Secure Stateless Trust Negotiation

Andreas Klenk and Georg Carle
Technische Universität München
Network Architectures and Services
Boltzmannstraße 3, 85748 München, Germany
{klenk, carle}@net.in.tum.de

Benoit Radier and Mikael Salaun
Orange - France Télécom R&D
Orange Labs
avenue Pierre Marzin 2, 22307 Lannion, France
{benoit.radier, mikael.salaun}@orange-ft.com

***Abstract*—Trust establishment is a challenge for services in distributed open environments. Trust Negotiation is a requirements-driven method for establishing trust between strangers and parties with limited mutual trust. Protocols for *stateless* trust negotiation use messages which contain the whole negotiation state. Stateless trust negotiation systems are advantageous in open environments due to their ability to recover from failures by simply processing the last message again. Furthermore stateless negotiation reduces resource consumption at the negotiating parties for long lasting negotiations. A drawback of stateless negotiation systems is that the complete negotiation state is prone to forgery by the other party in the negotiation. Stateless negotiation can not be used if it does not address this vulnerability. We propose a security extension to an existing XML based trust negotiation protocol in order to allow for secure stateless negotiations. Our novel alternating signature protocol detects manipulations of the negotiation state and introduces non-repudiation to agreements. Stateless trust negotiation extended with the alternating signature protocol can be a viable alternative to stateful negotiation approaches especially for long lasting negotiations in unreliable environments.**[1]

I. Introduction

Research in the area of next generation service platforms is driven by a need to cope with increasingly dynamic service relationships. It becomes difficult to derive trust from static knowledge about identities and services. A growing body of research targets *Trust Negotiation (TN)* as a method for establishing trust on the fly (see [1], [2], [3]). TN systems use a policy-driven iterative negotiation process to reach an agreement between two parties that need not have a prior trust relationship. Trust establishment works through an iterative exchange of digital credentials. Credentials can act as authorization tokens to obtain access to services or they can contain verifiable information linked to a negotiating party. Trusted third parties can certify the correctness of the information in the credentials. Credentials for frequent flyer programs, credit card information, or affiliations to organizations are just a few examples. The main focus of TN research is on the protection of credentials with sensitive information [4]. Access control policies define under which conditions credentials can be released. The TN protocols ensure that only a minimal set of credentials is exchanged and that all preconditions in the access control policy are met before releasing a particular credential.

Even though several stateful trust negotiation frameworks emerged, their operation in unreliable environments is still a research issue [5], [6]. Stateless trust negotiation systems [7] are beneficial under these circumstances because they allow for easy recovery in the face of failures. A negotiator which receives a corrupted message, or which does not receive a reply at all, can simply resend the last message. If the individual negotiator shares the load between multiple servers internally, it profits from easy fail-over and load-balancing on a per-message basis. Another issue is that trust negotiations can take a long time if humans are involved in the negotiation process for making decisions or for issuing credentials. Long lasting negotiations are costly using stateful negotiation because the involved parties must preserve the negotiation state until the negotiation terminates. Stateless negotiation protocols avoid this state at the currently inactive negotiation systems because all necessary information for the processing of a negotiation message is encapsulated within the message itself.The stateless nature of the negotiators becomes a disadvantage if one party tries to forge messages. It could remove preconditions, modify dependencies, and replace credentials to obtain an agreement that favors the adversary. End-to-end security protocols (e.g. TLS, IPSEC) protect only against manipulations by malicious third parties. Only with an effective protection against forgery, stateless negotiation protocols become feasible. The challenge is to distinguish legitimate modifications of the negotiation state that are in accordance with the negotiation protocol, from forgery of all other portions of the negotiation state. The preconditions, dependencies and credentials that have been exchanged in previous rounds must not be modified.

This paper presents the following contributions: We propose a novel alternating signature protocol for the protection of XML negotiation messages to detect forgery of negotiation states. We apply XML Signature [8] to sign portions of the messages for verifying the integrity of the negotiation. However, XML Signature alone is not sufficient, because it only protects the *old* parts of the negotiation state against changes. It cannot determine if the *new* parts of the negotiation state, that have been appended by the other party, comply with the specification of the negotiation protocol and are in accordance with the policies that steer the negotiation. Our algorithm reverses the last negotiation step of the other party by first removing all legitimate changes that are in accordance with

the negotiation protocol and by verifying that the remaining state has not been modified. We introduce a generic model of XML negotiation states that allows us to formalize the protocol messages during stateless trust negotiation. By using this model we demonstrate how the alternating signature protocol extracts the parts of the negotiation state that should have been immutable at the current iteration of the negotiation. Different simulated attacks will show how our implementation of the alternating signature protocol detects manipulations. We present experimental results from measurements with the prototype to argue how a careful choice of cryptographic algorithms improves the negotiation performance.

The remainder of this paper is structured as follows. Sec. II gives a brief overview of our negotiation process. Sec. III describes the formal model of the negotiation and Sec. IV introduces the alternating signature protocol. Different simulated attacks are presented in Sec. V and performance measurements are given in Sec. VI. Sec. VII summarizes related work and we conclude in Sec. VIII.

II. Trust Negotiation in Four Phases

The VersaTrust negotiation framework we presented in [7] has been designed for stateless trust negotiation. The protocol has four phases: 1.) The negotiation is initiated by a *Resource Request* 2.) The objective of the *Negotiation Phase* is to find the safe disclosure sequence of credentials. A credential can only be released after the requirements stated in its access control policy have been satisfied. Credential requirements specify type and properties of credentials that the other party must provide. An iterative exchange of credential requirements leads to a tree of interdependent requirements. If one path from the root to a leaf has credentials that can be released unconditionally and all dependencies on this path can be satisfied, a potential agreement option has been discovered. 3.) After one agreement option has been selected, the *Credential Exchange Phase* starts where the parties commence a careful exchange of the promised credentials. They always verify that all preconditions of a particular credential are satisfied before they release this credential. 4.) The negotiation either succeeds after all credentials have been successfully exchanged or it terminates with a failure if some requirements are not fulfilled.

The Negotiation Phase is dedicated to the requirements exchange and the identification of possible solutions and must complete before the Credential Exchange Phase starts. This separation has the advantage that the negotiation can fail during the Negotiation Phase without revealing sensitive information or critical access-granting tokens. A credential is only released if all preconditions of this credential are met. Consider, for example, a negotiator that only releases an access token for its web service after it received a permission to withdraw a certain amount of money from the requester.

III. Model of XML Negotiation States

The Negotiation algorithm works through an exchange of XML documents that contain the complete negotiation state. The endpoints do not need to preserve state across multiple iterations of the negotiation, instead they can reestablish all information about a negotiation by parsing the current negotiation message. These documents grow at each iteration because the processing party appends its requirements, dependencies, and commitments to the document. All elements and data remain in the document until the negotiation terminates.

There are various ways for describing an XML document formally. We present our data model and operations in the context of XML negotiation states. An XML document is formed by a hierarchy of XML tags and can be represented by a directed graph $G_{negState}$. Adjacency matrices can be used to represent finite directed graphs. However, common

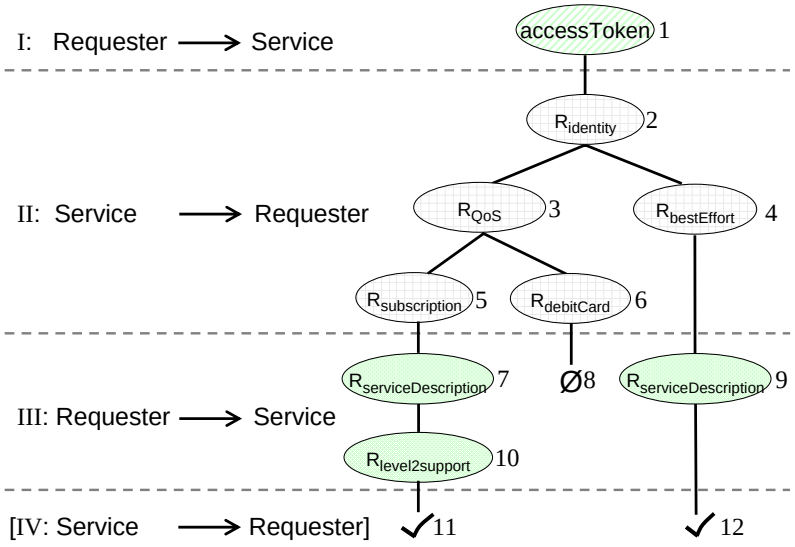

Fig. 1. Negotiation Tree after four Iterations during Negotiation Phase

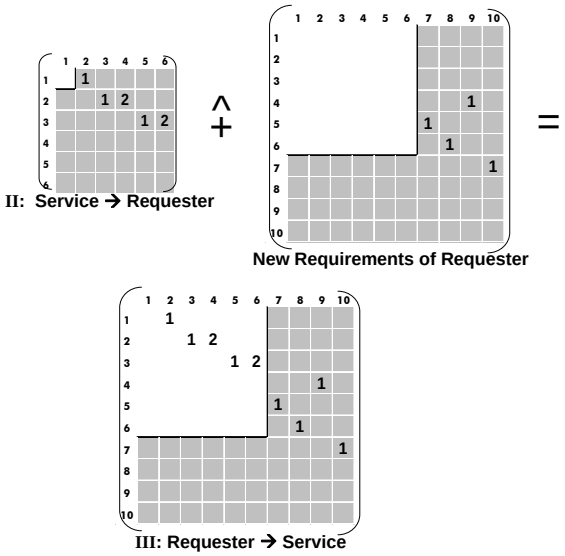

Fig. 2. Matrix Representation of Negotiation Tree

definition of adjacency matrices do not allow to preserve the order of the children of a node, which is important for XML InfoSets and XML canonicalization. We use a slightly different definition and map the directed graph $G_{negState}$ that has n vertices to a matrix $M_d^{n\times n}$ where an entry $a_{i,j} > 0, a_{i,j} \in \mathbb{Q}^+, ij \in \mathbb{N}, i \leq n, j \leq n$ denotes the presence of one edge from the vertex v_i to v_j. The child $a_{i,x}$ of v_i precedes child $a_{i,y}$ if $a_{i,x} < a_{i,y}$. The node labeling function $\sigma_d : V \mapsto \sum$ provides values from the alphabet $\sum$ for entity names, and $\gamma_d : V \mapsto \sum$ provides a set of attributes for a vertex and if present the XML data value. The whole XML Negotiation state is represented by the tuple $d =< r_d, M_d, \sigma_d, \gamma_d >$, where r_d is the root element.

A. Negotiation Phase

The Negotiation Phase is where each party states its requirements. The negotiation starts when a Requester *R* sends a resource request to a Service *S*. The service has various requirements regarding its credentials that would allow *R* to access the service. For example, it could require either a permanent subscription or the willingness to pay for the service. By processing these requirements the requester learns about the terms under which this service is available. The requester has probably its own requirements concerning the service usage, for instance, quality of service and privacy. The requester now appends its requirements to the negotiation state. The result is a tree structure (see Figure 1) of interdependent requirements. Father-child relationships denote conjunctive conditions where all requirements must be fulfilled. A sibling relationship denotes a disjunctive condition where only one condition must be satisfied.
The negotiation phase can be modeled as a sequence of matrix additions. We define a matrix addition between matrices with unequal dimensions $A^{n\times m}, B^{x\times y}, x \leq n, y \leq m$ in our context as follows:

$$(A^{n\times m}\hat{+}B^{x\times y})_{i,j} = \begin{cases} a_{i,j} + b_{i,j} & \text{for } \begin{matrix} a_{i,j} \in A, b_{i,j} \in B, \\ i \leq x \wedge j \leq y \end{matrix} \\ a_{i,j} & \text{for } \begin{matrix} a_{i,j} \in A, x < i \leq m \\ \vee y < j \leq n \end{matrix} \end{cases} \quad (1)$$

The requester at iteration j appends its requirements M_j^{req} and creates the negotiation state $M_{j+1} = M_j \hat{+} M_j^{req}$. This algorithm repeats for all $j \in \{1...n-1\}$, where n is the number of iterations during the negotiation phase. Figure 2 shows the matrix representation of the message M_{II} from Figure 1 step II and how the message M_{III} is constructed.
If one party is willing to release all requested credentials on a path without stating further requirements, it appends a *possible agreement* node. This path is a possible solution. There is another special XML element to indicate if a party is not willing to release a specific credentials. The path from the root to this leaf is marked with a *failed* node then.
The negotiation phase ends either when one possible solution has been found, or after all paths have been either marked as *failed* or *possible agreement*. The selection of an optimal path can be arbitrarily complex. As we are mainly concerned about the negotiation protocol, we refer to the work on agents, preference based selection or rely on human judgment to make an intelligent choice.

B. Credential Exchange Phase

The purpose of the *Negotiation Phase* was to discover one viable solution for both parties. However, it is still possible that one party does not deliver the promised credentials. The *Credential Exchange Phase* ensures that one credential is released only after all preconditions stated in its access control policy have been fulfilled. The order of the credential requirements in the path from leaf to the root in the negotiation tree has the property that it is a safe disclosure sequence. The credentials M_j^{cred} are only released after all requirements are satisfied. The protocol marks the chosen path and will iteratively exchange the credentials by appending them to the negotiation state $M_{j+1} = M_j \hat{+} M_j^{cred}$. The negotiation was successful after all requirements have been satisfied by corresponding credentials.

IV. Secure Stateless Trust Negotiation

We first describe the attacker model before we propose a security extension of the protocol for stateless trust negotiation to counter these threats.

A. Attacker Model for Stateless Negotiation

There are two types of adversaries for stateless negotiation:
1) A *malicious third party* that is located on the message path between two negotiating parties and is able to intercept, manipulate and replay negotiation messages.
2) The *other party is a malicious adversary* that tries to cheat during the negotiation. End-to-end security is no protection if the other party in the negotiation wants to take unfair advantage by forging the negotiation state in parts or as a whole. Each peer wants to protect the negotiation state from malicious modifications because this document states promises about the future behavior of the peers and requirements that the peers must fulfill. If an adversary succeeds in manipulating the negotiation state, it can generate agreements that favor itself and are very unfavorable for the victim.

B. Alternating Signatures and Message State Reduction

We devised a novel algorithm with alternating signatures and message state reduction to verify that previous messages have not been manipulated. The purpose of the algorithm is to ensure that the other party did only append to the negotiation state and did not change any preexisting parts of the message. Before a message M_i is sent, the current peer will apply an XML Signature [8] covering the whole negotiation state. This signature includes the history of all requirements and tokens exchanged up to that point. The peer will append the signature to the message and will send it to the other party.

The other party will process the message and send a reply M_{j+1} with an extended negotiation state. When the first peer receives the answer to its original message, it must first verify the integrity of message M_{j+1}, before it can start processing. The signatures of the two messages do obviously not match $sign(M_{j+1}) \neq sign(M_j)$ because the message M_j has been extended with additional state, by appending requirements or credentials.

Upon reception of the message, the receiving peer must remove all nodes from the XML document that the other party appended. In our model of XML negotiation states, the dimensions of the matrix have grown to accommodate the new nodes. In terms of our model, we must reduce the dimension of the received message $M_{j+1}^{b\times b}$ to obtain the representation of the original message $M_j'^{a\times a}$with $b > a$. We define the matrix $I^{m\times n}$ as follows:

$$(I^{m\times n})_{i,j} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} \text{ with } i \in 1,\ldots,m, j \in 1,\ldots,n \tag{2}$$

The matrix multiplication $M^{l\times m} \times I^{m\times n} = R^{l\times n}$ preserves elements that are within the new dimensions of $R^{l\times n}$. By using two matrix multiplications, we obtain the message $M_j'^{a\times a}$:

$$I^{a\times b} \cdot M_{j+1}^{b\times b} \cdot I^{b\times a} = M_j'^{a\times a} \tag{3}$$

The implementation of the protocol uses a *new nodes list* to indicate which new nodes have been appended to the document by the other party. One peer must first remove the new nodes before it can verify the signature. Be aware that the resulting document might still not fully match the original document because of different representations of equivalent documents due to syntactic freedoms of XML. One XML standard is very helpful in obtaining a valid signature after the document has been modified two times: XML Canonicalization from [9] ensures that XML documents, which can have varying representations, but are logically equivalent in a given application context, will be transformed in an unambiguous XML representation. The digital signature of the reduced document must match the previous signature of this peer. The digital signatures of both peers are contained in the messages. The signatures will only match if the resulting document after the reduction of M_{j+1} is equivalent with the document M_j. Non-matching signatures indicate that the previous negotiation state has been tampered with and the negotiation fails. If the new nodes lists is corrupted the algorithm will remove an incorrect set of nodes and the signatures will also not match. After successful signature verification the rest of the message is handled by the negotiation algorithm as described in [7]. It ensures that the new parts of the message comply with the semantics of the negotiation protocol and the access control policies.

The Figure 3 shows how both negotiating parties apply this algorithm. The Requester includes $sig_{MI} = sign(M_I, key_{Requester})$ in its message M_I. When it receives M_{II} it removes the changes that the Service applied and verifies that $sig_{MI} = sign(reduce(M_{II}, changedNodes), key_{Requester})$. The Service acts also according to this algorithm. The algorithm will be applied successively for each message until the negotiation ends. The complete history of requirements, promises, commitments and choices is thereby protected. If the signatures do not match for one message, the negotiation is terminated and the incident will be logged. A list of previously processed signatures is used to prevent an excessive replay of negotiation messages and to avoid Denial of Service attacks.

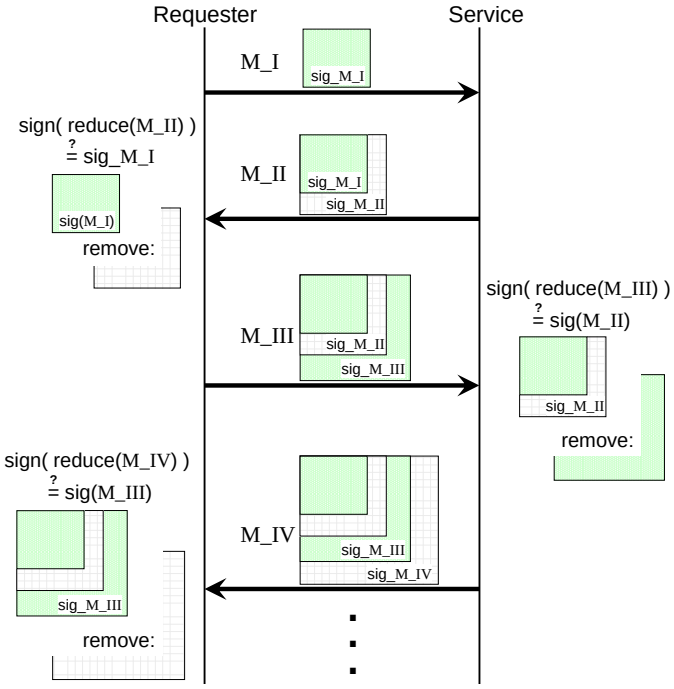

Fig. 3. Message Reduction and Verification with Alternating Signatures

C. Different Options for Cryptographic Signatures

Our framework uses XML Signature from [8] to protect the integrity of the complete negotiation state and includes two signatures in the document: one for the Requester and one for the Service. One option is to use RSA [10] public key cryptography. The use of RSA makes fail-over and load-balancing easier because other backup servers can verify the digital signature with the public key of the failed server. An XML firewall could integrate the logic of the alternating signature protocol to verify the signatures and drop manipulated messages. Another important property of using public key cryptography together with digital certificates is non-repudiation. Both parties sign the whole agreement successively by applying this algorithm. A third party can verify the integrity and the origin of the agreement by checking the signatures.

An alternative option is to use the keyed-hash message authentication code (HMAC) from [11]. HMAC uses a secret key and a cryptographic hash function to calculate the

message authentication code of a negotiation state. The secret keys in our protocol are exclusive for Service and Requester each. Only the peer that used a key for its signature must be able to verify the signature. If one negotiator wants to use load-balancing between multiple servers internally, the secret key must be shared with all other servers that can act as backup.

V. Experimental Verification of Security

We assume that a suitable cryptographic protocol for end-to-end security (e.g. TLS or IPSEC) is in place to ensure confidentiality and to protect against attacks by malicious third parties. Hence, our security analysis focuses on the other party in the negotiation that forges negotiation states. We augmented one party for each experiment with a software module to simulate different attacks on the negotiation depicted in Figure 1. In our experiments the negotiation state belonging to the Service was manipulated by the Requester and vice versa. The following list shows the attacks we simulated and their impact on the agreement:

1) *Attack* Adversary removes nodes
a) *Test* Remove $R_{subscription}$ and $R_{debitCard}$
Impact Agreement to access service without paying or subscription
b) *Test* Remove credential C_{QoS}
Impact Agreement without QoS guarantee
2) *Attack* Adversary modifies node content
Test Modify $R_{debitCard}$
Impact Require less money to access service
3) *Attack* Adversary moves nodes
Test Swap R_{QoS} and $R_{bestEffort}$
Impact Obtain service with QoS for free
4) *Attack* Adversary adds nodes
Test Sneak in additional unconditional agreement option for QoS $R_{freeQoS}$
Impact Obtain service with QoS for free
5) *Attack* Adversary manipulates new nodes list
a) *Test* Do not state all nodes that were added
Impact Exclude from processing by negotiation algorithm
b) *Test* Try to hide added $R_{freeQoS}$ from attack 4)
Impact Cover forgery of negotiation states

The negotiation state forgery of attacks 1),2),3) are detected by verifying the digital signatures of the document portions that should have been immutable. The attacks 4) and 5) cannot be detected by XML Security alone, but require the message state reduction that obeys the protocol specification. All nodes that follow the protocol rules will be removed from the document before the signature is verified. Any XML node that does not comply with the algorithm [7] will stay in the test document and will lead to non-matching signatures. Thwarting the attack 5b) also relies on the negotiation algorithm that enforces that nodes in the changed nodes list belong to the remote party and handles these nodes in accordance with the access control policies.

VI. Performance Evaluation

We conducted performance measurements to determine the impact of the different cryptographic algorithms. One algorithm we used was the keyed-hash message authentication

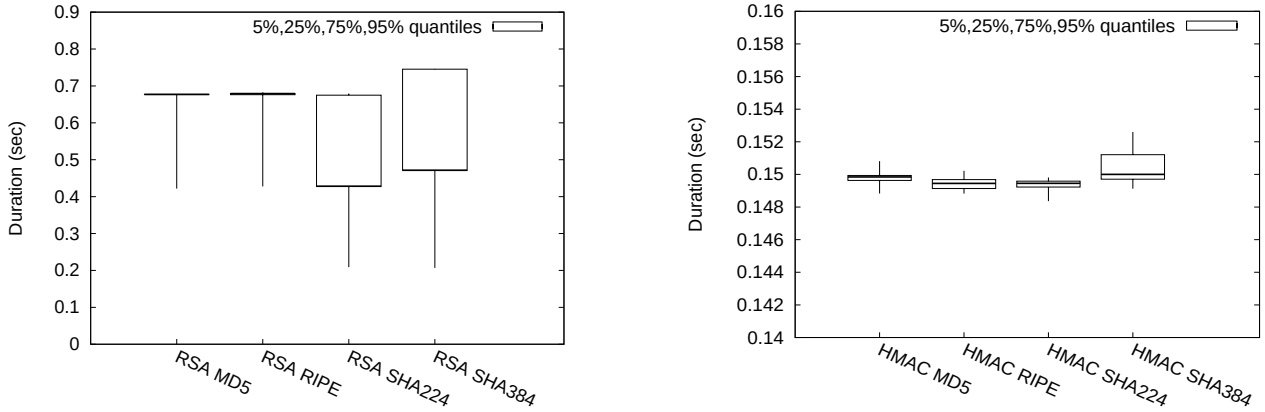

Fig. 4. Duration of Complete Negotiation with different Cryptographic Hash Functions a) using RSA (left) b) using HMAC (right)

code with a secret key (HMAC) from [11]. We also investigated the impact of public key cryptography with the RSA [10] algorithm. The measurements throughout the experiments were performed with two computers with Intel Pentium 4 2.80GHz CPU, 1GB of RAM, running under Ubuntu Linux with Kernel 2.6.24. The computers were interconnected via a 100Mbit/s local area network. We used libxml2 (http://www.xmlsoft.org) for parsing the XML document into a DOM representation and for its implementation of Canonical XML. Requester and Service apply XML Signature with the XMLSec Library (http://www.aleksey.com/xmlsec/) which relied on the cryptographic routines from openssl.

We used a more complex scenario than the one presented in this paper with an iterative exchange of twelve messages between Requester and Service. That means that each party receives and sends a negotiation message six times. Figure 4 a) shows the performance of a complete negotiation, protected by HMAC. The negotiations terminate for all chosen security algorithms after approximately 0.15 seconds with small variance. In contrast, the RSA algorithms lead to much longer negotiation times and higher variance with a median of about 0.45 seconds for RSA SHA224 (see Figure 4 b).

We performed another experiment to determine the impact of message sizes on cryptography and scalability. We summed the size of all messages that were exchanged during the negotiation and determined the effective negotiation rate. The symmetric nature of the protocol implies that the computational effort of Service and Requester is roughly equal. Figure 5 shows that the size of the negotiation messages determines the processing capacity.

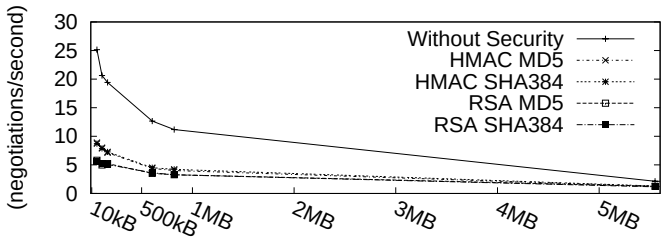

Fig. 5. Message Size and Security Algorithms determine Negotiation Rate

Future work includes the examination of a hybrid strategy that uses HMAC for the negotiation and public-key cryptography only for signing the final agreement. Incremental signatures as presented in [12] are also a promising direction for improving the negotiation performance, as large parts of the negotiation message are immutable.

VII. Related Work

Automated Trust Negotiation was introduced in [1] as a method for establishing trust between strangers through a careful credential exchange. The objective of trust negotiation is to disclose only the minimal set of credentials necessary for the successful termination of the negotiation. The paper in [4] presents an overview of fundamental concepts and related work on Trust Negotiation. To the best of our knowledge, our work in [7] is the only TN protocol that allows for stateless operation of the negotiating systems. Different from our stateless negotiation protocol, the stateful TN frameworks must protect their communication against malicious third parties only. The Traust [13] TN system relies on end-to-end security with SSL. The approach published in [14] uses WS-Security to secure its messages.

The *Trust-X* framework of [3] uses XML for its Trust Negotiation Language, disclosure policies and credentials. An extension of *Trust-X* presented in [5] introduces fault tolerance. The recovery strategy works with check points and persistent state in a local database. The authors of [6] also use check points to introduce fault tolerance at the protocol level and to recover from system failures. Both approaches rely on runtime state at the negotiating systems and require additional persistent state for the check points which must be preserved until the negotiation terminates. Stateless trust negotiation provides fault-tolerance and works well for long-lasting negotiations. The advantage is that it reduces local resource consumption and that it facilitates load-balancing.

VIII. Conclusion

Stateless trust negotiation is well suited for long-lasting negotiations and negotiations in unreliable environments. However, the stateless design of the negotiation systems introduces additional security challenges. In this paper we presented secure stateless trust negotiation. We introduced a novel method for protecting the integrity of XML negotiation states with alternating signatures. We simulated different attacks on this protocol to demonstrate how the algorithm detects forgery of negotiation states. Our experimental results showed the performance impact of different XML security algorithms. While we have only described how alternating signatures protect stateless trust negotiations, this approach can be extended to other stateless XML negotiation systems as well.

Acknowledgements

We are grateful to Blaz Primc, Alexander Schmid and Hannes Angst for their contributions to the prototype and their valuable discussions.

References

[1] W. H. Winsborough and N. Li, "Protecting sensitive attributes in automated trust negotiation," in *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*. New York, NY, USA: ACM Press, 2002, pp. 41–51.

[2] B. Smith, K. E. Seamons, and M. D. Jones, "Responding to Policies at Runtime in TrustBuilder," in *POLICY*, 2004, pp. 149–158.

[3] E. Bertino, E. Ferrari, and A. C. Squicciarini, "Trust-X: A Peer-to-Peer Framework for Trust Establishment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 827–842, Jul. 2004.

[4] B. Elisa, F. Elena, and A. C. Squicciarini, "Trust Negotiations: Concepts, Systems, and Languages," *Computing in Science and Engineering*, vol. 06, no. 4, pp. 27–34, 2004.

[5] A. C. Squicciarini, A. Trombetta, and E. Bertino, "Supporting Robust and Secure Interactions in Open Domains through Recovery of Trust Negotiations," in *ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 57.

[6] N. Dragoni, F. Massacci, and A. Saidane, "A self-protecting and self-healing framework for negotiating services and trust in autonomic communication systems," *Computer Networks, Special Issue on Autonomic Networks*, 2008.

[7] A. Klenk, F. Petri, B. Radier, M. Salaun, and G. Carle, "Automated Trust Negotiation in Autonomic Environments," in *IWSOS*, 2007.

[8] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, E. Simon, D. Eastlake, J. Reagle, and D. Solo, "XML-Signature Syntax and Processing," W3C Recommendation, Cambridge, MA, USA, Feb. 2002.

[9] J. Boyer, "Canonical XML Version 1.0," W3C Recommendation, Cambridge, MA, USA, Jan. 2001.

[10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[11] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, United States, 1997.

[12] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing," in *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1994, pp. 216–233.

[13] A. J. Lee, M. Winslett, J. Basney, and V. Welch, "Traust: a trust negotiation-based authorization service for open systems," in *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2006.

[14] J. Li, J. Huai, J. Xu, Y. Zhu, and W. Xue, "Tower: Practical trust negotiation framework for grids," in *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2006.