m\}_K$. Encryption with a public key is denoted by $E_X(m)$. A signature with a private key is denoted by $Sig_X(t)$: token t, signed with the private key of agent X. Nonces are denoted by a capital N with an index.

We assume that all messages are integrity-protected without explicitly adding this to our notation.

3.4 Assumptions

We require clients and DAS to execute certain actions when a client becomes a member of a domain. A client principal X and its DAS S_X must agree on a secret symmetric key K_{XS_X} when X joins. X and S_X also exchange their public keys. Both is assumed to happen securely. S_X uses the cryptographically secure hash function h to calculate $h(X.PK_X)$ and signs this with its private key. $Sig_{S_X}(h(X, PK_X))$ will be referred to as a *Public Key Token*.

Two client principals B and A can later exchange their public keys through the following message exchange:

$$B \to A: \ (B, A, PK_B, Sig_{S_B}(h(B, PK_B))) \qquad \text{(Key Exchange Query)}$$
$$A \to B: \ (A, B, PK_A, Sig_{S_A}(h(A, PK_A))) \qquad \text{(Key Exchange Response)}$$

Note that this does not achieve key authentication. If DAS exchange their keys, this corresponds to exchanging self-certified keys.

3.5 Protocol Specification

We describe our protocol and motivate each message field.

In general, we take care to follow the guidelines by Abadi and Needham [11] wherever possible. Note that we encrypt every message, which adds to security.

Step 1: Requesting a Credential We let B from domain D_B initiate the authentication run. The first step is to acquire a *Credential* from B's Domain Authentication Server, S_B, to use for authentication with A in domain D_A.

$$B \to S_B: \ \{B, S_B, A, PK_A, Sig_{S_A}(h(A, PK_A)), N_B\}_{K_{BS_B}} \qquad \text{(Message 1)}$$

It is our protocol convention to indicate sender and receiver in the first two message fields. The message then states the responder (A), her public key (PK_A) and the *Public Key Token* from a previous key exchange. B refers to this authentication run by the nonce N_B, used for freshness and to avoid that principals mistake a message from one run for a message from a different run.

Step 2: Granting the Credential If S_A's public key is known, S_B can verify the *Public Key Token* and thus ensure that B uses the correct public key to communicate with A. In addition, as described in Section 2, S_B's answer depends on its knowledge about the channel to S_A, S_A itself, the domain D_A and A itself. S_B creates a *Trust Token* containing at least the following information:

1. Whether S_B is in possession of the correct public key for S_A (secure channel). If the key is not known: through which operation S_B can acquire S_A's public key.

2. Information about S_A, D_A and A that is supposed to help B in estimating whether S_A and A will be 'honest' and act faithfully. These are not numerical values but string descriptions that S_B and B can interpret. Such knowledge may also be derived from previous encounters.

S_B creates a *Credential* for B to use with A. It sends the *Credential* together with the *Trust Token*:

$$S_B \to B: \ \{S_B, B, Sig_{S_B}(h(B, A, PK_B, N_{S_B})), N_B, N_{S_B}, trust_{D_A}\}_{K_{BS_B}} \qquad \text{(Message 2)}$$

The *Credential* in the third field binds and signs B's identity, A's identity, B's public key and a nonce that S_B has freshly generated for this run, N_{S_B}. This is crucial to ensure that principals cannot mistake a message from one run for a message from another. Also note that the *Trust Token* is not part of the *Credential*. From this point on, B and S_B can identify this conversation by the tuple (N_B, N_{S_B}).

Step 3: Forwarding the Credential to A If B is satisfied with the information he obtains from the *Trust Token*, he may initiate the conversation with A. B sends:

$$B \to A: \ E_A(B, A, S_B, Sig_{S_B}(h(B, A, PK_B, N_{S_B})), N_B, N_{S_B}) \qquad \text{(Message 3)}$$

Note that this is a mere forwarding action. B indicates its responsible DAS; the nonces serve to counter replays.

Step 4: Forwarding the Credential to S_A A cannot verify the *Credential* (because she does not have S_B's public key), thus she forwards it with necessary additional information to S_A.

$$A \to S_A: \ \{A, S_A, B, S_B, PK_B, Sig_{S_B}(h(B, A, PK_B, N_{S_B})), N_{S_B}, N_A\}_{K_{AS_A}} \qquad \text{(Message 4)}$$

B, S_B, PK_B and N_{S_B} are forwarded in order for S_A to be able to verify the *Credential*. From this point on, A refers to her conversation with S_A by nonce N_A.

Step 5: Verifying Freshness If S_A is in possession of S_B's public key, it can immediately verify the *Credential*. Else it must acquire the public key by other, possibly insecure, means. Either will be indicated in the *Trust Token* for A later.

With the *Credential* verified, two issues remain. The first is the freshness of the *Credential*. This can only be verified with another message exchange. The second issue is that A needs a token that enables her to authenticate to B. Thus, S_A sends

$S_A \rightarrow S_B: \ E_{S_B}(S_A, S_B, B, A, N_{S_B}, N_{S_A})$ (Message 5)

S_A indicates the initiator B (field 3) and the responder (field 4). It uses information from the *Credential* to verify that S_B has indeed participated in this authentication run. Note that the triplet (B, A, N_{S_B}) is enough for this purpose. S_A also adds a nonce N_{S_A} by which it will refer to this conversation from now on.

Step 6: Freshness and Credential for A S_B can identify the authentication run through the triplet (B, A, N_{S_B}). In order to answer the freshness query, it responds with nonce N_{S_A} plus an *Authentication Token* for A to use with B: $Sig_{S_B}(h(A, N_{S_B}))$. This binds A's identity to nonce N_{S_B}, which is known to B.

$S_B \rightarrow S_A: \ E_{S_A}(S_B, S_A, Sig_{S_B}(h(A, N_{S_B})), N_{S_A})$ (Message 6)

Step 7: Authentication Decision S_A can verify the *Authentication Token*, and it has now verified that the nonce N_{S_B} refers to a fresh authentication run. S_A can now create a *Trust Token* for A, just as S_B did for B in the other domain.

$S_A \rightarrow A: \ \{S_A, A, Sig_{S_B}(h(A, N_{S_B})), N_A, trust_{D_B}\}_{K_{AS_A}}$ (Message 7)

Step 8: Authentication Response A can evaluate the *Trust Token* to decide whether to continue.

If A decides to proceed, she generates a new session key K_{BA} and sends:

$A \rightarrow B: \ E_B(A, B, Sig_{S_B}(h(A, N_{S_B})), N_B, K_{BA})$ (Message 8)

Note that A can safely use B's public key which it now knows to be the correct one. This establishes a secure channel between A and B.

B is in possession of S_B's public key and can thus verify A's *Authentication Token.* This completes the authentication.

3.6 Perfect Forward Secrecy

It is straight-forward to enable Perfect Forward Secrecy with a Diffie-Hellman Key Exchange. We replace K_{AB} with Diffie-Hellman values:

$A \rightarrow B: \ E_B(A, B, Sig_{S_B}(h(A, N_{S_B})), N_B, g_A, p, DH_A)$ (Message 8)

$B \rightarrow A: \ E_A(B, A, N_B, DH_B)$ (Message 9)

4 Verification of Protocol Goals

In the following, we verify that each of the protocol goals is achieved by our protocol.

Goal 1: Trust-rated Fresh Injective Agreement There are secure channels between the DAS and their clients because we may assume that the respective shared keys are not compromised.

We must distinguish between the communication between S_A and S_B and the communication between A and B. The communication between A and B is clearly a Dolev-Yao channel and it is not secure until both principals have verified the authenticity of the public keys. The channel between S_A and S_B, however, is a secure channel if the DAS have been able to verify the authenticity of each other's public keys. Otherwise it is also a normal Dolev-Yao channel, and insecure. We can therefore simplify the discussion: either the channel is secure or it is not. There is no need to discuss the latter case as there is no way to achieve secure authentication. Thus it is sufficient to examine the case of a secure channel between S_A and S_B.

We examine the case of B wishing to authenticate to A. In Step 2 of the protocol, S_B generates a *Credential* for B by creating and signing $h(B, A, PK_B, N_{S_B})$. The hash function binds these four values together. The signature Sig_{S_B} can only be created by S_B (because only S_B is in possession of the necessary private key). When S_A receives this token in Step 4 and verifies the signature, it can be sure of the fact that B wishes to authenticate to A, that B is a member of D_B, that B has public key PK_B and that S_B refers to this authentication run by the nonce N_{S_B}. Note that all values that S_A needs to calculate and check the hash value are included in the message. The only thing that S_A still has to check is the freshness of the nonce N_{S_B}.

This happens in Steps 5 and 6. S_A queries S_B by referring to the tuple (B, A, N_{SB}). S_A shows that it knows the value of N_{S_B} and is referring to the correct authentication run. It also sends the identity of A, thus assuring S_B that A is a member of D_A. In Step 6, S_B answers S_A's request by replying with the *Authentication Token* $Sig_{S_B}(h(A, N_{S_B}))$. This token binds the identity of A to N_{SB}, and S_A knows that N_{SB} is fresh. After Step 6, S_A can be sure of B's identity and authenticity. It forwards this authentication information to A in Step 7, and A can accept (forward the *Authentication Token*) or refuse (stop there).

We examine the authentication of A to B. S_B creates the *Authentication Token* $Sig_{S_B}(h(A, N_{S_B}))$ for A to authenticate to B. When A receives the *Authentication Token* in Step 7 and forwards it to B in Step 8, B can be assured of A's identity. B can verify S_B's signature and knows that S_B must have created the *Authentication Token* – and thus that S_B believes the information about A to be correct.

We compare this with the definition of Agreement. The set ds of values that the principals need to agree on are exactly the *Credential* and the *Authentication Token*: $Sig_{S_B}(h(B, A, PK_B, N_{S_B}))$ and $Sig_{S_B}(h(A, N_{S_B}))$. The principals obviously agree on these values if and only if the protocol has been completed successfully. The condition of injectivity holds as well. When B completes his run as initiator, apparently with A, then A was acting as responder in her run, apparently with B. All principals refer to each authentication run with a nonce of their own, plus their peering principal's nonces in their replies. Thus it cannot happen that a principal mistakes a message to be from a different run, or vice versa. Thus each run of B corresponds to a unique run of A. This is Agreement. It is *trust-rated* due to the *Trust Tokens*.

For the *Freshness* property, we observe that all principals create and maintain their own nonces by which they refer to a combination of channel and run. Thus all entities can be sure that the messages they receive are fresh ones and contain fresh values.

Goal 2: Key Establishment In message 8, A creates and sends a session key K_{AB} to B.

Goal 3: Freshness of the Session Key In message 8, A replies with nonce N_B and a new session key.

Goal 4: Mutual Belief A generates K_{AB} for the purpose of communicating with B. A knows that it is a good key. It is obviously fresh, and it can only be known to A and B because A encrypts it with PK_B, which A knows to be B's public key (thanks to S_A checking S_B's signature). If the authentication run was successful, B can also be sure that the session key is valid and fresh. Since A has sent the session key, B can deduce that the session key must be valid for this communication. Thus it must be a good key. In other words, B believes K_{AB} to be a good key for communication with A, and A believes K_{AB} to be a good key for communication with B. This is Mutual Belief.

Goal 6: Perfect Forward Secrecy Perfect Forward Secrecy can be achieved with the Diffie-Hellman Key Exchange described above. The channel between A and B is secure because A's and B's public keys have been verified by their respective DAS.

5 Discussion of Security

Our security model is the Dolev-Yao model ([2]), generally considered to represent the strongest possible attacker (only limited by the cryptographic primitives). We evaluated the security of our protocol with the AVISPA Model Checker [12] (OFMC backend).

We will now describe how the specification was modeled in AVISPA, which also uses the Dolev-Yao model. Information can be passed in two ways in AVISPA. The first is as a constant (environment parameter). This value cannot be interfered with, in contrast to the second method, which is to pass information to a principal during a run (a 'variable').

The (symmetric and asymmetric) keys between the clients and their DAS plus the *Public Key Tokens* are thus passed as environment parameters. To model the secure channel between S_A and S_B, the respective public keys are also passed as environment parameters. A and B learn each other's *Public Key*

Tokens during a message exchange, thus such a principal's *Public Key Token* is always a variable. The same applies to *Trust Tokens.*

The intruder I may try to impersonate any principal but he does not know their secret or private keys. I is allowed to be a member of both D_A and D_B, i.e. I has established K_{IS_A} and K_{IS_B} with the DAS and has received *Public Key Tokens.* The intruder is allowed to participate in protocol runs (under his own identity) or be a non-participant, or both, at the same time.

The protocol goals are modelled as follows:

1. Authentication as Injective Agreement can be modelled directly in AVISPA. This is applied to *Credential*, *Authentication Token*, *Trust Tokens* and K_{AB}.
2. We model Freshness by having each principal create a fresh nonce for each conversation with another principal and checking the value of this nonce when the reply arrives. We require Agreement on the value of the nonce between the principals using it.
3. We define the goal 'secrecy' for K_{AS_A}, K_{BS_B}, K_{AB}, the private keys and the *Trust Tokens* – i.e. these values must remain unknown to the intruder.

We evaluated all possible combinations of A, B and I interacting. However, due to the 'explosion of the state space', our evaluation was limited to three concurrent sessions. The evaluation showed our protocol to be secure for these scenarios. We also found during our verification that runs with three parallel sessions did not find any new attacks compared to just two parallel sessions.

A formal proof for arbitrarily-sized systems is difficult to provide and remains future work (just as for many other cryptographic protocols). One possibility to extend the model checking results could be Lowe's 'Completeness Checking' [13], which provides a proof that under certain constraints a protocol, which is secure for a small system, is also secure for arbitrarily-sized systems. However, this theory only covers secrecy and not Agreement. A different approach would be to employ other model checkers like Scyther [14], which can also deal with unbounded scenarios. However, Scyther in its current form is geared towards checking authentication as Non-Injective Synchronization and does not cover Injective Agreement. The definition of Synchronization is subtly different from Agreement.

6 Further Considerations

Our scheme exhibits a few properties that are worthy of some consideration.

For example, we expect our scheme to scale reasonably well: it effectively 'decouples' domain activities. The DAS only need to know the keys of their client principals and those DAS that they have already communicated with. There is no need for *a priori* key verification, and there are no certification chains to follow. It is thus easy to introduce new domains. Where this results in an insecure channel between domains, our scheme remains flexible: clients can make an informed decision in such situations.

Key revocation can be easily added, without the need to distribute Key Revocation Lists. When a client revokes a key after *Public Key Tokens* have been exchanged, the DAS will refuse to create *Authentication Tokens*. Keys can also be updated easily: peers send a new key to their DAS (via a domain-internal protocol) and the DAS responds with a new *Public Key Token*.

One can also observe that the scheme provides a rudimentary defense against Denial-of-Service attacks. The DAS in the responder's domain D_A, for example, only becomes active if it receives a valid message from A. There is no way that other principals can trigger a computationally expensive answer, especially if they're from outside D_A. The protection for the initiator's S_B, meanwhile, lies in the fact that it can drop messages from outside its domain that do not include one of the nonces that S_B has recently created for an authentication run.

Our scheme could be extended to enable DAS load distribution within a domain: several DAS could share their (symmetric and asymmetric) keys. An anycast mechanism, possibly with proximity property, would enable clients to address their DAS.

7 Related Work

X.509 and the PGP/GPG Web of Trust are probably the best-known PKIs. X.509 uses CAs, and some argue that this concept offers the highest degree of security that is possible today. Proponents of Webs of Trust counter that a CA is not *per se* trustworthy and that the user can determine better whom to trust. A problem of PKIs is key revocation. This is usually handled with Key Servers and Key Revocation Lists, requiring great diligence. We do not intend to take sides here but rather point out that our concept establishes itself between hierarchical PKIs with CAs and 'flat' Webs of Trust.

The Kerberos protocol [15] can be used for authentication between domains. The foreign Ticket Granting Server must be registered with the local Key Authentication Server. The concept is transitive but requires manual configuration, resulting in higher maintenance.

Protocols for Identity Federation (IF) (e.g. [16]) extend this with the notion of 'Circles of Trust'. They enable the portability of an identity between domains. Keys must be exchanged *a priori* and out-of-band. Many IF protocols additionally use TLS in the underlay, thus establishing secure channels over which the actual authentication protocols are defined. Our scheme is different in both regards.

Other approaches try to eliminate the CA by distributing its functionality, e.g. by using Multiparty Computation (MPC) as presented by Narasimha and Saxena et al. in [17], [18]. They describe a decentral method for group membership control. The scheme requires all principals to execute the protocol faithfully and does not aim to counter the Sybil Attack. If intruders gain access to the group, the scheme is compromised. Defenses can be established with Verifiable Secret Sharing Schemes, e.g. as in [19], but this requires either a high number of broadcast operations or needs to introduce a CA.

If there is no secure channel yet, authentication cannot be secure. The Resurrecting Duckling [20] model assumes that the first contact is not compromised by an intruder and establishes a context for future contacts. The Voice-over-IP system Zfone [8] extends this approach by taking measures against a man-in-the-middle attack on first contact: it requires the (human) users to initiate their conversation by reading a 'Short Authentication String' (SAS) to each other that is derived from the exchanged Diffie-Hellman values.

Concerning trust, Maurer presented a model for reasoning about trustworthiness in a PKI context in [21]. Similar methods could be developed for a client's evaluation of a *Trust Token*.

8 Conclusion

We have presented a four-party authentication scheme and a new protocol for domain-based scenarios. The protocol is an attempt to design an authentication process that is largely decentralized yet remains efficient. We have evaluated its security with model checking. A rigorous proof remains future work.

Since the applications we have in mind include use cases where no prior knowledge exists, we have introduced the concept of trust-rated authentication. Trust-rated authentication uses a *Trust Token* to signal important information (pre-existing secure channel or not, trust in other server, trust in its peers, . . .) from the authentication server to its client. The client can then decide depending on its current security requirements.

Our overall scheme positions itself somewhere between hierarchical PKIs and Webs of Trust: principals trust distinguished principals in their domain.

References

1. Boyd, C.: Security architecture using formal methods. IEEE Journal on Selected Topics in Communications **11** (1993) 694–701
2. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(2) (1983) 198–208
3. Skype Ltd.: Skype (homepage). http://www.skype.com (February 2008)
4. Ellison, C., Schneier, B.: Ten risks of PKI: What you're not being told about public key infrastructure. Computer Security Journal **16**(1) (2000) 1–7
5. Douceur, J.R.: The Sybil Attack. In: Peer-to-Peer Systems: 1st International Workshop, Cambridge, MA, USA (IPTPS 2002). Revised Papers. (2002) 251–260
6. Zimmermann, P.R.: The official PGP user's guide. MIT Press Cambridge, MA, USA (1995)
7. Jøsang, A.: An algebra for assessing trust in certification chains. In: Proceedings of the Network and Distributed Systems Security Symposium (NDSS 1999), Internet Society (1999)
8. The Zfone Project: Zfone (homepage). http://zfoneproject.com (2007)
9. Lowe, G.: A hierarchy of authentication specifications. In: Proceedings of the 10th IEEE Computer Security Foundations Workshop, Rockport, MA, USA (CSFW '97). (1997)

10. Boyd, C., Mathuria, A.: Protocols for authentication and key establishment. Information Security and Cryptography. Springer (2003)
11. Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. IEEE Transactions on Software Engineering **22**(1) (1996) 6–15
12. The AVISPA Project: Automated Validation of Internet Security Protocols and Applications (homepage). http://www.avispa-project.org/ (January 2008)
13. Lowe, G.: Towards a completeness result for model checking of security protocols. Journal of Computer Security **7**(2) (1999) 89–146
14. Cremers, C.: Scyther - Semantics and Verification of Security Protocols. Ph.D. dissertation, Eindhoven University of Technology (2006)
15. Neuman, B.C., Ts'o, T.: Kerberos: an authentication service for computer networks. IEEE Communications Magazine **32**(9) (1994) 33–38
16. Goodner, M., Nadalin, A.: Web Services Federation Language (WS-Federation). OASIS Specification (work-in-progress). http://www.oasis-open.org (January 2008)
17. Narasimha, M., Tsudik, G., Yi, J.H.: On the utility of distributed cryptography in P2P and MANETs: the case of membership control. In: Proceedings of the 11th IEEE International Conference on Network Protocols 2003. (2003) 336–345
18. Saxena, N., Tsudik, G., Yi, J.H.: Admission control in Peer-to-Peer: design and performance evaluation. In: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks. (2003) 104–113
19. Pedersen, T.: A threshold cryptosystem without a trusted party. In: Advances in Cryptology - EUROCRYPT '91. Volume 547. (1991) 522–526
20. Stajano, F., Anderson, R.: The Resurrecting Duckling: security issues for ad-hoc wireless networks. In: Proceedings of the 7th International Workshop on Security Protocols, Cambridge, UK. (1999)
21. Maurer, U.: Modelling a Public-Key Infrastructure. In: Proceedings of the 1996 European Symposium on Research in Computer Security (ESORICS '96), Rome, Italy. (1996)