

An Autonomous Attestation Token to Secure Mobile Agents in Disaster Response

Daniel M. Hein and Ronald Toegl

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
{dhein,rtoegl}@iaik.tugraz.at

Abstract. Modern communication and computing devices have the potential to increase the efficiency of disaster response. Mobile agents are a decentralized and flexible technology to leverage this potential. While mobile agent platforms suffer from a greater variety of security risks than the classic client-server approach, Trusted Computing is capable of alleviating these problems. Unfortunately, *Remote Attestation*, a core concept of Trusted Computing, requires a powerful networked entity to perform trust decisions. The existence and availability of such a service in a disaster response scenario cannot be relied upon.

In this paper we introduce the Autonomous Attestation Token (AAT), a hardware token for mobile computing devices that is capable of guaranteeing the trusted state of a limited set of devices without relying on a networked service. We propose a *Local Attestation protocol* with user interaction that in conjunction with the AAT allows to prevent unauthorized access to an emergency mobile agent platform.

Keywords: Disaster Response, Mobile Agents, Trusted Computing, Attestation.

1 Introduction

In recent years, it has been proposed to use agent systems in disaster response [1,2]. Intelligent agents are envisioned to assist and help coordinating teams of robots and humans, thus increasing their efficiency and ultimately helping to save lives. In this scenario agent security is of paramount importance. Unfortunately, due to the nature of mobile agent systems, mobile agent security poses challenges that go far beyond those known from the classic client-server network paradigm.

Mobile agents are self-contained and identifiable computer programs that can move within a network and act on behalf of a user or another entity [3]. Building on the taxonomy of security threats against mobile agents presented by Jansen and Karygiannis [4], we can discern different attacker roles and scenarios. From a high level perspective, those are agent vs. platform, platform vs. agent, agent vs. agent and other vs. agent platform. Specifically, the agent platform should have the characteristics of a reference monitor which employs a set of technologies such as process separation, access control, cryptography, safe code interpretation or proof carrying code [5].

Current mobile agent systems deployed on stationary or mobile devices are not able to prove that they actually employ those security mechanisms and that their configuration has not been tampered with. A promising approach is to apply *Trusted Computing*, as propagated by the *Trusted Computing Group (TCG)*. Balfe and Gallery [6] have shown that Trusted Computing can improve the security of mobile agent systems, by building on security components integrated in off-the-shelf hardware, public key infrastructures, and trusted third parties. However, in a disaster response scenario the latter two services might not be available.

To overcome this challenge, we propose to enhance Trusted Computing based stationary and mobile devices with an additional Autonomous Attestation Token (AAT). This plug-in device additionally provides strong token based authentication for mobile agent systems.

Outline. The remainder of this paper is organized as follows: Outlines of Trusted Computing and mobile agent systems complete Section 1. Section 2 gives an overview of the related work and Section 3 presents the mobile agent systems in a disaster response scenario. Section 4 elaborates the concept of Local Attestation. Section 5 proposes a protocol that prevents access of unauthorized users and compromised systems to an emergency mobile agent system and Section 6 discusses an implementation architecture. The paper concludes in Section 7.

1.1 Trusted Computing

The *Trusted Computing Group*¹ (*TCG*) has specified the *Trusted Platform Module (TPM)* [7] for general purpose computer systems. Similar to a smart card, the TPM features cryptographic primitives, but it is physically bound to its host device. A tamper-resilient integrated circuit contains implementations for public-key cryptography, key generation, cryptographic hashing, and random-number generation.

Likewise, for mobile platforms, the *Mobile Trusted Module (MTM)* [8] has been specified. The MTM standard defines different sets of features (profiles) that implementors can choose from. All of these profiles contain the features required for the purpose outlined in this paper. Therefore, in this paper the terms TPM and MTM are interchangeable.

The TPM implements high-level functionality such as reporting the current system state and providing evidence of the integrity and authenticity of this measurement, known as *Remote Attestation* [9]. This is done with the help of the Platform Configuration Registers (PCRs), which can only be written via the *Extend* operation. A PCR with index i in state t is extended with input x by setting

$$PCR_i^{t+1} = \text{SHA-1}(PCR_i^t || x).$$

Before executable code is invoked, the caller computes the code's hash value and extends a PCR with the result. This process builds a *chain of trust*, starting from the firmware, covering bootloader, kernel, and system libraries etc., up to

¹ <http://www.trustedcomputinggroup.org>

application code. Ultimately, the exact configuration of the platform is mapped to PCR values. This property makes it impossible to hide a malicious program on a thus protected computer. If such a system state fulfills the given security or policy requirements, we refer to the system state as a *trusted state*.

The TPM is capable of signing the current values of the PCRs together with a supplied nonce. This is called a *Quote* operation. To protect the platform owner's privacy, the unique Endorsement Key, injected by the TPM manufacturer, is not used for this signature. Rather, a pseudonym is used: an *Attestation Identity Key (AIK)*. The authenticity of an AIK can be certified by an online trusted third party, called PrivacyCA [10]. Then a remote *Verifier* can analyze the Quote result and decide whether to trust the given configuration or not.

Another high-level feature of a TPM is that it can *Bind* data (often a symmetric key) to a platform by encrypting it with a *non-migratable* key. Such a key never leaves the TPM's protected storage unencrypted. An extension to this is *Sealing*. A key may be sealed to a specific (trusted) value of the PCRs. A sealed key will not leave the TPM, and it is only used by the TPM, if the PCRs hold the same values that were specified when the key was sealed. Thus, use of the key can be restricted to a single trusted state of the TPM's host computer.

1.2 Mobile Agents

Mobile agents incorporate the concepts of code mobility and user task delegation. Mobile agents are programs that are capable of acting on behalf of a user, and traveling between machines, to better fulfill their tasks. To start the transfer, the mobile agent's code, data, and state is marshaled into a package on the sending machine. This package is then sent to the target machine, where the agent is reconstructed and continues execution. To enable agents to fulfill tasks for their users, they should be autonomous, interactive and adaptable. Autonomy gives them a degree of control over themselves, interactivity allows them to interact with other agents and the environment, and adaptability enables them to react to other agents and the environment. The mobile agents paradigm offers reduction of network traffic and enables asynchronous interaction [11]. A host system, consisting of hardware, firmware, operating system and agent middleware that is capable of executing mobile agents is called agent execution environment. The overall network of agent execution environments together composes a *Mobile Agent Platform (MAP)* in which the agents may roam.

Applications for mobile agents include information gathering, web services, remote software management, network management and many more. Schurr et al. [2] introduce a mobile agent system that helps coordinating fire fights by facilitating vehicle management. This includes planning of routes, resource allocation and even deciding which fire to fight. They combine the agent system with sophisticated visualization technology to enable informed decisions by human personal in a different location. According to Scerri et al. [1] the use of mobile agents to coordinate heterogeneous teams of robots, agents, and humans provides the safest and most effective means for quick disaster response.

2 Related Work

A challenge of Attestation is to report the result of the verification to the user before he unveils confidential information (for example a password) to the system, as malicious software may display fake trust reports. Parno [12] concludes that a local channel between user and TPM is needed.

McCune et al. [13] propose the concept of an *iTurtle* device which can be trusted axiomatically. To achieve *user-observable verification* it should be as simple as possible, and thus easy to understand and certify. McCune et al. propose a USB device with LEDs indicating the trust status. The authors argue that integration of the TCG's scheme was too complex, due to the cryptographic mechanisms involved.

Mobile agent systems suffer from a wider range of security threats than the classic client-server paradigm, aptly described in [4]. These threats are difficult to counter with traditional methods of computer security. Farmer et al. [14] introduce a list of "impossible" security problems for mobile agent systems. The gist is that the MAP has to be trusted to perform as expected.

Besides the TPM, hardware security modules [15] in general have a long history of protecting cryptographic material. However, mobility and integrated support for analysis and verification of a client's state has not been considered yet. Wilhelm et al. [16] propose a specialized, tamper-proof hardware module that provides the so-called trusted processing environment for an agent execution environment. The idea is that if a client trusted the trusted processing environment manufacturer, the trust could automatically be extended to a host with such an environment. Trusted Computing, as proposed by the TCG, has been considered to increase the security of an MAP by [17]. Balfe and Gallery propose specific methods on how Trusted Computing can increase the security of a mobile agent system [6]. SMASH [18] is a trust enhanced MAP that relies on SELinux and Trusted Computing.

3 Mobile Agent Security in Disaster-Relief Scenarios

We assume that after a disaster stroke, one or more response teams are dispatched to the disaster area. The response teams are equipped with mobile computing and communication devices and use a MAP to coordinate their efforts. In this scenario we call these devices *In-Field Devices (IFDs)*. Examples for IFDs include general purpose desktop and laptop computers, PDAs and mobile phones. A prerequisite is that IFDs are able to measure their chain-of-trust; therefore being equipped with a TPM. Amongst possible applications for the MAP are the coordination of disaster response work, allocation of resources (like ambulances), or situation visualization.

Response teams may be supported by a command center that, amongst other services, operates a data center and a core network. We assume that this core network is well protected and maintained. In Figure 1 this is referred to as the trusted network. This trusted network may also encompass several trusted servers that execute computational expensive agents.

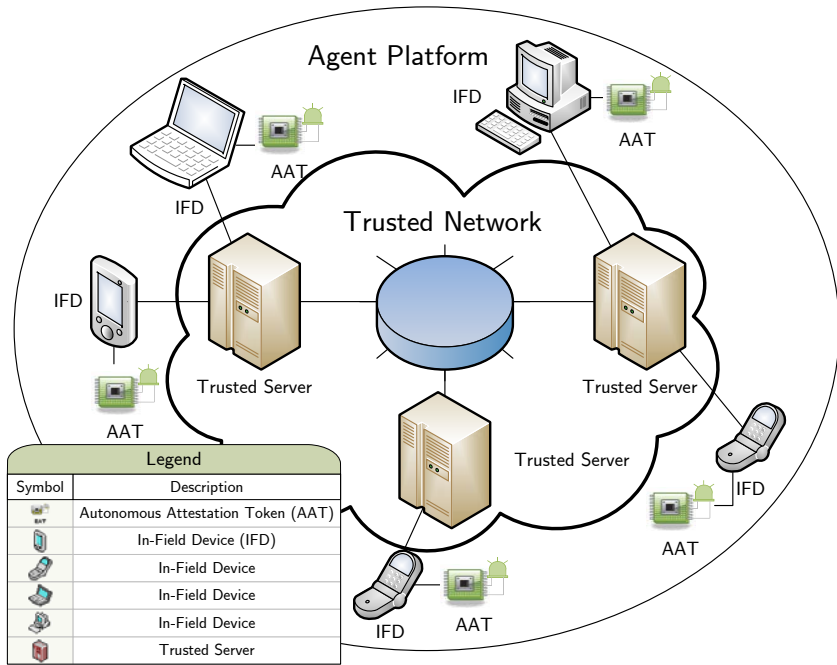


Fig. 1. Mobile agent platform for disaster response scenario

We assume that the IFDs are capable of connecting to an Internet Protocol based network. Establishing connectivity in emergency situations, possibly over multiple bearers, is by no means an easy task. Therefore, we must assume that at times the connection to the core network will be interrupted. Even worse, a core network might not exist at all, because the command center is not yet established. In this case, IFDs will form local or ad-hoc networks to support key disaster response personal.

Regardless of the underlying network topology, we assume that the MAP uses a secure network that connects the different agent execution environments. Furthermore, we assume that access to this network is secured by a single cryptographic key.

Security in such a scenario is of paramount importance. A malicious entity could cause untold harm by attacks. Attacks could encompass the intentional redirection of resources like ambulance vehicles or teams of fire fighters to wrong locations. The danger of malicious entities is especially prominent if the disaster was caused by deliberate actions, like arson or a terrorist attack.

The access to the MAP must therefore be solely restricted to authenticated and authorized personnel. In addition to a basic authentication mechanism we identify the need for Trusted Computing mechanisms to protect against tampered IFDs. It is highly desirable to enable the protection of a platform against software attacks, thus ensuring its configuration and behavioral integrity. With

TPM equipped systems, it is possible to build a chain of trust. This chain of trust does not prevent execution of malicious software, but makes all malicious software on a system evident in the system Quote.

However, current TCG-based implementations of the Remote Attestation mechanism rely on online, *remote* servers to certify the identity of the IFD, verify the authenticity of system reports, analyze the configuration of the IFD, and display the security status to the user.

The requirement for a remote party is in direct conflict with the challenges faced in disaster response, as connections to servers may fail and also no second device may be available to display the IFD's trust state.

Sealing, the TCG's alternative mechanism, may also be used off-line, but it is too inflexible: data or agent code can only be sealed to a single configuration of a single platform. When configurations change, for example by migrating agents, a central authority would be needed to assess the trusted state of the new platform and migrate the data.

For mobile platforms only, the TCG describes *Secure Boot* [19]. Secure Boot builds not only a chain-of-trust, but it also verifies intermediate states and aborts the boot process, if the system has been compromised. However, Secure Boot is restricted to a single configuration of a single device and the process is not under direct control of the user.

Usability, next to security, is an important requirement, especially in a disaster response scenario. It cannot be expected of disaster relief personnel to remember twenty character passwords or to employ similar complex security measures. Therefore, the solution must provide high security with only moderate demands on usability. Technically, high security amounts to a cryptographic MAP access key with sufficient length (for example a 256-bit AES key). We believe that plugging in a security token and entering a short four to six character password is acceptable and can also be performed in crisis situations.

4 Local Attestation

As outlined above, standard Trusted Computing mechanisms building on the TPM alone cannot fulfill the security requirements in disaster response scenarios. We propose to extend these concepts by introducing a second hardware security token, the *Autonomous Attestation Token (AAT)*. We envision the AAT to be either the size and format of a smart card or a microSD card. This allows to plug it into any available device, be it a laptop computer or a mobile phone.

A hardware security token like the AAT could potentially provide various cryptographic services. In the following sections we describe a basic use case where the AAT protects the access key to an emergency MAP in a specially shielded hardware storage unit. The key is released to the requester, if and only if the requester can provide sufficient authentication and evidence that it is in a trusted state. As the user needs to be physically present, we use the term *Local Attestation* to designate the process of deciding if a requester, in this case the IFD, is in a trusted state using our proposed AAT as verifier.

The mobile attestation token implements a true, decentralized trust decision process, which does not rely on third parties. It can easily accommodate different platforms with varying valid configurations, and it can implement a direct feedback channel to the user. The AAT, in addition to performing the platform state verification, also serves as a proof of possession in authentication protocols. Therefore, it not only ensures a specific state of the platform, but also provides evidence of the identity of the owner.

4.1 Operational Challenges

The Local Attestation approach poses several challenges related to the deployment and maintenance of AATs. Deployment is concerned with issuing an AAT to every entity that requires access to the MAP. Every AAT must contain a set of valid configurations for all devices of its owner. In addition to this, it must also contain the public part of the Attestation Identity Keys of said devices. These keys are used to sign the system Quotes of these devices.

Deployment includes revocation of an AAT. For our proposed scenario (cf. Section 3), we presume that there is no permanent central authority that grants access to the emergency MAP, thus establishing the need for Local Attestation. Local Attestation enables IFDs to ensure continuing collaboration on a distributed, yet trusted MAP. In such a case, there is also no central authority that can revoke access to the MAP. Therefore, we must rely on operational procedures, that is physically taking away the AAT from its owner.

The maintenance challenge stems from the nature of platform configuration measurements as defined by the TCG. A measurement is a SHA-1 hash of an application that is computed before the application is executed. Therefore, every time a software component of an IFD is changed, for example during an update, its hash value changes. This change must be reflected by the known-good-values database of the verifier.

We believe that careful operational practices, as they can be expected of emergency response organizations, will allow to overcome these challenges in practice.

5 An Attestation Based Key Release Protocol

In the emergency MAP scenario given above the AAT protects a single cryptographic key or a set of keys that grant access to this MAP. The objective of the protocol introduced in this work is to protect the MAP against misuse by unauthorized or malicious entities or maliciously modified platforms of authorized users. In order to achieve this goal the protocol relies on both Local Attestation and immediate user interaction. An interesting feature of our proposed protocol is that it employs a token specific return channel in the shape of a red and a green LED, similar to iTurtle proposed by McCune et al. [13].

On request by its user, the IFD initiates the connection procedure to the MAP emergency network. To achieve this end, it requires the protected key material from the AAT. The detailed key release protocol flow is illustrated in Figure 2.

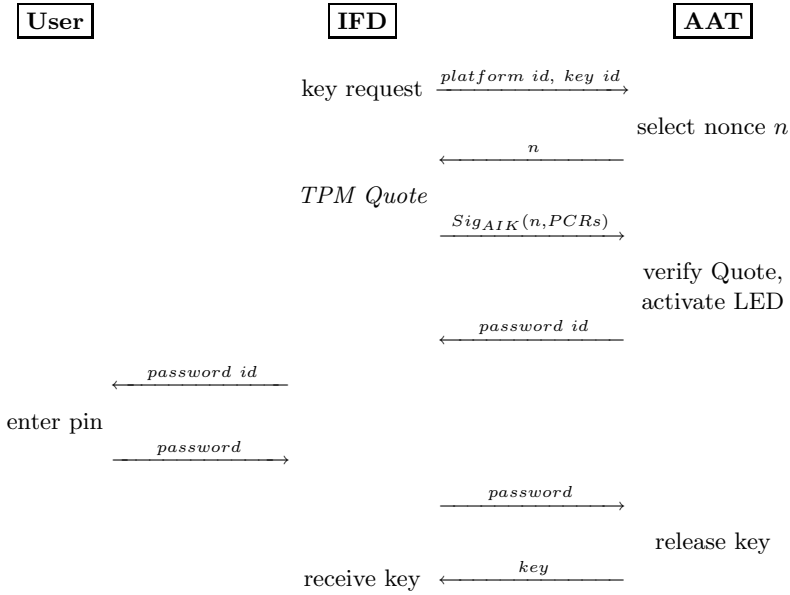


Fig. 2. Key release protocol based on Local Attestation and user interaction

Initially, the IFD sends a key request containing the unique identifier (*key id*) of the required key and a platform identifier (*platform id*). The platform identifier uniquely identifies the platform type, thus facilitating the verification process. In response to this the AAT answers with a nonce n .

The IFD forwards n to the TPM, which incorporates it in the generation of a Quote of the platform configuration. The Quote is signed by the TPM using the private part of an Attestation Identity Key (AIK). The AIK must be premeditated and the corresponding public key must be stored on the AAT. When the Quote is analyzed by the AAT, the signature, the nonce and the platform configuration report are checked. If both the nonce and the signature are acceptable, the actual trust decision is made based on the Platform Configuration Register values reflected in the Quote. This is simply a comparison of the values with a known-good-values database.

If the AAT finds the IFD trustworthy, it sends a password identification (*password id*). At the same time the AAT turns on a green LED that signals to the user that the platform is in a trusted state. In the case that the IFD is not in a trusted state, the AAT signals this by flashing a red LED. The LEDs guarantee a direct, trustworthy, one-way channel to the user.

After the trust in the IFD has been established and the password identification is sent to it, it prompts the user to enter the password for the given identification. The user checks the LED status. Only if the green LED glows, she entrusts the IFD with the required password. This password is then forwarded to the AAT which finally releases the connection key to the IFD, if the password is correct.

This protocol is immune to the general Cuckoo attack. This attack is also called grandmaster postal chess problem, and describes a technique where a malicious party forwards requests to any entity that is capable of answering them correctly [20]. A universal solution for this problem is an open research topic [12,13,21]. By requiring premeditation of the AIKs between the IFD and the AAT, it is not possible to forward the Attestation part of this protocol to just any other IFD. The Cuckoo attack can only work if this part is forwarded to one of the valid devices of the same user. Therefore, for a successful attack it is necessary to physically steal one of the emergency communication devices of the AAT owner. Furthermore, the user must be given a modified device that forwards the Attestation part of the protocol. Also it is not sufficient to steal just an AAT because of the specific AIKs and the need of a user supplied secret. Therefore, our proposed protocol increases the security system by providing protection against software attacks on the IFD.

6 The AAT Hardware Architecture

In this section, we propose a hardware architecture that is capable of implementing the Local Attestation protocol. The emergency MAP application scenario requires the AAT to cooperate with mobile devices. This influences the form factor and interface of the AAT (microSD/smart card), as well as the digital hardware design properties such as die area and circuit speed. The AAT is only necessary to establish the initial connection to the MAP, that is once per power cycle. Therefore, we conclude that circuit speed and power consumption are insignificant factors in this scenario. Verification cycles of two or three seconds seem acceptable and the only requirement on power consumption is that it has to stay below the limit of what a mobile host device can supply. This can be utilized to reduce the die area, an important factor in the total cost of the circuit. This simplifies AAT design efforts, as they can be concentrated on tamper resistance, and area minimization. The high level hardware architecture of the AAT we propose is depicted in Figure 3.

The AAT hardware architecture is composed of an RSA signature engine, a SHA-1 hash engine, a true random number generator, a Static Random Access Memory (SRAM), a Non-Volatile RAM (NV-RAM), a processor, a red LED, and a green LED. The RSA signature engine and the SHA-1 hash engine are required for the verification of the signature on the TPM Quote. The RSA engine must accommodate the same key lengths as a TPM (typically 2048 bits). The random number generator, which uses a physical randomness source with bias correction is necessary for the generation of fresh nonces. The NV-RAM contains both the secret key(s) that must be protected, and the known-good-values required for verification of the host platform state. Therefore, the AAT in general and the NV-RAM specifically must be shielded against tamper attacks [15]. The tamper resilience must be equal or better than the tamper resilience of the host platforms TPM, otherwise the AAT would provide a point of attack.

The actual verification of the host platform configuration is executed by the processor in conjunction with the SRAM. The processor also serves as controlling

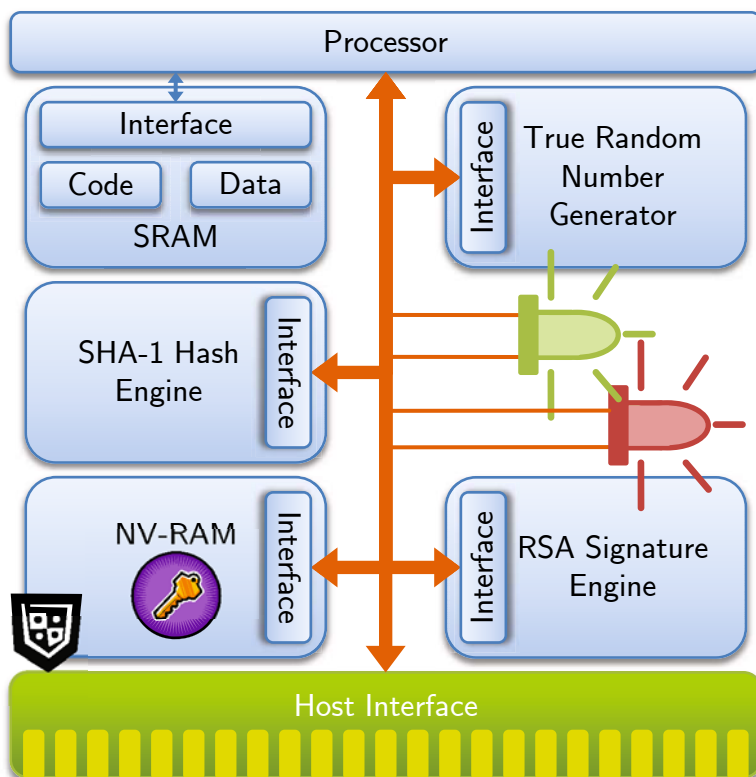


Fig. 3. The AAT architecture

instance for the complete device. For an Attestation based on known-good-values a simple 8-bit microcontroller is sufficient. The protocol supplies a platform type identifier which confines the relevant known-good-values set, thus simplifying the verification.

The architecture introduced above represents a minimal set of components to implement an AAT. Actually, the AAT functionality can be implemented on any programmable smart card or hardware security module that provides the necessary components and possesses an interface compatible with mobile devices.

7 Conclusion

Mobile agent systems and disaster response are a mixture that combines high security requirements with security risks that exceed those of the client-server paradigm. Despite this fact, the advantages and applications of this combination render it desirable. Current research indicates that Trusted Computing might provide solutions to the various security problems of mobile agent systems. In this paper we propagate the use of a Local Attestation token, the AAT, to eliminate

the need for a Trusted Computing specific remote verifier and allow decentralized, possibly ad-hoc mobile agent cooperation. Expanding on the idea of the AAT, we introduce a key release protocol that unites Attestation, a separate feedback channel, and user interaction to limit agent platform access to authorized users only. The protocol provides security against remote or local software attacks on an in-field device and is immune to a general Cuckoo attack. Furthermore, we argue that a hardware implementation of the AAT is relatively simple compared to full-featured hardware security modules or even some smart cards and propose a possible architecture. Thus, our AAT provides a powerful and simple to use security solution for mobile agents in disaster response scenarios.

Acknowledgments. The work reported in this paper was supported by the European Commission through projects SECRIKOM, FP-7, contract no. FP7-SEC-218123 and OpenTC, FP-6, contract no. 027635.

References

1. Scerri, P., Pynadath, D., Johnson, L., Rosenbloom, P., Si, M., Schurr, N., Tambe, M.: A prototype infrastructure for distributed robot-agent-person teams. In: *AA-MAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 433–440. ACM, New York (2003)
2. Schurr, N., Marecki, J., Tambe, M.: The future of disaster response: Humans working with multiagent teams using DEFACTO. In: *AAAI Spring Symposium on AI Technologies for Homeland Security* (2005)
3. Rothermel, K., Schwelm, M.: Mobile agents. In: *Proceedings of the 1st International Workshop*, pp. 155–176. Springer, Heidelberg (1997)
4. Jansen, W., Karygiannis, T.: NIST special publication 800-19 - mobile agent security (2000)
5. Necula, G.C., Lee, P.: Untrusted agents using proof-carrying code. In: Vigna, G. (ed.) *Mobile Agents and Security*. LNCS, vol. 1419, pp. 61–91. Springer, Heidelberg (1998)
6. Balfe, S., Gallery, E.: Mobile agents and the deus ex machina. In: *21st International Conference Advanced Information Networking and Applications Workshops (AINAW 2007)*, May 2007, vol. 2, pp. 486–492 (2007)
7. Trusted Computing Group: TCG TPM specification version 1.2 revision 103 (2007), <https://www.trustedcomputinggroup.org/specs/TPM/>
8. Trusted Computing Group: TCG mobile trusted module specification version 1.0 revision 6 (June 2008), <https://www.trustedcomputinggroup.org/specs/mobilephone/>
9. Coker, G., Guttman, J.D., Loscocco, P., Sheehy, J., Sniffen, B.T.: Attestation: Evidence and trust. In: Chen, L., Ryan, M.D., Wang, G. (eds.) *ICICS 2008*. LNCS, vol. 5308, pp. 1–18. Springer, Heidelberg (2008)
10. Pirker, M., Toegl, R., Hein, D., Danner, P.: A PrivacyCA for anonymity and trust. In: *TRUST 2009*. LNCS, vol. 5471. Springer, Heidelberg (2009)
11. Pham, V.A., Karmouch, A.: Mobile software agents: an overview. *IEEE Communications Magazine* 36(7), 26–37 (1998)
12. Parno, B.: Bootstrapping trust in a “trusted” platform. In: *HOTSEC 2008: Proceedings of the 3rd conference on Hot topics in security*, Berkeley, CA, USA, USENIX Association, pp. 1–6 (2008)

13. McCune, J.M., Perrig, A., Seshadri, A., van Doorn, L.: Turtles all the way down: Research challenges in user-based attestation. In: Proceedings of the Workshop on Hot Topics in Security (HotSec) (August 2007)
14. Farmer, W., Guttman, J., Swarup, V.: Security for mobile agents: Issues and requirements. In: National Information Systems Security Conference, NISSC 1996 (1996)
15. Anderson, R., Bond, M., Clulow, J., Skorobogatov, S.: Cryptographic processors—a survey. *Proceedings of the IEEE* 94(2), 357–369 (2006)
16. Wilhelm, U., Staamann, S., Buttyan, L.: Introducing trusted third parties to the mobile agent paradigm. In: Vitek, J., Jensen, C. (eds.) *Secure Internet Programming*. LNCS, vol. 1603, pp. 471–491. Springer, Heidelberg (1999)
17. Wu, X., Shen, Z., Zhang, H.: The mobile agent security enhanced by trusted computing technology. In: International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2006), September 2006, pp. 1–4 (2006)
18. Pridgen, A., Julien, C.: SMASH: Modular security for mobile agents. In: Choren, R., Garcia, A., Giese, H., Leung, H.-f., Lucena, C., Romanovsky, A. (eds.) *SELMAS*. LNCS, vol. 4408, pp. 99–116. Springer, Heidelberg (2007)
19. Dietrich, K., Winter, J.: Secure boot revisited. In: *TrustCom 2008 Proceedings*, in *ICYCS Proceedings*, pp. 2360–2365 (2008)
20. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton (2001)
21. Stumpf, F., Tafreschi, O., Röder, P., Eckert, C.: A robust integrity reporting protocol for remote attestation. In: *Proceedings of the Second Workshop on Advances in Trusted Computing, WATC 2006 (Fall 2006)*