Information Security Journal: A Global Perspective
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t768221795

Trust Ensuring Crisis Management Hardware Module

Apostolos P. Fournaris [a]
[a] Hitachi Europe SAS, Information and Communication Technologies Lab., European R-D Centre, France

Online publication date: 13 April 2010

To cite this Article Fournaris, Apostolos P.(2010) 'Trust Ensuring Crisis Management Hardware Module', Information Security Journal: A Global Perspective, 19: 2, 74 — 83
To link to this Article: DOI: 10.1080/19393550903404910
URL: http://dx.doi.org/10.1080/19393550903404910

Information Security Journal: A Global Perspective, 19:74–83, 2010

ISSN: 1939-3555 print / 1939-3547 online
DOI: 10.1080/19393550903404910

Taylor & Francis
Taylor & Francis Group

Trust Ensuring Crisis Management Hardware Module

Apostolos P. Fournaris
Hitachi Europe SAS, Information and Communication Technologies Lab., European R-D Centre, Sofia Antipolis, France

ABSTRACT Mobile agent systems (MAS) suffer from security holes that in a crisisdisaster management system can be fatal. Trusted computing group's TPM chip can be used to solve the problem but only partially. The extreme physical conditions and particularities of the crisis management agent platform do not permit the full exploitation of the TPM's features. To solve this problem the use of a special purpose hardware module, physically connected to a host crisis management device as a local trusted third party, has been proposed. In this paper, we analyze the functionality and structure of such a hardware module, called Autonomous Attestation Token (AAT) and show how a successful attack can be launched on it. To counter this attack, we propose a more sophisticated key release protocol for the communication between the AAT and the host device. This is achieved by securing the communication channel between the two devices. Also, a detailed hardware structure of the AAT is proposed. This hardware structure support the proposed key release protocol. To further analyze this, we identify the basic operations needed by the AAT hardware components and propose a sequence of actions and associated signals that those components need to follow to support those operation.

Keywords Crisis management systems, mobile agent security, trusting computing, trusted platform module

1. INTRODUCTION

In a modern system, responsible for managing critically dangerous situations, a wide variety of factors should be considered if such a system is to be marked successful. Such factors could be the seamless flow of information, the data integrity and the system's efficient control and management. However, an important factor that has an increasing effect in the credibility of a crisis management system is security and trust. When dealing with emergency situations, information must be directed accurately to the correct agency but most of all this information must be delivered unchanged and protected from unintended access and malicious attacks. Therefore, the question that rises in the above scenario is how trust can be ensured when information is exchanged through the data channels of a communication system designed for critical situations such as danger, crisis events, or accidents (e.g., forest fires, dump overflows, terrorist actions, power plant failures).

Address correspondence to Apostolos P. Fournaris, VLSI Design lab, Electrical and Computer Engineering Dpt., University of Patras, Patros–Rio Campus, 26500, Greece.

Mobile Agent systems are gaining ground in modern communication because they that can offer sophisticated services to network structures with diverse characteristics. A mobile agent can move through a network and can be executed in a different machine than the one in which it was created (Braun & Rossak, 2004). Therefore, it can achieve very high flexibility. Mobile agent systems are efficient for data collection, sharing, and management; thus, they make ideal candidates for crisis management systems where such services are of fundamental value. However, the mobile agent paradigm suffer from some serious security problems (Chess, 1998) that are based on the main mobile agent principle of allowing code execution on your computer machine that is created on a different machine. There should be a well-defined guarantee that the executed mobile agent code can be trusted and that it is not part of a malicious program set by the agent issuing machine to compromise your computer.

An answer to problems of trust in computer systems is given by the Trusted Computing Group in a form of a Trusted Platform Module (TPM), a hardware module similar to a smart card that is securely bound to its host computer system (TCG, 2007). Many researchers (Wu et al., 2008; Shen & Wu, 2008; Tan & Moreau, 2001; Wilhelm et al., 1998) have remarked that trusted computing using TPM chips can be effectively applied to mobile agent systems, ensuring in that way that a mobile agent comes from a trusted source. Through a chain of trust approach, a TPM can ensure that the host computer system cannot be tampered with or misused even if the computer user is malicious. One of the basic aspects, however, of ensuring this notion of trust is through a process called "remote attestation" that involves the communication of the TPM-based computer system with a remote trusted computer system in order to request required security characteristics (e.g., keys, credentials). This service cannot always be available in a crisis situation where unpredictable events can occur, the mobile agent system may not always be fully available, and the communication channel is always under extreme stress. A solution to this problem is given in (Hein & Toegl, 2009) where an operation denoted as "local attestation" is introduced. The functionality of this operation is based on a Autonomous Attestation Token (AAT) where a series of keys, credentials per computer system (TPM based), are stored in a secure way. The retrieval of these credentials can only be done after a successful local attestation session.

In this paper, an analysis on functionality and especially on the key release protocol of the AAT is done and security problems are discovered. We describe an attack scenario on communication of the host device and the AAT where the key stored in the AAT can be eavesdropped from a third unauthorized entity. This attack is feasible because in the key release protocol of (Hein & Toegl, 2009) the communication channel is insecure and there is no way for the host device to verify if the AAT is a legitimate chip or a false malicious non AAT device. To solve these problems, a sophisticated key release protocol is proposed that can secure the communication channel and guarantee the authenticity of the AAT chip through already existing structures of the AAT chip. Also, the hardware structure of the AAT unit is investigated and a detailed AAT architecture is proposed. The interconnections between the cryptographic units inside the AAT are analyzed and the control logic of the system is described in detail. To demonstrate the exact functionality of the AAT in the proposed key release protocol, the basic operation needs for a successful protocol session are identified, analyzed, and translated into a sequence of action steps that the AAT internal units must perform to execute this session with out fail.

The paper is organized as follows. In section 2, basic trusted computing principles and the TPM structure are presented. In section 3, mobile agent systems in disaster-crisis management are described, and in section 4 attestation using hardware modules such as the AAT are analyzed. Section 5 describes the attack model that can be followed to compromise the AAT communication, and in section 6 a key release protocol is proposed and analyzed that resists the indicated attacks. In section 7, a hardware structure for the AAT is described in detail, and the AAT exact functionality is proposed. Section 8 concludes the paper.

2. TRUSTED COMPUTING AND TPM

Trusted Computing is an emerging technology developed for designing systems that can be considered trusted. The approach of this technology is based on the principle that if the use of malicious code or behavior is impossible to be executed in a computer system, then the system itself is secure and can be trusted for any possible secure transaction. For this

reason, the software and hardware devices of a trusted computer system must be measured so as to estimate the system's level of trust. The services that can be attributed to a computer system following the above methodology are determined by the level of trust that this computer system is capable of providing.

In order to better describe the functionality and principles of the trusted computing approach, a consortium of IT enterprises was formed recently. This consortium, known as the Trusted Computing Group, is responsible for applying, implementing and extending the Trusted Computing ideas to well known and established computer systems either by introducing new hardware or software modules or by designing appropriate protocols for those modules. From its moment of formulation, the TCG has done some serious work in achieving the above goals, and through the TCG, new, innovative technologies have been proposed and described.

An established such technology, realized as a full product by some companies, is the Trusted Platform Module. This is a hardware module-chip with enhanced security characteristics that operates as an independent security measuring hardware mechanism applicable to any computer system. The TPM through a specific measurement approach (TCG, 2007) ensures that the host system can be trusted and therefore can be used in a security demanding environment.

2.1. Trusted Platform Module Structure and Functionality

Currently, the TPM is a smart-card like hardware chip that is security bound to the computer system (usually soldered on the system's motherboard). In the TPM 1.2 version (TCG, 2007), the chip is equipped with all the necessary components in order to support strong security features. Apart from the I/O interface, necessary for the TPM communication with the external world, inside the TPM there are a series of cryptographic hardware components. The generation of random numbers (usually public, private key pairs) is assigned to a true random generator unit that collects entropy or use unpredictable values like thermal noise to create random number values. For digital signature and authentication-authorization support, there is a public key encryption/decryption unit and a hash function unit. At the moment, the TPM functionality supports only RSA keys and 160 bit hashes; therefore, the TPM public key encryption/decryption unit is an RSA module (supporting a variety of key lengths) while the hash function unit employ the SHA-I 160 bit hash algorithm. The TPM also supports the secure storage of security-sensitive values (e.g., public-private key pairs or measurement states) in special memory units. Those memory units are a nonvolatile, secure memory and a series of special purpose Platform Configuration Registers (PCR) that can only store an extended version of their previous values (usually a hashing of their previous value). All those units are security-protected in order to resist hardware attacks. Finally, there is a processor unit along with assorted memory RAM and ROM units that is responsible for implementing the TPM functionality.

The TPM is used for a variety of different functions. The most important one is the measurement of the system for ensuring trust. This operation is done by establishing a root of trust for the system's behavior. More specifically, the TPM that can be considered present from the power of a computer system, measures the system from boot through a daisy chain process (chain of trust). First, the TPM root of trust gains control of the system (it is usually a subset of the BIOS called secure BIOS), then the BIOS is measured and if trusted, control is passed to it. The chain of trust is established in a similar fashion through all the stages in the boot sequence of a computer system (Secure BIOS, BIOS, boot loader, OS kernel and OS). If the chain is not broken in any stage of the boot sequence, then the system can be trusted. The chain of trust functionality is implemented through the TPM's PCRs. The idea behind the PCR approach is that the data provided by each measurement are always concatenated to the value of an appropriate PCR that contains a hash value of a previous measurement. The result of this concatenation is hashed and the outcome is stored in the same PCR. This is called an extend operation. A history of the extend operations performed to a specific PCR value is kept outside the TPM. The sequence of extends on the history file, managed by software outside the TPM, is compared to the current PCR value in the TPM and must be the same for trust to be ensured.

An extension of the chain of trust functionality is the process of reporting the current system's trust state to the external environment (i.e., a remote computer system) and to provide evidence of this report integrity and authenticity. This operation, supported by the

TPM, is called remote attestation. During remote attestation, a protocol is executed between the TPM's host machine and the remote machine. This protocol involves the use of a *quote*, which is a message send by the TPM host machine including the system's trust level. This quote can be verified by the remote machine through an authentication process (included in the mentioned protocol) and thus certifies that the TPM host system can be trusted.

3. MOBILE AGENT SYSTEMS

The mobile agent paradigm has been identified by several authors as a promising and innovative way of structuring and managing services on distributed computing network systems. A mobile agent consists of code, data, and its current execution state that is controlled by an entity called agent owner. The mobile agent is usually send to be executed over the network to a different entity called the agent executor. The agent can be send in a secure way to ensure its confidentiality and its integrity and its origin can be authenticated. When the mobile agent is downloaded to the agent executor, the agent owner transfers the mobile agent control to the new agent host (agent executor). The agent executor instantiates the agent on a special environment called the agent platform (AP). The mobile agent, when is executed in the AP, can interact with services and other agents of this platform in order to fulfill the task for which it was designed by the agent owner.

An agent owner loses control of a mobile agent after the later is released in the agent platform to be managed and run by an agent executor. Therefore, the smooth functionality of a mobile agent system is related to the efficient management applied by the mobile agent execution environment (the agent executors). Apart from technical problems related to each agent executor it is the security credibility of each executor that can affect the whole agent platform. In other words, if the execution environment of a mobile agent sponsors malicious behavior then the data collected, manipulated or transmitted from this mobile agent may extrapolate the executor's malicious behavior to the whole AP. Thus, before enabling the downloading and execution of a mobile agent, the agent owner must be able to trust the agent executor. On the other hand, a legitimate agent executor cannot accept mobile agents from any source in an AP regardless of the fact that this source may have validated authentication. There can be no evidence that even a legitimate agent owner is not compromised over time in a distributed network environment and therefore publish malicious mobile agents.

The above problems can be solved efficiently through trusted computing (Wilhelm, 1999; 1998). By equipping the agent platform machines (agent owner, agent executors) with TPM chips, all involved parties are bound by trust. Each agent platform machine can use the TPM feature of remote attestation in order to be informed of the level of trust of another agent machine concerning an agent. In that way, an agent executor can know before taking control of an agent if this agent was issued by an agent machine that is not compromised, or that this agent was not tampered with by any other agent machine. In similar fashion, an agent will not be transmitted to an agent executor that has malicious behavior (e.g., works as a trojan, virus, worm or zombi machine) since through the TPM's remote attestation this will be evident and the machine will not pass the remote attestation tests.

3.1. Mobile Agent System Security in Disaster-Crisis Management

Mobile agents can be used efficiently in systems where collection, distribution of data, and manipulation on those data is of vital importance. Thus, mobile agent systems make a suitable candidate for use in crisis management systems as indicated in (Hein & Toegl, 2009). In a crisis scenario, after a disaster situation, a mobile team of a wide variety of authorities are dispatched in the area to minimize the disaster effects. The obvious choice would be to establish a voice communication network for the efficient cooperation and management of the team. However, a better handling of the crisis situation require much more than voice data. The data exchanged in a crisis management network might be on site photos, documents, notes, voice recordings (i.e., witness testimonies), or a mixture of the above. Thus, the communication requirements indicate that there is a need for a fully functional computer communication distributed network. It can be assumed that an IP-based distributed network can suffice. Each member of the team can be equipped with a computer system able to connect to the network and run the services provided by this network.

This computer system can be any device that can support network IP communication regardless of the network channel communication that it employs. Therefore, there can be a wide variety of devices involved in a crisis management network, such as notebook or desktop computers, personal digital assistents (PDAs), and smartphones under 3G-GSM, 802.11, or Tetra channel network.

The communication network can be run as a mobile agent system. There is a series of agents issued from specific trusted machines, acting as the network infrastructure, that can support the crisis management system's collect, distribute and manipulate services on each member's computer system. All the above described in field computers (IFC) along with the trusted infrastructure machines constitute the Crisis Management Agent Platform (CMAP). Every new IFC need to connect to the CMPA in order to have access to the crisis management system's services. Ideally, trust could be ensured by embedding TPM chips in every IFC. In that way, each agent owner and executor can validate another machine's trust by remote attestation. However, due to the critical nature of the crisis management network, there is a lot of diversity on the conditions that this network must work in. There a need for the CMAP network to be fully functional from the moment the first authority team arrives on the site of crisis. In that case, many parts of the network will not be functional, like server machines used for remote attestation arbiters. Also, dew to the on site conditions, which are expected by default to be harsh and hostile, network remote connectivity is bound not to be optimal. Obviously, in practice the existence of TPM cannot fully guaranty trust between IFC's since those devices cannot without failure attest themselves remotely. The TCG's TPM supports some additional mechanisms for displaying and verifying the trust state of a computer system, such as sealing and secure boot. However, those solutions are highly inflexible and offer poor usability (Hein & Toegl, 2009). Flexibility and usability are of vital importance on a crisis management system since complex procedures may very well delay the team personnel and result in critical errors in stressful conditions. Therefore, there may exist serious security and practicality holes in TCG's TPM only approach for a trusted CMAP that if malicious entities manipulate them they can cause additional damage to an already disastrous situation.

4. ATTESTATION THROUGH SECURITY HARDWARE MODULE

In order to enable a more sophisticated trust attestation mechanism, some researchers have suggested the use of additional special purpose hardware chips. The purpose of those chips is to provide a tamper proof secure and trusted environment for executing agents and support the use of trusted third party entities (Wilhelm et al., 1999) or just offer a positive or negative answer on the system's question of trust (McCune et al., 2007). However, a promising approach is described in (Hein & Toegl, 2009) proposing a device (Autonomous Attestation Token) that does not need to be burdened with the execution of whole agent systems but needs hold only the keys for using those agents and unlocking their services. This device is designed for a CMAP.

This Autonomous Attestation Token (AAT) is described as an SD card that is physically attached to an IFC and upon request from its host releases the keys related to this host only if the host provides sufficient credential that it is in a trusted state. The keys for each IFC host along with the IFC host's id and public key are stored in the AAT secure memory. We can assume that the AAT has a unique id number and a set of cryptographic keys (public, private key pair) that should not be transmitted in any way through the communication channel. Apart from the above, the AAT must hold a series of valid configuration states (PCR values) for each IFC in order to be able to verify the trust state of a host.

In a way, the AAT plays the role of a local trusted third party. The process of verifying the host's trust level is called local attestation since it is similar to remote attestation but does not require a network communication channel because it is performed locally (the AAT is attached to the host device). In a AAT enabled CMAP, the mobile agents of the CMAP are issued by trusted servers and are cryptographically secured using specific keys. In that scenario, the assorted keys of a CMPA agent are necessary if such an agent is to be managed by an agent executor (an IFC device). The keys, as proposed in (Hein & Toegl, 2009), are provided by the AAT through local attestation using the protocol of Figure 1.

The local attestation protocol uses a nonce to guaranty the freshness of the exchanged messages. Initially, the IFC has to provide the AAT with an identification

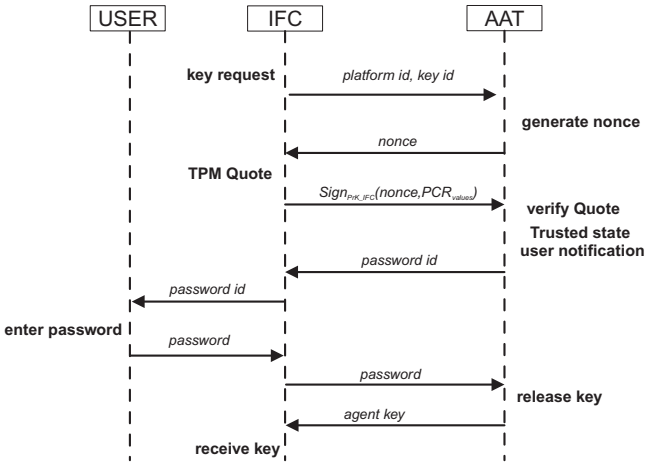

FIGURE 1 The protocol of the IFC and AAT.

of the platform it uses (platform id) and the key that requires (key id). A random value is then provided by to the IFC as a nonce. Then, the IFC configuration along with the nonce are encrypted using the private key of the IFC host and are sent under the name TPM quote to the AAT. The AAT using the platform id and key id retrieves from its memory the IFC host's public key and decrypts the TPM quote. In that way the AAT can verify that the nonce is the one sent earlier. After comparing the sent IFC configuration with the one stored in its memory, the AAT can also verify if the IFC is in a trusted state by powering on a green or red light. In the first case, it prompts the user for a password (through the power on of a green light), and upon correct retrieval it releases the requested agent key. Note that the public key of the IFC is stored in the IFC's TPM while the public key of the IFC is stored in the AAT's secure memory. Neither key is publicly known.

5. ATTACKS ON THE AAT

The AAT as described in (Hein & Toegl, 2009) is protected from man-in-the-middle attacks (MITMA) like the grandmaster postal chess problem. Also, through the use of a random nonce for each run of the protocol, replay attacks are difficult to be launched. The AAT is considered tamper proof and EMC sealed so that the operations inside this device cannot be probed. However, successful attacks are feasible on the IFC-AAT communication by listening to the communication channel. It can be assumed that an attacker is able to monitor the channel between the two entities (IFC-AAT) by implanting a monitor device in either the IFC or the AAT or by monitoring the electromagnetic (EMC) signal emission of the channel (Tanaka, 2008). This scenario is perfectly possible if a legitimate user acts in a malicious way, if a legitimate user is fooled into accepting monitoring, or if an attacker gains possession of an IFC, AAT device along with a user password (i.e., through means of social engineering). In all the above cases, trusting computing cannot be applied to avoid this attack since it is not an attack on the IFC device.

In the attack model described above an attacker can easily gain possession of both the user password and the agent keys. At first, the attacker does not tamper with the IFC device but rather leaves the protocol to unfold and monitor the channel locally. After one successful authentication and key release through the AAT, the attacker has knowledge of the user password and the key. Then he or she can tamper with the IFC device (the same device or a different one), run malicious code on it, and program the device for attacks on the CMAP. The device is not in a trusted state anymore so when inserting the AAT in order to connect to the CMAP for a second time, the key will not be released, but the attacker no longer require the AAT to have access to the key. He or she already has the key and can gain access to the CMAP without the AAT using an untrusted and malicious IFC.

Another problem of the AAT key agreement protocol is that the IFC by default is bound to consider the AAT device trusted. The AAT does not need to prove its identity in any part of the protocol. A malicious device can impersonate an AAT as long as it follows the AAT protocol and has a green and red light on it. The rogue AAT does not contain any valid known PCB values nor the public keys of any IFC host but it does not need to. Following the key release protocol, fooling the user into typing a legitimate password and storing that password in its memory is enough for compromising the whole system. If the rogue chip is equipped with a wireless transmitter it can even send the collected keys through the air to an unauthorized user without detection.

6. PROPOSED IFC-AAT KEY RELEASE PROTOCOL

To address the problems presented in the previous section, we can propose an IFC-AAT key release

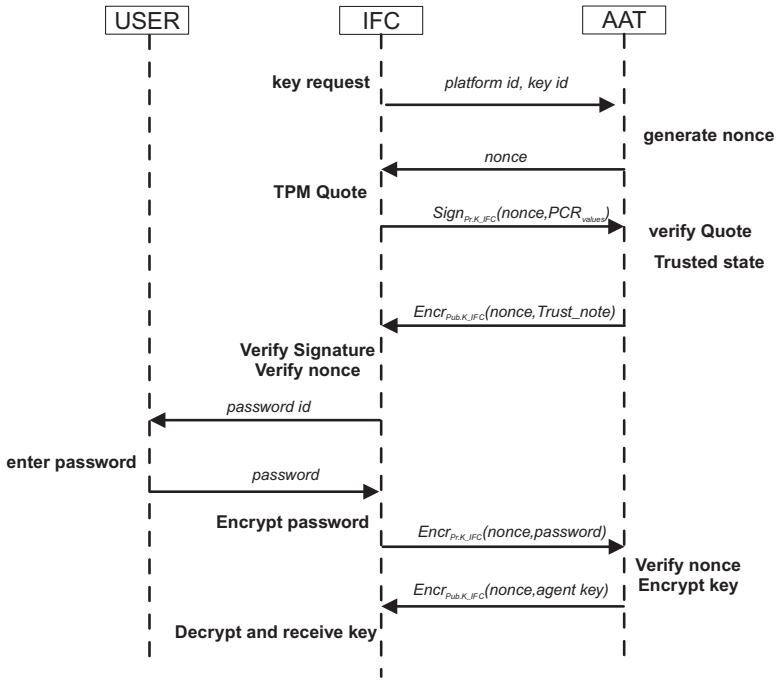

FIGURE 2 **The proposed IFC–AAT Key release protocol.**

protocol that can guaranty security even on an insecure channel without adding significant overhead to the whole process. The proposed protocol is presented in Figure 2. The key release protocol is initiated when the IFC requests a key from the AAT by providing the platform id and the key id. As a response, the AAT sends a nonce. Then, the IFC generates a TPM quote using the provided nonce and the PCR values measurement of its TPM. The TPM quote is signed using the IFC private key. The AAT receives the TPM quote and, using the associated to the IFC platform id public key, verifies that the PCR values match the one's stored in its memory. The AAT then generates a Trust note that includes an positive or negative message analogous to the PCR values matching process and a password id or a random number correspondingly. The Trust note along with the nonce are encrypted using the IFC's public key and sent to the IFC. When the message is received, the IFC decrypts it and reads the Trust note. If the trust state is an acknowledge message the IFC reads the password id and prompts the user to insert the password. The password along with the nonce is then encrypted using the IFC's private key and sent to the AAT for verification. The AAT verifies the nonce and the password after decrypting the provided message with the IFC's public key, encrypts the nonce with the agent key using the IFC's public key, and releases the result to the IFC. The key is retrieved by decrypting the AAT message with the IFC's private key and by verifying the nonce.

In the proposed protocol of Figure 2 the message exchange is protected from replay attacks through the use of a random nonce generated by the AAT. This nonce is also concatenated in every message that is encrypted to ensure that this message is related to the current run of the protocol. Also, after decryption, the message is expected to include the nonce value thus the message receiver can verify that the message is a true encrypted value and not some garbage data inserted into the channel. The channel is secured by encrypting all messages including sensitive information along with the nonce generated by the AAT. Knowledge of the IFC's public key by the AAT attests that the hardware device connected to the IFC is a legitimate AAT.

7. PROPOSED HARDWARE STRUCTURE FOR AN AAT

The hardware structure of the AAT can be determined by the functions that it must fulfill. A generic AAT design was presented in (Hein & Toegl, 2009), but details on this structure were not provided. The

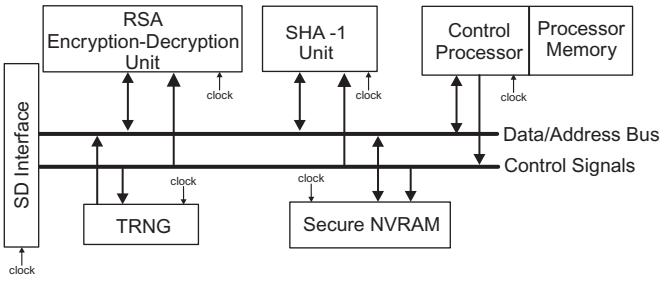

FIGURE 3 **The proposed hardware structure of the AAT.**

AAT due to its connection with a TPM has a TPM-like structure and includes an RSA signature unit, a processor unit, a nonvolatile memory unit, a true random number generator unit (TRNG), and a SHA1 hash function unit. Our goal is to evaluate the AAT hardware capabilities and propose a detailed description of the AAT hardware design along with its related functionality. Note that due to the enhancement of the IFC-AAT key release protocol with encryption-decryption the AAT hardware functionality is different from Hein & Toegl (2009).

The proposed hardware structure of an AAT chip is shown in Figure 3. The system is structured around a data/address bus where all the data values are transfered for reading by a requesting unit of the AAT. This bus also serves as an address bus connected to the memory unit for a successful memory data reading. There is also a bus connected to all the units of the chip that is responsible for controlling those units. Signals of this bus are in general managed from the processor. The processor unit is responsible for controlling the whole AAT system and realizing the key release protocol by enabling, in regard to the protocol, AAT units and performing operations that do not require the involvement of other units (i.e., comparisons or memory search). For this task, the processor has stored in its memory a microcode program implementing the protocol and a series of data required by the protocol. The SHA-1 unit is implementing the SHA-1 hash function and the RSA encryption-decryption unit is responsible for performing the arithmetic operation of modular exponentiation ($m^e mod n$) as defined in RSA public key scheme (Menezes et al., 2001). The SHA-1 unit has a data input/output and a control signal indicating the beginning of a hash function operation. The RSA unit has as inputs the modulus n value (part of the RSA public key), the message m to be encrypted-decrypted and the public or private key e along with a control signal indicating the beginning of a modular exponentiation. The TRNG unit is connected to the data path through its data output and has a control signal indicating the beginning of a random number generation. The NVRAM has a chip-select control signal and a read/write control signal while connected to the data/address bus in order to read the address and use it to write or read the data values in or out of it. The system has a universal clock and works in synchronous way.

In order to ensure a high security level, the RSA keys used in the AAT should be of length 2048 bits. As a result, the data related to the RSA encryption and decryption will have similar bit length. However, there is no feasible processing system able to operate with buses of such bits. Therefore, the 2048 bit values are broken into several blocks (to match the bus bit length) and reconstructed inside the related AAT units (the RSA encryption-decryption unit). The same problem exists with the SHA-1 unit that handles 160 bit values and is solved in a similar way.

7.1. Proposed AAT Functionality

Based on the key release operation for which the AAT was designed, the exact role of the various AAT components into the realization of the protocol can be determined. More analytically, there are the following basic operations that can be identified from Figure 2:

1. platform-key identification
2. generate nonce
3. verify signature
4. verify PCR
5. generate trust note
6. exponentiate (encrypt and decrypt)

Each above operation is executed in the processor unit with the collaboration of some of the other AAT units. The SD interface is responsible for the AAT communication with the external world by transforming the values internal to the AAT into SD format communication values and vice versa. The first message that is received from the SD interface is the interpreted from the processor as a *Platform-key identification*. So the processor reads the data that are present in the data bus for a predetermined number of clock cycles required by the SD interface in order to complete the receiving of the platform-id and key-id.

The transmitted values are compared with the values stored inside the processor memory; the addresses of the key, the premeditated PCR values, and the public key of the IFC that are stored in the NVRAM are retrieved. Those addresses are saved in the processor register file and the *platform - key identification* operation is concluded.

The *generate nonce* operation is initiated when the processor enables a control signal connected to the TRNG unit marking the beginning of random number generation. After a predefined number of clock cycles the TRNG outcome reaches the data bus, is red from the processor and saved in its memory.

The *verify signature* operation is initiated when the IFC RSA signature is received from the SD interface and stored in the processor. The whole process follows the RSA signature verification scheme (Menezes et al., 2001). Initially, the processor sends the address of the IFC public key to the address bus and enables the chip select and read signal of the NVRAM. The processor enables the control signal marked for modular exponentiation in the RSA Encryption-Decryption unit and supervises the transmission in blocks of the IFC public key to the data bus and the RSA Encryption-Decryption unit. Note that the RSA Encryption-Decryption unit needs 3 values before beginning the encryption/decryption operation. So, after receive of the public key (the public key in RSA include the modulus n and the exponent e) the processor puts in the data bus the signature to be verified, in blocks. The RSA Encryption-Decryption unit is programmed to expect after a predetermined number of clock cycles (required for receiving the public key) the signature to be verified. After the signature is received, the RSA Encryption-Decryption unit decrypts the signature to get the hash value of the signed values. In parallel, to the decryption process, since the data bus is free, the processor sends to the data bus the signed data and enables the control signal of SHA-1 unit marking the beginning of hash functioning. The whole signature verification operation is timed in such a fashion that the hash function result reaches the data bus and is saved in the processor memory when the RSA decryption is concluded. When the processor reads the data bus that has the RSA decryption result, it makes a comparison between this value and the hash function result. If the two values match, then the processor stores the PCR values. The *verify signature* operation is concluded and the *verify PCR* operation may begin.

The *verify PCR* operation is initiating by retrieving from the NVRAM the premeditated PCR values for the IFC platform. This is done by putting the address of those values in the address bus and enabling the chip select and read control signals of the NVRAM. The data that are put in the data bus and compared to the ones stored in the processor after the *verify signature* operation. The outcome of the comparison determines the operation *generate trust note* that will follow.

The *generate trust note* operation employs the PCR computation results to create an appropriate trust note. When the PCR matching operation returns a positive answer (the PCR values are the same), then the password id is retrieved from the processor register file using the key id and platform id values. Then, the Trust note is created by concatenating the positive answer (a predetermined known value) and the password id. If the PCR matching returns a negative answer (the PCR values do not match), then the trust note is created by concatenating the negative answer (a predetermined known value) along with a random number. The random number is generated by running the *generate nonce* operation.

Apart from the above operations, the *encrypt* and *decrypt* operations are also needed in order to fully realize the proposed key release protocol. Each of the two operations requires 3 input values in order to produce a result and then employ the same arithmetic operation (modular exponentiation) to come up with that result. Their realization is similar to *signature verification* operation. Initially, the processor sends the address of the IFC public key (n, e) to the address bus and enables the chip select and read signal of the NVRAM. The processor enables the control signal marked for modular exponentiation in the RSA Encryption-Decryption unit and supervises the transmission in blocks of the IFC public key to the data bus and the RSA Encryption-Decryption unit. Then it takes a similar transmission in blocks of the data to be encrypted or decrypted. After a determined number of clock cycles the RSA Encryption-Decryption unit puts the encryption or decryption result to the data bus, ready to be used by the processor. The *encrypt* and *decrypt* operations share the same realization sequence since they use the RSA Encryption-Decryption unit in the same way. They even share the same public key in the same key release session. Therefore, they can be

generalized in one operation called *exponentiate* operation. Before encryption and after decryption, the nonce should be added or verified to the associated message. The nonce should be concatenated to the message in any case.

8. CONCLUSIONS

Enhancing the security of mobile agents systems is a very important factor, especially if such systems are used in critical information systems for data gathering, managing, and decision making. Crisis-disaster management systems are among the top targets for requiring extreme security. Trusting computing offers a strong security infrastructure based on a hardware module called TPM that can achieve and enforce trust between all the involved crisis management agencies. However, this notion of trust cannot be fully guaranteed because a disaster situation is managed in conditions that are harsh, and as a result important features of a TPM chip, such as remote attestation, cannot be always applied. The use of an additional AAT smart card such as chip physically connected to a computer system that is part of a crisis management system, is needed (Wilhelm et al., 1999; Hein & Toegl, 2009). We analyzed the ideas behind the AAT, describe possible successful attacks scenarios on the AAT, and analyze the problems behind the existing key release mechanism of the AAT. As a result of this analysis, a key release protocol is proposed that does not suffer from the AAT problems. A feasible hardware structure for the AAT, fully supporting the key release protocol, is also proposed, and the functionality of this structure on building blocks is analyzed. Finally, the basic operations of the AAT hardware structure are identified and a realization sequence featuring the collaboration of the various AAT hardware units is proposed. Through the proposed approach for the AAT, we manage to solve the problem of key reading from eavesdropping the communication channel by encrypting all exchanged sensitive data without adding communication overhead to the protocol.

ACKNOWLEDGEMENTS

The work of this paper is supported by the European Commission through the SECRICOM FP7 European project under contract FP7 SEC 218123

REFERENCES

Braun, P. and Rossak, W. R. (2004). Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit. San Francisco: Morgan Kaufmann.

Chess, D. M. (1998). Security issues in mobile code systems. In In mobile agents and security (pp. 1–14). Springer-Verlag.

Trusted Computing Group. (2007). TCG TPM specification version 1.2.

Hein, D. and Toegl, R. (2009). An autonomous attestation token to secure mobile agents in disaster response. In *The First International ICST Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec 2009)*. Italy, June 3–5, 2009.

McCune, M., Perrig, A., Seshadri, A. and van Doorn, L. (2007). Turtles all the way down: Research challenges in user-based attestation. In *Proceedings of the Workshop on Hot Topics in Security (HotSec)*, August.

Menezes, J., Van Oorschot, C., Vanstone, A. and Rivest, R. L. (2001). Handbook of applied cryptography. USA: CRC Press.

Shen, Z. and Wu, X. (2008). A trusted computing technology enabled mobile agent system. *Computer Science and Software Engineering, International Conference*, 3, 567–570.

Tan, H.-K. and Moreau, L. (2001). Trust relationships in a mobile agent system. In *Mobile Agents*, 2240 in LNCS, 15–30. London: Springer.

Tanaka, H. (2008). Evaluation of information leakage via electromagnetic emanation and effectiveness of tempest. IEICE - Trans. Inf. Syst, E91-D(5), 1439–1446.

Wilhelm, U., Staamann, S., and Buttyan, L. (1998). On the problem of trust in mobile agent systems. In Internet Society's Symposium on Network and Distributed System Security San Diego, CA, USA.

Wilhelm, U. G., Staamann, S., and Buttya'n, L. (1998). Introducing trusted third parties to the mobile agent paradigm. In Secure Internet Programming: Security Issues for Mobile and Distributed Objects (pp. 471–491). Berlin: Springer-Verlag.

Wu, X., Shen, Z., and Zhang, H. (2008). Secure key management of mobile agent system using tpm-based technology on trusted computing platform. *Computer Science and Software Engineering, International Conference*, 3, 1020–1023.