MOBILE TRUST NEGOTIATION

Authentication and Authorization in Dynamic Mobile Networks

Timothy W. van der Horst, Tore Sundelin, Kent E. Seamons, and Charles D. Knutson
*Brigham Young University, Provo, Utah**

Abstract We examine several architectures for extending the nascent technology of automated trust negotiation to bring nonidentity-based authentication and authorization to mobile devices. We examine how the location of trust agents and secure repositories affects such a system. We also present an implementation of one of these models. This protocol leverages software proxies, autonomous trust agents, and secure repositories to allow portable devices from different security domains (i.e., with no pre-existing relationship) to establish trust and perform secure transactions. This proposed system is called surrogate trust negotiation as the sensitive and resource-intense tasks of authentication are performed vicariously for the mobile device by a surrogate trust agent.

Keywords: Trust negotiation, authentication, authorization, access control, mobile computing, proxy, software agent, credential repository

1. Introduction

Interpersonal transactions are often contingent upon relevant attributes of the involved parties (e.g., nationality, age, job title, financial resources, etc.). These transactions can be quite intricate and involved. In the digital world, such interactions have historically been viewed as static identity-based schemes, handled out-of-band using alternative means, or simply avoided. One proposed solution for this problem of real-time, attribute-based digital interactions is called *automated trust negotiation* [WSJ00, BS00, WYS+02] (see Section 2).

*This research was supported by funding from DARPA through AFRL contract number F33615-01-C-0336 and SSC-SD grant number N66001-01-1-8908, the National Science Foundation under grant no. CCR-0325951 and prime cooperative agreement no. IIS-0331707, and The Regents of the University of California.

Trust negotiation appears well-suited for a mobile environment because mobile devices usually operate outside their trusted domain and thus have a greater need to determine whether a stranger can be trusted. This application becomes particularly compelling in light of the proliferation of such devices, their associated usage models, and their intuitive contextualization as digital representatives of their respective users.

The development of such a system presents significant obstacles. Mobile devices, due to their size, ease of transportation, and high value, are both ideal targets for theft and prone to physical accidents which can lead to their demise. In addition, they can be easily lost. Trust negotiation relies upon elements of public key cryptography and policy compliance checking that are often excessively burdensome on mobile devices. Also, because mobile network topologies are often unpredictable, such a system must handle interactions between devices of mixed capabilities in varied infrastructure configurations. Limited resources, battery limitations, processing power, and connectivity also plague mobile devices.

We use the foundation of trust negotiation to examine an advanced authentication system compatible with the limited capabilities of many mobile computing devices, and present one solution to this problem. The goal of our system is to enable mobile devices to safely and efficiently perform sensitive transactions on behalf of their owners in circumstances in which this was previously not possible.

2. Trust Negotiation

Mobile trust negotiation is designed to support automated trust negotiation between strangers that meet in the physical world and desire to perform sensitive transactions between their mobile devices (PDA, cell phone, etc.). For example, suppose two military groups from separate nations meet on the battlefield while conducting joint operations. The commanders desire to authenticate and authorize each other in order to reliably share fresh information on enemy positions and tactics. During a natural disaster, emergency response personnel from local, state, and government agencies converge to the scene and desire to share information with authorized personnel. A consumer can complete an e-commerce transaction while in an airport and be assured that he is communicating with a trustworthy business.

Trust negotiation solves the problems associated with classical authentication and authorization schemes by allowing individuals outside a local security domain to safely access sensitive data and services [WSJ00, BS00, WYS+02]. It enables two parties to perform secure transactions by first establishing trust through a bilateral, iterative process of request-

ing and disclosing digital credentials and policies. Digital credentials are the electronic analogues of paper credentials, and may be used to verify such attributes as identifying information, licensing certifications, and association memberships. These credentials are digitally signed by an issuer and assert the veracity of certain attributes of the owner. The properties of public key cryptography guarantee that these credentials are both unforgeable and verifiable.

Along with credentials, trust negotiation relies on access control policies, which protect sensitive resources such as services, data, credentials, and even other policies from unauthorized access. By specifying the necessary credentials that a party must possess in order to access a specific resource, policies provide a means by which any user may be granted or refused access to a resource in real-time. Associating policies with particular resources allows trust negotiation to thrive in a dynamic environment in which users and resources are constantly changing. As both parties in a given transaction may have sensitive resources protected by applicable policies, trust negotiation often occurs with respective parties progressively fulfilling the other parties' policies while iteratively making policy-based credential requests of their own.

Trust negotiation has two main requirements in order to operate. First, a trust agent is needed to perform a negotiation on the user's behalf. Second, a secure repository is needed to store the sensitive information that is needed by the trust agent during the negotiation. This information includes, but is not limited to, credentials, private keys, and policies.

Trust agents are intelligent, autonomous software modules that can be used to establish trust on behalf of their owner with another trust agent. An agent makes use of access control polices to protect and manage its owner's credentials, policies, and keys during a negotiation. There are various configurations for a trust agent in a mobile environment. When the trust agent resides on the device, it is called a local agent. When it does not reside on the device, it is called a remote agent. TrustBuilder [WYS+02] is an existing implementation of a trust negotiation agent.

3. Secure Repositories

A secure repository is necessary to store the sensitive data used for trust negotiation. Repositories can be local or remote.

3.1 Local Repositories

A local repository, as its name indicates, is stored locally on the user's device. Many existing schemes, such as an encrypted file system or a

set of encrypted files, can be used to store sensitive data locally. The encrypted storage can be protected by a password or biometric possessed by the user. There are several application specific methods, e.g., Window's EFS, as well as several widely deployed standards, such as PKCS#12, that could be used to achieve this result.

Another type of local repository stores the sensitive data in an encrypted form on a secure module that could be attached to the mobile device. An example of this is Sony's Memory Stick. Through the use of MagicGate [Ara00], a Memory Stick can store its contents in an encrypted form and only release them to someone that can successfully authenticate.

Another secure module that could be used with a mobile device is a smart card. Smart cards have several advantages over other local repositories. Access to the card is protected by a password or biometric. The private keys never have to exist outside of the card since all necessary processing can be done within the card. However, the space available on these cards is very limited. There is normally about 32KB of space for both an application and its data. It makes sense, therefore, that only the private keys should be stored on the smart card. Any other data could be stored in an encrypted form on the mobile device.

In order to compromise the user's sensitive information the mobile device, the smart card, and either the password or biometric used to access the card would all have to be compromised. This dilutes the risk of carrying sensitive information in a mobile environment. The physical attributes of the smart card lend itself to the protections of physical credentials while maintaining their digital protection properties as well.

A local repository provides several advantages. First, it requires no communication with remote devices. The user can also choose whether all or a subset of credentials should reside on the device. The user's sensitive information is always with him and available for use. Many existing, widely-tested systems are currently available.

Local repositories also have several disadvantages. One problem is synchronization. If a user has several mobile devices, he has to replicate his sensitive information on every device. This could be considered less secure, because there are more copies of the sensitive information and thus a greater possibility that one of the copies of the data would be compromised. When a credential expires, is revoked, or for any reason needs to be updated or removed, the changes would have to be replicated on every device that is possessed by the user. This could be a costly and time-consuming process. Also, since the repository is local, one must always be in possession of the repository when access to the sensitive information is desired.

3.2 Remote Repositories

A remote repository provides a central location for a user to store and manage his credentials. The remote repository could be administered by the user on a machine of his choosing or he could delegate that responsibility to a trusted third party to host it on his behalf.

Remote repositories can be divided into two different categories. Sandu et al. [SBG02] define these categories as virtual soft tokens and virtual smart cards. Virtual soft tokens are a network-based storage solution of sensitive credentials. Credentials are located on an online server and are stored in an encrypted form such that only the user may decrypt them. Since the server cannot decrypt the credentials, the user is in control of their disclosure. When the user desires to use his credentials, he can authenticate to the server and retrieve his encrypted credentials; he can then decrypt and use them. Ideally the credentials should be cleared from the device when the transaction that required them has completed. This would prevent undue exposure of the sensitive credentials and keys. An example of a virtual soft token is the MyProxy system [NTW01]. Another example is the proposed standard: Securely Available Credentials (SACRED) [AF01][GJN04][Far03]. This standard has the added benefit of application and device-independence.

In the virtual smart card paradigm, the remote repository acts like a smart card. There are, however, several subtle differences between these two solutions. In contrast to the physical card, the private key is never completely known to the virtual card. This is accomplished through the 3-key RSA algorithm. In this algorithm the private key is split into two-parts, the user and the virtual smart card each hold a part of the key. Through a shared signature scheme, the two parties can create a valid digital signature that neither side by itself could create. Virtual smart cards also have the benefit of instant revocation. Removal of the server-side component neutralizes a compromised user-side key. Unfortunately, not all RSA keys can be converted into the 3-key format and thus it is not plausible to move many existing certificates to this system.

NSD Security's Practical PKI [BYBS03] is an example of a virtual smart card. It uses Microsoft's Cryptographic API or PKCS #11 to make the credentials available to any application.

Remote repositories offer many advantages. Credentials are always up-to-date and are accessible from any location, even if the user does not have his mobile device with him. They have the added bonus of being application and device independent. Also, there is no sensitive information stored on the mobile device, so if the device is ever lost the

credentials are not lost with it. Since the online repository is unable to decrypt the credentials stored there, the user controls their disclosure.

There are several disadvantages which plague remote repositories. They must be available at transaction time and thus create a dependence on a third party in order to complete the transaction. If the online repository is not accessible from where the mobile device is located, or is merely not accessible, it is useless. An online repository creates an additional communication overhead: each time a transaction requires credentials, the mobile device must interact with it.

3.3 Hybrid Repositories

Both local and remote repositories have their benefits and drawbacks. The local repositories have very little communication overhead, and do not require access to an online server at the time of the transaction. They also, however, require that the user bring the repository with them, and the propagation of updates in this model can become complicated. Remote repositories, on the other hand, always have up-to-date credentials and allow the user to access those credentials from any device of his choosing. Since the mobile device contains no sensitive credentials, when the device is lost, nothing but the device is lost. However, the communication overheard, accessibility, and availability issues can limit the effectiveness of online repositories.

A combination of these two systems could lead to the elimination of many of the drawbacks that are inherent in these two repositories. We propose that a virtual soft token be used with a physical smart card to accomplish this agglomeration.

The smart card would first authenticate the user, and then be used to authenticate to the online repository. A local repository of all or some of the sensitive credentials in the repository could then be created. For added security the local cache could be created such that only the smart card would be able to access the decrypted contents. The smart card could have the private keys preloaded, or it could receive the private keys directly in an encrypted form from the repository. A smart card could do the decryption of the sensitive credentials that are stored on the mobile device, or it could give the decryption key to the application that requires the credential. Both should be made available as an option.

A user should also be able to choose to go fully remote, fully local (a full copy still resides in the remote repository), or a mix of the two. This configurability would provide great flexibility to the user, which is essential due to the wide variety of situations that exist in the mobile

environment. In any case, a user would require something he knows (his password) and something he has (his smart card).

There are several disadvantages that still exist in this model. Depending on the configuration, there could still be a communication overhead and availability/accessibility issues. There is also the cost of the additional hardware required, e.g., the smart card and smart card reader. The cost of this hardware, though, is rapidly decreasing. The additional hardware can be lost or stolen. Hopefully, since a smart card looks like a credit card, users will be able to treat it with similar regard and thus keep it safe, e.g., not leaving it attached to the mobile device.

4. Surrogate Trust Negotiation

We have created a *surrogate trust negotiation* prototype system that makes use of the ideas presented above. We adapted the trust negotiation agent, TrustBuilder, to negotiate trust on behalf of the user even if the user cannot directly communicate with it. This type of agent was chosen so that we could encompass the greatest range of mobile devices based on the resource requirements of a remote agent.

Although the hybrid repository discussed above shows promise for use in this environment, the creation of such a repository is left as future work. In the system presented below the trust agent maintains a local repository with the user's credentials. Even though the repository is local to the trust agent, it can be seen as a remote repository to the mobile device. This creates centralized storage that adds security and convenience to the system by avoiding the dangers of storing sensitive credentials on mobile devices and by allowing credential updates to be immediately accessible to all the user's devices. A user's mobile devices share a pre-existing relationship that enabled remote invocation of the trust agent. This relationship can be terminated by either side if the mobile device is compromised (see Section 4.2).

The mobile devices directly involved in the transaction are called *primary devices*. The requester of a transaction is referred to as the *client*, while the other device is the *server*. These designations, client and server, are not static as it is reasonable to assume that both will routinely switch roles as one requests a transaction from the other and vice versa. In surrogate trust negotiation, a *proxy* is any device that serves as an infrastructural intermediary between a primary device and its associated trust agent.

Figure 1 illustrates three general topologies that effectively categorize our usage models: *bilateral*, *unilateral*, and *intermittent* access to a wired infrastructure. *Bilateral* describes scenarios in which both primary de-

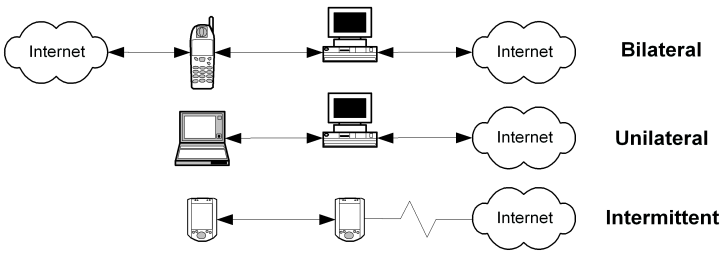


Figure 1. Topology Taxonomy.

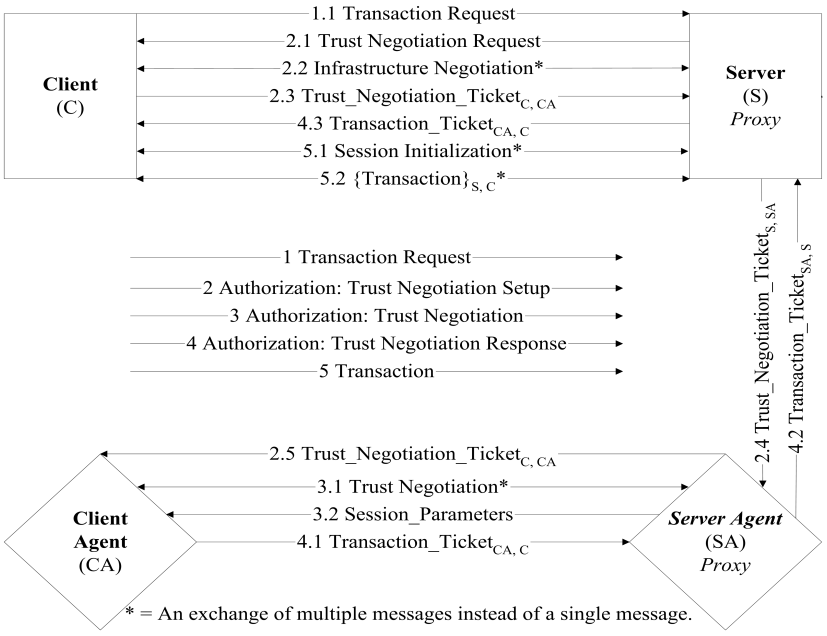
vices have reliable, economical, and adequate bandwidth to the Internet. The next, *unilateral*, describes any situation in which only one device has a consistent connection with sufficient bandwidth. The final categorization, *intermittent*, depicts situations in which neither device has consistent access to a wired infrastructure. For clarity and brevity we will explore our system in terms of a unilateral topology only, though this system would work equally as well in the bilateral topology. In the unilateral topology one device will serve as a proxy to forward the negotiation request of the other device to its trust agent.

4.1 Networking Messages

Our surrogate trust system is designed for platform independence and operability with numerous networking protocols. This section presents a high-level discussion of the elements necessary to perform trust negotiation and establish secure communications for the transaction.

For simplicity, we will discuss the networking messages as occurring in three distinct phases: *transaction request*, *authorization*, and *transaction*. In the authorization phase, it is logical to further divide this phase into three sub-phases: *trust negotiation setup*, *trust negotiation*, and *trust negotiation response*. These phases and their composite messages appear in Figure 2.

An exchange begins with the transaction request phase, in which the client requests a transaction from the server. This is represented by the *Transaction Request* message, 1.1, in Figure 2. The trust negotiation setup phase begins when the server replies to the client and indicates that the requested transaction is protected and that trust negotiation must be used for authentication (shown as the *Trust Negotiation Request* message, 2.1). If the client is incapable of performing the trust negotiation protocol or chooses not to participate, this is communicated and the connection is broken. Otherwise, both devices then decide together



{payload}$_{X, Y}$ = A message payload that is formatted for the end-to-end connection from host x to host y.

Trust_Negotiation_Ticket$_{C, CA}$ = CA_ID | {Transaction_ID}$_{C, CA}$

Trust_Negotiation_Ticket$_{S, SA}$ = {Trust_Negotiation_Ticket$_{C, CA}$ | Trust_Negotiation_Role | Trans_ID}$_{S, SA}$

Session_Parameters = Sesssion_ID | Client_Write_Session_Key | Client_Authentication_Session_Key | Server_Write_Session_Key | Server_Authentication_Session_Key

Transaction_Ticket$_{CA, C}$ = {Trust_Negotiation_Result | Session_Paramters}$_{CA, C}$

Transaction_Ticket$_{SA, S}$ = {Transaction_Ticket$_{CA, C}$ | Trust_Negotiation_Result | Session_Paramters}$_{SA, S}$

Figure 2. Network Messages.

which has the best access to the Internet in order to serve as a proxy (shown as *Infrastructure Negotiation*, 2.2).

Following infrastructure negotiation, both client and server create a *Trust_Negotiation_Ticket* (2.3,2.4) that is sent to their respective trust agent. The ticket reliably notifies the trust agent that its associated primary device desires to participate in trust negotiation for a specific transaction. When the server acts as the proxy, the client sends a ticket to the server bundled with the location of the client's security agent. On the other hand, when the client serves as the proxy, the server creates a similar ticket but also includes an identifier for the requested transaction and its associated policy. The nature of trust negotiation tickets will be further discussed in Section 4.2.

Following the receipt of the appropriate trust negotiation ticket, the device that has been elected to function as the proxy connects to the

server's trust agent and sends a message containing tickets from each primary device, 2.4. The server's agent examines the ticket from its respective primary host and verifies its request to negotiate trust. Following this confirmation, the server's agent connects to the client's agent and sends the appropriate trust negotiation ticket, 2.5. The client's agent will then likewise verify the validity of its ticket.

After both security agents have verified their associated primary devices' intentions, the trust negotiation portion of the authentication phase begins. Since the server is the device that is protecting the transaction, its security agent is responsible for initiating trust negotiation between the agents. As was briefly mentioned in Section 2, the server's security agent begins the process by disclosing policies and/or credentials to the client's agent, which then responds likewise. This bilateral exchange, 3.1, continues until the server's agent deems that the policy (included in the *Trust_Negotiation_Ticket*) governing the transaction has been satisfied or that the negotiation has failed. Factors that could contribute to a failure include the lack of necessary credentials, expired credentials, or the number of iterations exceeding a threshold.

Upon completion of a successful trust negotiation, the client and server trust agents establish the cryptographic key material (see Section 4.2) necessary to create a secure link for performing a transaction between the primary devices. This key exchange is denoted by the *Session_Parameters*, 3.2, message in Figure 2. Following this, the trust negotiation response portion of the authorization phase begins and this key material is sent back through the proxy to the respective primary devices in the form of *Transaction_Tickets* (4.1, 4.2, 4.3). If trust negotiation was successful, the primary devices decrypt these tickets and use the session parameters that they contain to initialize the secure link, which is depicted by the *SessionInitialization* exchange, 5.1. When channel initialization is complete, the server and client can securely perform the sensitive transaction, 5.2.

4.2 Security Provisions

There are three security goals necessary for our proposed system: integrity, authenticity, and confidentiality. The reasons for these goals are threefold. First, all are required to reliably initiate trust negotiation. Second, they ensure the safe delivery of key material to the primary devices following trust negotiation. Finally, they allow for a secure transaction between the primary devices following delivery of key material.

Another vulnerability to consider is that of a compromised mobile device. An ameliorating factor on the extent of potential damage is that

the trust agent, the credentials, and keys reside on a physically secure server. These keys never leave this machine, and thus, even if one mobile device is compromised, these private keys as well as imprinted relationships with other devices remain secure. Furthermore, termination from the security agent's side of an end-to-end link is trivial if a user suspects a device has been compromised and re-initialization likewise in the case that the device is later recovered.

Cryptographic Tickets. Since the mobile devices may not be in direct communication with their trust agents, they must use another method to send reliable and confidential messages to their trust agents through a non-trusted third-party. This can be achieved using a cryptographic ticket, an encrypted container that holds data. These tickets are securely communicated between a mobile device and its associated trust agent because they are encrypted according to the pre-established relationship that was formed between these entities.

Our system uses two types of tickets. The *Trust_Negotiation_Ticket* is an instruction created by the primary devices to its trust agent to initiate or accept the request to initiate trust with another entity. *Transaction_Tickets* contain the result of the negotiation and, if successful, the *Session_Parameter* message which contains the keys necessary to form a secure channel between the primary devices.

Secure End-to-End Protocol. The key material that was generated by the trust agents on behalf of their primary devices is used to create two different kinds of keys: an authentication key, and a write key. Each side uses a unique key to encrypt messages and a different unique key to encrypt a message verification. This creates a total of four keys. Using these keys, a secure session is then initialized between the primary devices as specified by the selected transmission protocol. This exchange is depicted by the *SessionInitialization* messages in Figure 2.

Following session initialization, all transmitted messages will be formatted according to the security provisions of the selected transmission protocol. For example, IPSec's Encapsulating Security Payload (ESP) [Ken02] protocol is capable of providing all of our target characteristics (i.e., connectionless sessions, integrity, authenticity, and confidentiality). However, an actual implementation of the system can use any established protocol which fulfills our definition of an end-to-end link. In general, end-to-end messages will be formatted by wrapping the payload data with the necessary header information and then encrypting and authenticating the result. In Figure 2, messages formatted according to the end-to-end protocol are denoted by the syntax $\{\text{payload}\}_{sender,recipient}$.

Thus, using the procedure described above, the security characteristics of the implemented connectionless protocol, in this case ESP, can be leveraged to secure the communication channels. However, as the channel between trust agents is assumed to be secure by virtue of the utilized trust negotiation protocol (e.g., [HJM+02]), only the channels between the primary devices and between a primary device and its associated trust agent need adhere to our definition of an end-to-end link.

4.3 Implementation

We have implemented a surrogate trust negotiation prototype system [Sun03]. The hardware core of the prototype system was comprised of two WiFi-enabled iPAQ handhelds running Microsoft's Pocket PC operating system, which served as the primary devices. The physical and link layer of the primary channel was 802.11b. On top of this, basic TCP/IP sockets were used for communication between the primary devices. Both the server trust agent and client trust agent were run on Pentium 4 machines running Windows XP. The SOAP RPC protocol was used as a means of communication between negotiating trust agents as well as between primary devices and their respective trust agents.

5. Conclusions and Future Work

We have examined the role of secure repositories and trust agents in an architecture for enabling secure transactions between portable devices that have no pre-existing relationship. We have shown how the decision of the type of repository affects the safety of a user's sensitive information while in a mobile environment. Also, the choice of repository determines what types of mobile devices can benefit from this architecture.

We have outlined surrogate trust negotiation, a flexible model that effectively leverages the combined capabilities of network proxies, software agents, and secure repositories. This system also makes trust negotiation accessible to the greatest number of mobile devices since it shifts the resource-intensive task of authentication to a remote agent. The use of a local repository on this remote trust agent allowed us to obtain many of the desirable properties of a remote repository. Surrogate trust negotiation lays the foundation for the maturation of effective, new technology in the rapidly evolving research space of secure mobile transactions.

The system, however, is only suitable for the bilateral and unilateral topologies. We are currently working on a system that will satisfy the requirements for intermittently connected devices. The foremost problem in this topology is the inability to access a remote trust agent. Consequently, the resource-intensive task of authentication must be ac-

complished on the mobile device by a completely local trust agent. We are also working on a system that would provide the user with the flexibility to choose how and where the trust agent and repository will exist. This would involve creating a hybrid repository and trust agent capable of mixed degrees of locality and remoteness.

References

[AF01] A. Arsenault and S. Farrell. Securely Available Credentials – Requirements. *IETF Informational RFC 3157*, August 2001.

[Ara00] S. Araki. The Memory Stick. *IEEE Micro*, July-August 2000.

[BS00] Piero Bonatti and Pierangela Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS-7)*, pages 134–143, Athens, Greece, November 2000. ACM Press.

[BYBS03] J. Basney, W. Yurcik, R. Bonilla, and A. Slagell. The Credential Wallet: A Classification of Credential Repositories Highlighting MyProxy. *Communication, Information and Internet Policy*, September 2003.

[Far03] S. Farrell. Securely Available Credentials Protocol. *IETF Internet Draft, draft-ietfcat-sacred-protocol-bss-09*, November 2003.

[GJN04] D. Gustafson, M. Just, and M. Nystrom. Securely Available Credentials (SACRED) – Credential Server Framework. *IETF Informational RFC 3760*, April 2004.

[HJM+02] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. Seamons, and B. Smith. Advanced Client/Server Authentication in TLS. *Network and Distributed System Security Symposium Conference Proceedings*, February 2002.

[Ken02] S. Kent. IP Encapsulating Security Payload (ESP). *IETF Standards Track RFC 2406*, July 2002.

[NTW01] J. Novotny, S. Tueke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. *IEEE Symposium on High Performance Distributed Computing*, August 2001.

[SBG02] R. Sandhu, M. Bellare, and R. Ganesan. Password-Enabled PKI: Virtual Smart Cards versus Virtual Soft Tokens. *PKI Research Workshop*, April 2002.

[Sun03] Tore Sundelin. Surrogate Trust Negotiation. *M.S. Thesis, Computer Science Department, Brigham Young University*, July 2003.

[WSJ00] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated Trust Negotiation. *DARPA Information Survivability Conference and Exposition*, January 2000.

[WYS+02] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. Negotiating Trust on the Web. *IEEE Internet Computing*, November-December 2002.