Credentials and Beliefs in Remote Trusted Platforms Attestation

Andrea Bottoni and Gianluca Dini
University of Pisa
Dept. of Information Engineering
I-56122 Pisa, Italy
Email: {a.bottoni,g.dini}@iet.unipi.it

Evangelos Kranakis
Carleton University
School of Computer Science
Ottawa, ON K1S 5B6, Canada
Email: kranakis@scs.carleton.ca

Abstract— **Remote attestation in trusted computing is about the ability of a local platform to authenticate the hardware and the software stack running on a remote trusted platform. We say that this process is *successful*, if a local application is able to authenticate each layer in the remote stack; it is *meaningful* if, by using this information, the local application can make its own evaluation on the safety of the platform environment where the remote application is running.**

In this paper we analyze the credentials and beliefs that are necessary to a local application in order for the remote attestation process to be both successful and meaningful.

I. Introduction

Computer platforms are becoming widely available and are fundamental to the successful spreading of electronic business and commerce. This makes the need to protect information even more impelling, particularly on the type of platforms we use directly (e.g., PCs). The need for stronger trust and confidence in computer platforms increases with connectivity and physical mobility. In addition to threats associated with connecting to the Internet, physical mobility increases the risk of unauthorized access to the platforms including actual theft. Trusted platform technology provides mechanisms that are useful in both circumstances, by allowing systems to extend trust to clients running on these platforms.

Trusted platforms are computer platforms characterized by specialized hardware designed for security operations. Various initiatives in trusted computing [6] aim at designing software building blocks and interfaces that exploit the functionalities of the trusted platform technology. Among the several security-related functionalities that these platforms offer, *remote attestation* allows a local platform to authenticate a combination of hardware and software stack running in a remote platform. A local platform, by determining the environment of a remote platform, is in the position to better evaluate the amount of trust it is willing to extend on the remote one.

In this paper we focus on the process of remote attestation which is done by means of digital signatures, as in [4] and [3], and that rely on the existence of a hierarchical public key infrastructure for identity, certificates and key management. In this context, we analyze the credentials and beliefs that are necessary to a local application in order for the remote attestation process to be both *successful* and *meaningful*. We say that a remote attestation process is successful if, by combining the trust that a local platform has in different certification authorities, and the content of all the certificates making up the various chains, the local platform is able to identify the composition of the remote platform. On the other hand, the remote attestation process can be said to be meaningful if, by using the information obtained during the remote platform authentication and the beliefs of the local platform, the latter is able to consider the environment of the former as safe. To the best of our knowledge, this aspect has never been addressed before.

For simplicity, but without loss of generality, we will assume that the remote platform is made up of three layers, that is, the application A, which runs on top of the operating system OS, which runs on top of the dedicated security hardware, the trusted platform module (TPM).

The rest of the paper is structured as follows: we start by recalling the main constructs used by the theory employed in our analysis (Section II). Then, we analyze the credentials and beliefs that are needed when authenticating a single remote layer (Section III), and we generalize the discussion to a set of layers making up the remote platform hardware and software stack (Section IV). Finally, we analyze the credentials and beliefs needed to make the attestation process meaningful (Section V).

II. Notation

In order to understand the remote attestation process we need to be able to reason about the content of certificates, to see how they combine together into chains, and how different chains combine together with the local platform's beliefs. We, therefore, need a vocabulary for describing the content of certificates and beliefs, and rules in order to understand what can be inferred by their combination. To this end, we have chosen to model the certificates and the beliefs in this paper with a logic for authentication. This logic is introduced in [5], and has already been used for describing the concepts and primary APIs employed in Microsoft's Next Generation Security Computing Base (NGSCB) [3]. In what follows we will give a brief overview of the notation and theorems used by this logic. The description tries to catch the main aspects intuitively, and is not to be considered rigorous.

The theory described in [5] is centered around the concepts of *principals*, *channels*, and *statements*. Informally, a principal is what produces a statement. Examples are a person, a machine, or a role. A channel is a particular principal, it is the only one able to produce statements that can be recognized by a computer. Examples of channels are cryptographic keys, wires, keystrokes, IP numbers.

In this paper we are interested in channels made up of an asymmetric key pair. Using K and K^{-1} for identifying the public and private key respectively, a statement s signed by the private key corresponding to K is expressed by the theory with K **says** s. In other words, when we write K **says** s we mean that a certificate has been issued which contains the statement s and the signature performed on it by the key K^{-1}.

A certificate signed by a certification authority CA with its private key $\mathsf{K_{CA}}^{-1}$ asserting that the principal A owns the key pair ($\mathsf{K_A}$, $\mathsf{K_A}^{-1}$), is expressed with $\mathsf{K_{CA}}$ **says** $\mathsf{K_A} \Rightarrow \mathsf{A}$ and it is read as "$\mathsf{K_{CA}}$ says that $\mathsf{K_A}$ speaks for A". In the previous expression, "$\mathsf{K_A} \Rightarrow$ A" is an example of statement. Intuitively, it means that everything that is said (signed by) $\mathsf{K_A}$ is considered as said by A. Or, in other words, that $\mathsf{K_A}$ has the authority to make statements on behalf of A.

We assume that the *hand-off axiom* holds. It states that a principal has the right to specify who can speak for it. In other words, if A **says** $\mathsf{K_A} \Rightarrow \mathsf{A}$, then anyone can safely assume that $\mathsf{K_A} \Rightarrow \mathsf{A}$. This can be generalized to the case in which the statement is done by someone speaking for A (*extended hand-off axiom*). As an example, consider $\mathsf{K_{CA}}$ **says** $\mathsf{K_A} \Rightarrow \mathsf{A}$. Anyone reading this certificate and believing on the authority of CA over A (believing $\mathsf{K_{CA}} \Rightarrow \mathsf{A}$) will believe anything signed (said) by the key $\mathsf{K_A}$ as if it were said by A itself (will believe $\mathsf{K_A} \Rightarrow \mathsf{A}$).

It is possible to obtain *compound principals* by combining "simple" principals and operators. The compound principal we are interested in is the one which is obtained by using the operator *quote* (|). As an example, consider the case in which the trusted platform module (TPM) needs to relay a statement s of the operating system (OS) running on the platform. The TPM can do so by issuing the following statement: $\mathsf{K_{TPM}}$ **says** OS **says** s. This is a certificate signed by $\mathsf{K^{-1}_{TPM}}$ which contains a reference to OS, and the statement which is relayed, i.e. s. The meaning is that the TPM is "quoting" the OS when it says s. An alternative way of writing the previous certificate using the operator | is $\mathsf{K_{TPM}|OS}$ **says** s, and can be read as "$\mathsf{K_{TPM}}$ quoting OS says s". Notice that, since TPM is simply relaying a statement from OS, there is no guarantee that actually OS said s. This would, in fact, require that $\mathsf{K_{TPM}} \Rightarrow \mathsf{OS}$.

The following important axiom (which we will call *role axiom*) holds for the operator quote: for every principal A and every role R, $\mathsf{A} \Rightarrow \mathsf{A|R}$. Let us consider this axiom with reference to the TPM key $\mathsf{K_{TPM}}$ and the operating system OS running on top of the TPM. When the TPM executes the program OS, it is limiting its behavior according to a certain set of rules, that is the set of rules that correspond to the OS' specifications. Since behaving according to a set of rules can be seen as acting according to a given role, we can equate a program with a role [5]. Therefore, the previous axiom can be used in the case where the principal A is the key $\mathsf{K_{TPM}}$ and the role R is the program OS, whose ID is $\mathrm{ID}_{\mathsf{OS}}$:

$$\mathsf{K_{TPM}} \Rightarrow \mathsf{K_{TPM}}|\mathrm{ID}_{\mathsf{OS}} \tag{1}$$

Intuitively the axiom holds because the TPM has the authority to speak for itself when it is limiting its authority, such as when it is running OS. The importance of this axiom will be more evident in Section IV.

Other theorems and axioms of the logic will be used in the following sections as they are needed. In this paper, for brevity, we have chosen not to detail them, in favor of giving intuitions on the line of reasoning. We refer to [2], for a more theoretical insight on the formula used, to [5] for a rigorous description of the logic, and to [7], [1] for its application in modeling different systems.

III. Authenticating a Remote Layer

The remote attestation process allows a local platform to cryptographically authenticate the hardware and the stack of software running on a remote platform. As part of this, the local platform must firstly be able to authenticate each single layer that constitutes the remote stack, secondly to authenticate the stack in its entirety. In this section we will consider the first aspect, i.e. the authentication of a single layer in the remote platform.

In what follows, we will use the term *entity* to refer to a generic layer in the remote platform stack. So doing, the discussion in this section will apply to the authentication of both a hardware component, like the trusted platform module, and of a software image which is part of the software stack running on a remote platform.

In order to uniquely authenticate a remote entity, it is necessary to authenticate its manufacturer beforehand. In fact, besides the entity name being meaningful only in the manufacturer's namespace, the entity's key pair (or ID, if it is a software) is normally certified by the manufacturer that ships it. In the more general case, a manufacturer will be an organization with its own public key infrastructure. Often, the PKI employed will be hierarchical, and not necessarily modeled on the organizational chart.

In order to fix ideas, but without loss of generality, we will consider that a remote entity E is a product (software or hardware) made by an organization with a simple 2-layered PKI infrastructure. Within the organization, the root certification authority, which we call V_E, certifies its trusted platform compliant products by means of an intermediate certification authority, called Tr. This delegation of authority is expressed by means of the following certificate[1]:

$$K_{V_E} \textbf{ says } K_{V_E/Tr} \Rightarrow V_E/Tr/* \tag{2}$$

Tr, in turn, will generate a certificate for the product E, in order to certify its identity, as follows:

$$K_{V_E/Tr} \textbf{ says } ID(E) \Rightarrow V_E/Tr/E \tag{3}$$

Consider the case in which a platform L outside E's organization wants to authenticate E. If L receives the certificates shown above, then, it cannot infer anything useful, since it does not know who K_{V_E} is, and which authority it has. In order for L to be able to authenticate a generic entity which is located in the tree of V_E, L needs to believe that K_{V_E} is the key of V_E, and that V_E is the certification authority of the tree $V_E/*$, rooted in V_E itself. To this end, the following belief in necessary:

$$K_{V_E} \Rightarrow V_E/* \tag{4}$$

By using (4), together with (2) and (3), and by using the transitivity property of $\Rightarrow$, and the extended hand-off axiom, L will be able to infer $ID(E) \Rightarrow V_E/Tr/E$, that is, that $ID(E)$ corresponds to the entity E, which is located in the organization V_E, under the certification authority Tr.

In the general case, if L is to be able to authenticate every entity belonging to the organizations $O_1 \ldots O_n$, then it has to know (believe) that $K_{O_i} \Rightarrow O_i/*$ for every $i = 1 \ldots n$. A straightforward approach is for L to build a repository of these root certificates. Since this repository will contain an entry for every organization, this approach in not scalable and is likely to be subject to problems of maintenance. As an example, every entry needs to be updated whenever a change or revocation of an organization root key occurs. Moreover, as new organizations need to be authenticated, their corresponding entries will have to be inserted in the repository.

The usual approach to circumvent this problem is to use "external" certification authorities, that are not part of the organizations, but are nevertheless trusted to certify the organizations' roots certification authorities. If, for instance, L trusts the certification authority CA to perform the certification of different organizations, then it can put in its repository of root certificates just the entry

$$K_{CA} \Rightarrow * \tag{5}$$

In this way, when L receives the credentials of the entity E belonging to the organization V_E, which is certified by CA

$$\begin{array}{l} K_{CA} \textbf{ says } K_{V_E} \Rightarrow V_E/* \\ K_{V_E} \textbf{ says } K_{V_E/Tr} \Rightarrow V_E/Tr/* \\ K_{V_E/Tr} \textbf{ says } ID(E) \Rightarrow V_E/Tr/E \end{array} \tag{6}$$

then, by using (5) it can understand

$$ID(E) \Rightarrow V_E/Tr/E$$

which is what it needs. Due to its critical function in the authentication of remote entities, we call CA a *root of trust for authentication* for L.

Summing up, the credentials and the beliefs that are necessary to L in order to authenticate a remote entity E are represented by expressions (6) and (5), respectively.

[1]In order to keep the notation intuitive, we will use the expression $V_E/Tr/*$ in place of the compound principal V_E/Tr **except** ..

IV. Authenticating a Set of Remote Layers

In the previous section we have analyzed the aspects related to the authentication of a layer in a remote platform. In this section we will extend the analysis to cover the whole remote platform stack, from the hardware up to the application.

In order for the local platform to authenticate the remote one, the former has firstly to authenticate every layer making up the latter, and secondly to assess that all the layers are actually part of the same software/hardware stack.

Let us focus on the first step, the authentication of the layers of the remote platform R. For simplicity, we assume that the vendors of the TPM, the OS, and the application ($\mathsf{V_{TPM}}$, $\mathsf{V_{OS}}$, and $\mathsf{V_A}$, respectively), are all structured in a 2-layered PKI as described in the previous section. In this case, we can obtain the credentials that are necessary to authenticate the three layers by applying the same reasoning of the previous section, and considering the remote entity as TPM, OS, and A, respectively:

$$\begin{array}{l} K_{CA} \textbf{ says } K_{V_E} \Rightarrow V_E/* \\ K_{V_E} \textbf{ says } K_{V_E/Tr} \Rightarrow V_E/Tr/* \\ K_{V_E/Tr} \textbf{ says } ID(E) \Rightarrow V_E/Tr/E \\ \texttt{for } E = \{TPM, OS, A\} \end{array} \tag{7}$$

The second step consists in assessing that all the authenticated layers are actually part of the same software/hardware stack. This process requires building a certificate chain from the trusted hardware all the way up to the application. To this end, each software layer (OS and A in this case) that wants to be certified generates a key pair and makes an *endorse* API call to the lower layer, passing, as parameters its public key and other data it wants to be certified [4]. The certificate that is generated through this process contains an ID which uniquely identifies the software layer making the endorse call. This ID can be the hash of the program image corresponding to the software layer, and, possibly, of its configuration parameters. Using K_{TPM} to refer to the key pair embedded in the TPM, K_{OS} for the temporary key generated by the operating system, and ID_{OS} for the code ID of the OS, the certificate generated by the endorse call of the OS is:

$$K_{TPM} \textbf{ says } K_{OS} \Rightarrow K_{TPM}|ID_{OS} \tag{8}$$

With this, TPM certifies that the key K_{OS} has the authority to speak for the trusted platform module running the operating system whose ID is ID_{OS}. Likewise, the certificate generated by the OS upon the endorse call of the application is:

$$K_{OS} \textbf{ says } K_A \Rightarrow K_{OS}|ID_A \tag{9}$$

By using (1) in (8) and its homologous ($K_{OS} \Rightarrow K_{OS}|ID_A$) in (9), we get to:

$$\begin{array}{l} K_{OS} \Rightarrow K_{TPM}|ID_{OS} \\ K_A \Rightarrow K_{OS}|ID_A \end{array}$$

and, by replacing K_{OS} in the second statement with the right side of the first statement, for the monotonicity of operator $|$, we get to

$$K_A \Rightarrow K_{TPM}|ID_{OS}|ID_A \tag{10}$$

In other words, every platform L receiving the certificates (8) and (9) can conclude that the key K_A speaks for a layer identified as ID_A that runs on top of a layer ID_{OS}, that, in turn, runs on top of a layer identified by the key K_{TPM}.

In order for L to identify these IDs as the application A, the operating system OS, and the trusted platform module TPM, respectively, L needs to receive all the certificates that authenticate each one of these layers, as explained at the beginning of this section. By using these certificates, together with its belief on the roots of trust, L can infer:

$$\begin{array}{l} ID(A) \Rightarrow V_A/Tr/A \\ ID(OS) \Rightarrow V_{OS}/Tr/OS \\ K_{TPM} \Rightarrow V_{TPM}/Tr/TPM \end{array} \tag{11}$$

Using these beliefs, together with the belief (10), the local platform is finally able to infer:

$$K_A \Rightarrow V_{TPM}/Tr/TPM \mid V_{OS}/Tr/OS \mid V_A/Tr/A \tag{12}$$

which means, in prose, that the key K_A is the temporary key of an instance of the application A, certified as trusted by its vendor $\mathsf{V_A}$. It is running on top of the operating system OS, certified as trusted by its vendor $\mathsf{V_{OS}}$. OS is running on a trusted platform TPM certified as trusted by its manufacturer $\mathsf{V_{TPM}}$.

Summing up, what it necessary to a local platform in order to authenticate an application running on a remote platform is its belief in one ore more certification authorities that authenticate the roots of the vendors' PKI (5), the set of certificates that make up the PKI of the vendors (7), and the set of endorse calls (8) and (9).

V. Assessing the Safety of a Remote Platform

In the previous section we have found what credentials and beliefs are necessary in order for a local platform L to authenticate the environment of a remote platform R, where an application is running. However, being able to authenticate the stack of a remote platform, does not answer an important question: is that configuration safe for the local platform? Answering this question means evaluating the safety of the remote platform.

The attestation process is meaningful to L if it can get to believe that:

$$\mathsf{V_{TPM}/Tr/TPM} \mid \mathsf{V_{OS}/Tr/OS} \mid \mathsf{V_A/Tr/A} \Rightarrow \mathsf{Tr_C} \quad (13)$$

where $\mathsf{Tr_C}$ is the set of configurations considered safe for the remote application, according to the actions that the local one wants to perform. In fact, by using this, together with (12), by the transitivity property of $\Rightarrow$, the application can infer $\mathsf{K_A} \Rightarrow \mathsf{Tr_C}$. This means, in prose, that the key $\mathsf{K_A}$ belongs to an instance of an application which is part of a platform in a known trusted state.

There are two issues related to (13). Firstly, since it is a belief, it needs to be derived from a trusted source in the form, as an example, of a certificate. Secondly, in order for a local application to be prepared to evaluate the safety of a random remote application, the former needs to receive evidence as in (13) for every possible remote platform configuration.

This second issue can be kept simpler by grouping TPMs, operating systems, and applications in trusted and untrusted, and then considering a remote platform safe if its stack is made up of trusted entities. The information related to grouping can be distributed by independent security organizations or by security companies, in the form of certificates. Alternatively, it can be entered manually into the local platform by its system administrator. As an example, the local system administrator could enter the following beliefs into the local platform:

$$\begin{array}{l} \mathsf{V_{TPM}/Tr/TPM} \Rightarrow \mathsf{Tr_{TPM}} \\ \mathsf{V_{OS}/Tr/OS} \Rightarrow \mathsf{Tr_{OS}} \\ \mathsf{V_A/Tr/A} \Rightarrow \mathsf{Tr_A} \\ \mathsf{Tr_{TPM}|Tr_{OS}|Tr_A} \Rightarrow \mathsf{Tr_C} \end{array} \quad (14)$$

With these, the local platform believes that TPM, certified by Tr, and, in turn, certified by $\mathsf{V_{TPM}}$ belongs to the group $\mathsf{Tr_{TPM}}$. By using the same reasoning, it believes that OS and A belong to the group $\mathsf{Tr_{OS}}$ and $\mathsf{Tr_A}$, respectively. The last belief states that if a member of the group $\mathsf{Tr_A}$ runs on top of a member of the group $\mathsf{Tr_{OS}}$, which runs on top of a member of the group $\mathsf{Tr_{TPM}}$, then the configuration is member of the group $\mathsf{Tr_C}$. Therefore, if the local application knows that $\mathsf{Tr_C}$ is a group of safe configurations for the remote application, then by using the beliefs in (14), together with (12), it can infer $\mathsf{K_A} \Rightarrow \mathsf{Tr_C}$, which is exactly what it needs. A more relaxed security profile could trust some authorities to certify which entities belong to which groups, as follows:

$$\begin{array}{l} \mathsf{CA_{TPM}} \Rightarrow \mathsf{Tr_{TPM}} \\ \mathsf{CA_{OS}} \Rightarrow \mathsf{Tr_{OS}} \\ \mathsf{CA_A} \Rightarrow \mathsf{Tr_A} \\ \mathsf{Tr_{TPM}|Tr_{OS}|Tr_A} \Rightarrow \mathsf{Tr_C} \end{array} \quad (15)$$

With these beliefs, the authority $\mathsf{CA_{TPM}}$, $\mathsf{CA_{OS}}$, $\mathsf{CA_A}$ will be trusted to assess the safety of TPMs, OSs, and applications, respectively. Due to their critical role in the evaluation of a remote platform, we call the beliefs in (14) or in (15) the *roots of trust for evaluation* for L.

The credentials that L needs in order to evaluate the safety of R are dependent on its roots of trust for evaluation. As an example, by using the roots of trust in (15), the credentials that L needs to receive to evaluate R (to understand that $\mathsf{K_A} \Rightarrow \mathsf{Tr_C}$) are:

$$\begin{array}{l} \mathsf{K_{CA_{TPM}}}\ \textbf{says}\ \mathsf{V_{TPM}/Tr/TPM} \Rightarrow \mathsf{Tr_{TPM}} \\ \mathsf{K_{CA_{OS}}}\ \textbf{says}\ \mathsf{V_{OS}/Tr/OS} \Rightarrow \mathsf{Tr_{OS}} \\ \mathsf{K_{CA_A}}\ \textbf{says}\ \mathsf{V_A/Tr/A} \Rightarrow \mathsf{Tr_A} \\ \mathsf{K_{CA}}\ \textbf{says}\ \mathsf{K_{CA_i}} \Rightarrow \mathsf{CA_i}, \mathsf{for}\ \mathsf{i} = \{\mathsf{TPM}, \mathsf{OS}, \mathsf{A}\} \end{array} \quad (16)$$

Notice that every combination of roots of trust for evaluation and credentials can be used, as long as they allow to determine $\mathsf{K_A} \Rightarrow \mathsf{Tr_C}$.

Summing up, what is necessary to a local platform in order to evaluate the safety of a remote platform, is its roots of trust for evaluation, i.e. trusted statements that provide information either on which platforms to consider safe (14), or on authorities to trust for the evaluation (15). In the latter case, the local platform needs also a set of credentials produced by these authorities that complement the beliefs allowing to evaluate the platform (16).

VI. Conclusions

Remote attestation is a new functionality offered by trusted platform computing, which may be useful to a local platform in order to extend trust to a remote one.

In this paper we have analyzed which credentials and beliefs are necessary in order for this process to be successful and meaningful. As part of this, we have highlighted how this process relies on two sets of key-beliefs, which we have called, respectively, the roots of trust for authentication and for evaluation.

Acknowledgments

This work has been supported by grants from the *Natural Sciences and Engineering Research Council of Canada* (NSERC), from the Canadian network of centres of excellence *Mathematics of Information Technology and Complex Systems* (MITAC), and from the *Commission of the European Communities* under the sixth framework programme project IST-004536 "Reconfigurable Ubiquitous Networked Embedded Systems" (RUNES).

References

[1] M. Abadi and T. Wobber. A logical account of NGSCB. In *Proceeding of the Formal Techniques for Networked and Distributed Systems - FORTE 2004*, volume 3235, pages 1–12. Springer-Verlag, September 2004.

[2] A. Bottoni. *Security and trust in multi-domain distributed systems*. PhD thesis, University of Pisa, 2006. In preparation.

[3] P. England, B. Lampson, J. Manferdelli, M. Peinado, and B. Willman. A trusted open platform. *IEEE Computer*, 36(7):55–63, 2003.

[4] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 193–206. ACM, Oct. 2003.

[5] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: theory and practice. *ACM Transactions on Computer Systems (TOCS)*, 10(4):265–310, 1992.

[6] Trusted computing group. http://www.trustedcomputinggroup.org.

[7] T. Wobber, M. Abadi, M. Burrows, and B. Lampson. Authentication in the Taos operating system. *ACM Transaction in Computer Systems*, 12(1):3–32, 1994.