Trust assessment of X.509 certificate based on certificate authority trustworthiness and its certificate policy

Zakia El Uahhabi* and Hanan El Bakkali

Information Security Research Team, ENSIAS-Mohammed V University, Rabat, Morocco
Email: zakia.eluahhabi@um5s.net.ma
Email: h.elbakkali@um5s.net.ma
*Corresponding author

Abstract: Nowadays, X.509 certificate is largely used to prove its holder identity in open networks. Then, the relying party (RP) needs an automated mechanism for evaluating its trustworthiness in order to decide whether to accept it or not. In this context, we provide him with this mechanism allowing him to decide if he should trust in a received certificate or not. In our previous work, we have proposed an architecture for calculating a certificate trust level. Using a defined algorithm, this level is computed depending on three parameters: the calculated trust level of certificate authority (CA), the certificate policy quality, and the rating of the certificate fields. In this paper, we improve the algorithm used to calculate a CA trust level on the basis of trust level of the CAs that had issued certificates for it and their extension fields. By this way, the calculated trust level reflects a real trustworthiness of certificate because it is computed on the basis of the real factors influencing this trustworthiness. It is then more relevant for a relying party when deciding whether to accept a received certificate or not.

Keywords: certificate authority; reputation score; trust level; X.509 certificate; public key infrastructure; PKI; certificate policy.


Biographical notes: Zakia El Uahhabi is a researcher and PhD student in the field of Computer Sciences and Information Technology from the ENSIAS, Mohammed V University, Rabat, Morocco. She obtained her Master’s in the field of network and systems from the ENSAT, Abdelmalik Saadi University, Tangier, Morocco in 2010. Her current research interest is in the field of trust management in public key infrastructure (PKI).

Hanan El Bakkali is an Associate Professor and Head of the Information Systems Security Option from the ENSIAS. She is a member of Information Security Research Team (ISERT). Her research interests are in electronic transactions security, PKI trust models, reputation systems, and access control models in workflow management systems.
1 Introduction

Public key infrastructure (PKI) is responsible for managing the certificates. It provides mechanisms for distributing, storing, generating, and revoking these certificates. They are signed by a certificate authority (CA) in order to prove that a public key is associated with a specific entity.

X.509 certificate is widely used to confirm the identity of its holder in open networks. It is generated by CA using the procedures which are described in the certificate policy (CP). Besides, the relying party (RP) may accept a received certificate if the issuer CA belongs to its trust domain. This does not reflect a current status of CA which may be compromised and used to issue fraudulent certificates for its certificate holder (CH). The attackers exploit any security breaches of CA’s system in order to obtain valid certificates. Then, RP cannot be sure about the trustworthiness of CAs in the near future. Also, their competency is not the same; it differs from one to another. In general, they can be evaluated by regular audits. This may not reflect their current status because these audits are not continuous and are executed during a period of time.

The RPs task is complex. He must decide whether to accept a received certificate or not. Principally, RPs belief must be based on the reading of the CP document. In reality, most of them will not be able read this document since it is long and technical. Also, it is difficult for RPs who has not the technical competences to judge it. In addition, any deficiency in the procedures defined in CP, which are followed by an issuer CA, may create a trust lack in the certificate that is issued by this authority. As a result, RPs need an automated mechanism to evaluate certificate trustworthiness.

In our previous papers (El Uahhabi and El Bakkali, 2016a), we provide him with this mechanism. We have proposed an architecture for calculating a certificate trust level using our suggested algorithm. In this paper, we present an improvement in CA trust level calculation algorithm. We enhance its computation process when getting a low number of CHs that leave their appreciation for a specific CA. Then, we take into account the certificates obtained by the authority that we evaluate its trustworthiness. According to (El Bakkali and Idrissi, 2001), issuing a certificate by a trusted CA for another CA implicates that the issuer authority places certain trust in it. In this context, we will evaluate this implicit trust by analysing the issued certificate extension fields. The new trust calculation process for a CA will be added to the previous one.

The rest of this paper is structured as follows. Section 2 presents a background on X.509 certificate followed by related work in Section 3. Section 4 describes our suggested trust framework architecture. In Section 5, we define our new proposed algorithm for calculating a CA trust level that is used for computing certificate trust level. The last section concludes our paper with future work.
2 Background on X.509 certificate

Before describing our approach, it is important to provide a brief background on X.509 certificate. In fact, X.509 certificate is a digital certificate that is used for binding a public key to its owner that can be an individual, server, organisation, or other entity (Henniger et al., 2006; Ahamad et al., 2016; Sadqi et al., 2015). It is signed by a trustworthy third party called a certification authority (CA) using its private key.

There are three versions about public key certificate, which are an X.509.v1, X.509.v2, and X.509.v3, as shown in Figure 1 (Jøsang et al., 2000). The first version has been developed into X.509v2 and X.509v3 in order to overcome weaknesses in the earlier versions (Jøsang et al., 2000). Version 2 contains all fields of version 1 and two others additional which are subject and issuer unique identifier, but it is not largely deployed in open internet.

Then, the third version comes with several extension fields, as described in Figure 1, in order to be used in a wide range of applications and environments. Therefore, version 3 of the X.509 certificate is widely used today. Generally, the certificate structure is described using ASN.1 format (Cooper et al., 2008). We present some of the important fields.

Figure 1  X.509 certificate with extension fields

2.1 Basic certificate fields

The certificate is structured as a sequence of three basic fields as shown in Figure 2:
Figure 2  Presentation of X.509 certificate in ASN

Certificate ::= SEQUENCE {
    tbsCertificate   TBSCertificate,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValue   BIT STRING }

TBSCertificate ::= SEQUENCE {
    version [0] EXPLICIT Version DEFAULT v1,
    serialNumber   CertificateSerialNumber,
    signature   AlgorithmIdentifier,
    issuer   Name,
    validity   Validity,
    subject   Name,
    subjectPublicKeyInfo   SubjectPublicKeyInfo,
    issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL
        -- If present, version MUST be v2 or v3
    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL
        -- If present, version MUST be v2 or v3
    extensions [3] EXPLICIT Extensions OPTIONAL
        -- If present, version MUST be v3
}

These fields are defined as follows (Cooper et al., 2008):

- **tbsCertificate**: it includes the issuer CA and subject names, a validity period, a public key related with the subject, and other information. This information is defined in detail in (Cooper et al., 2008). This field usually contains extensions, which are described in next subsection.

- **signatureAlgorithm**: this field includes the identifier of the cryptographic algorithm used for signing a certificate by the CA.

- **signatureValue**: it includes a digital signature of tbsCertificate.

### 2.2 Certificate extensions

X.509 v3 certificate includes a sequence of extension fields that provide some additional associations with users or public keys, and methods for managing relationships between CAs (Cooper et al., 2008). Each extension is marked as non-critical or critical. Critical extension must be processed, whereas non-critical one may be ignored by relying parties if they are not recognised (Cooper et al., 2008). At a minimum, applications conforming to X509v3 have to know the following extensions: certificate policies, basic constraints, policy constraints, key usage, name constraints, subject alternative name, inhibit anyPolicy, and extended key usage (Cooper et al., 2008). There are five extensions which are important for evaluating and measuring a certificate trust. These extensions are basic constraints, key usage, policy constraints, policy mappings, and certificate policies. We explain them and present their syntax according to RFC 5280 (Cooper et al., 2008):
• Basic constraints: this field indicates that whether the certificate subject is a CA and limits the certification path depth which includes this certificate. It must be critical for all certificates issued to CAs.

id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }
BasicConstraints ::= SEQUENCE {
cA BOOLEAN DEFAULT FALSE,
pathLenConstraint INTEGER (0..MAX) OPTIONAL }

• Key usage: it specifies the purpose for which a public key, included in the certificate, may be used.

id-ce-keyUsage OBJECT IDENTIFIER ::= { id-ce 15 }
KeyUsage ::= BIT STRING {
digitalSignature (0),
nonRepudiation (1), -- recent editions of X.509 have
-- renamed this bit to contentCommitment
keyEncipherment (2),
dataEncipherment (3),
keyAgreement (4),
keyCertSign (5),
cRLSign (6),
encipherOnly (7),
decipherOnly (8) }

• Certificate policies: This field specifies one or more policies followed by the CA for creating a subject certificate. Each policy is identified by an object identifier (OID) optional qualifiers. In a CA certificate, this information limits the set of policies for certification paths that include this certificate (Cooper et al., 2008).

• Policy mappings: It presents the list of policies associated with the subject CA that may be accepted as comparable to the issuerDomainPolicy (Cooper et al., 2008). This means the issuer CA considers that its issuerDomainPolicy is equivalent to the subjectDomainPolicy.

id-ce-policyMappings OBJECT IDENTIFIER ::= { id-ce 33 }
PolicyMappings ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
is issuerDomainPolicy CertPolicyId,
s subjectDomainPolicy CertPolicyId }

• Policy constraints: This extension is useful for controlling the CP processing through a trust chain. It allows a CA to prohibit policy mapping for child CAs (inhibitPolicyMapping) or to require explicit policies for the certification path remainder (requireExplicitPolicy).
3 Related works

There are several approaches that have been proposed for a trust assessment in PKI. Among these approaches, we highlight the following which are shown in tabular format. We explain them in Table 1, presenting their advantages and their gaps.

Our work consists of overcoming some gaps of previous works. For instance, the authors in (Wazan et al., 2017) determine a quality of a certificate on the basis of the evaluation of the procedures announced in CP/certificate practice statement (CPS) and a CA commitment to this policy. CA trustworthiness is assessed by measuring its commitment to this document based only on its reputation, which depends on the recommendations from the clients. However, the authors indicate that checking a given recommendation is limited only to the unidentified entities. That could lead to serious risks in issuing false recommendations by the known entities because they may be attacked. In our point of view, the determination of reputation is an important factor for evaluating CA trustworthiness but it is not sufficient. In our approach, CA trustworthiness is determined on the basis of different factors such as a CA security risk, its reputation, and evaluation of trust propagation along the built path from an issuer CA to subject CA in question. We note that the factors that impact on CA trustworthiness have an influence on certificate trustworthiness. Nevertheless, assessing a trustworthiness of CA is not sufficient for determining that of certificate. We need to know also the quality of the procedures followed by a CA for issuing this certificate and described in CP/CPS, and analyse its extension fields. In our work, we try to combine all the factors influencing certificate trustworthiness for evaluating it and obtaining an accurate result reflecting its real trustworthiness. At our best knowledge, there is no approach that integrates all these factors in the certificate trust evaluation process. We will provide more details about our approach in next section.

4 Proposed solution

4.1 Motivation and design goal

As presented in our previous work (El Uahhabi and El Bakkali, 2016a), we have proposed an approach for evaluating certificate trustworthiness in order to help RPs in making decisions about a received certificate. We have calculated its trust level depending on different parameters such as CA trust level (TLoCA). This TLoCA value is computed on the basis of CA reputation score, and its security maturity level.
<table>
<thead>
<tr>
<th>Proposed approach</th>
<th>Explanation</th>
<th>Advantages</th>
<th>Gaps identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix powers algorithms for trust evaluation in public-key infrastructures</td>
<td>Solution used for evaluating trust in PKI. It defines trust relationships between entities in a network as triplet (trust, distrust, uncertainty). The proposed algorithm can be used for generic graph.</td>
<td>It takes into account all existing trust paths between entities as a preliminary to any exchange between PKIs (Dumas et al., 2013). This can provide a precise trust evaluation.</td>
<td>Taking into consideration all existing trust paths between entities for evaluating trust in PKI may require a long time.</td>
</tr>
<tr>
<td>(Dumas and Hossayni, 2013)</td>
<td></td>
<td>• Solving an issue related to the authentication of partners that their identity and attributes are provided by CAs located in different aeronautical trust domains.</td>
<td>• Certificate size may be large, and each local CA must verify all its certificates for allowing a revocation service.</td>
</tr>
<tr>
<td>X.509 identity certificates with local verification (Bauer, 2012)</td>
<td>Each CA located in different aeronautical trust domains extends the certificate by adding properties for which it is authoritative. Finally, the local authority must verify the validity of the assigned properties included in the final extended certificate and signs it.</td>
<td>• Permitting to verify the certificate without establishing a certification path to a global trust anchor.</td>
<td>• Security and trustworthiness problem of the local CA are not addressed in this approach; attacking this authority could lead to the risk of trusting a malicious local CA and then a malicious certificate.</td>
</tr>
<tr>
<td>Rootopia: tool for defending against CA failures in the Web PKI (Braun and Rynkowski, 2013)</td>
<td>Proposed solution is used to identify and assess the set of relevant CAs for an individual user on the basis of his browser history.</td>
<td>Limiting the trusted CAs.</td>
<td>User may not store his history data. In this case, there is no history data is available, and it is not possible to predict his view about required CA.</td>
</tr>
<tr>
<td>X.509 check: a tool to check the safety and security of digital certificates</td>
<td>Proposed tool takes into consideration 30 security features. The authors assign to each feature a numerical weight value on the basis of its importance in evaluating a certificate security. According to this weight value, a certificate is categorised into three kinds: insecure, risk, and secure.</td>
<td>The proposed tool may be used by auditors and administrators without the need to be an expert.</td>
<td>Evaluated certificate may be fraudulent. It is necessary to assess a CA system security which is not addressed in this proposition.</td>
</tr>
<tr>
<td>(Alrawais et al., 2015)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trust management for public key infrastructures: implementing the X.509 trust broker (Wazan et al., 2017)</td>
<td>New entity called trust broker (TB) is added to the original X.509 trust model. It plays the role of both technical and legal expert. It evaluates a CA and then calculates the certificate level of assurance (CLoA).</td>
<td>Helping RP to make an informed decision about a received certificate in open environment.</td>
<td>• Verifying a provided recommendation is limited only to the unidentified entities. That could lead to serious risks in issuing false recommendations by the known entities because these entities may be attacked.</td>
</tr>
<tr>
<td>Proposed approach</td>
<td>Explanation</td>
<td>Advantages</td>
<td>Gaps identification</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Trust management for public key infrastructures: implementing the X.509 trust</td>
<td>A new entry called trust broker (TB) is added to the original X.509 trust model. It plays the role of both technical and legal expert. It evaluates a CA and then calculates the certificate level of assurance (CLoA).</td>
<td>Helping RP to make an informed decision about a received certificate in open environment.</td>
<td>• Verifying a CA commitment to its published CP depends only to the audit agency recommendation when any recommender has any experience with this authority. That may not reflect the current status of CAs and not provide a precise trust evaluation.</td>
</tr>
<tr>
<td>broker (Wazan et al., 2017)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation of trust metrics in X.509 public key infrastructure (Gonc et al.,</td>
<td>This approach presents three kinds of trust in the X.509 PKI: policy trust, authentication trust, and PKI trust. Authors specify ASN.1 structure to define the calculated trust values for each kind of trust, and include them in X.509 certificate.</td>
<td>Giving users a new tool to verify a certificate trust level.</td>
<td>Certificate that includes a higher trust level may be attacked. That is not addressed in this proposition.</td>
</tr>
<tr>
<td>2013)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A framework for calculating a risk associated with X.509 certificates by evaluating</td>
<td>Proposing an algorithm for evaluating CPS document which locates certain attributes in this document. Categorising risk in three levels – high, medium and low risk.</td>
<td>Giving an appropriate risk level to user in order to prevent him from getting harm.</td>
<td>Evaluating a risk associated with X.509 certificates on the basis only on CPS is not sufficient for obtaining an accurate result.</td>
</tr>
<tr>
<td>certificate practice statement (CPS) (Hawanna et al., 2016)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advising clients of the trust level of CA by assessing its certificates (Anooshiravan</td>
<td>Defining a model for evaluating a certificate. In this model, a certificate authority trust service collects a set of certificates that have been submitted by users and evaluates them according to the rules based on different factors.</td>
<td>Generating a CA trust level which is distributed to users for helping them to make decision about such authority.</td>
<td>• The collected certificates may be submitted by malicious users, which are not addressed in this approach. In this case, it cannot judge the CA based on these certificates.</td>
</tr>
<tr>
<td>et al., 2016)</td>
<td></td>
<td></td>
<td>• Collection of certificates, which are submitted by users, may require a long time. That has an influence on determining a CA trust level.</td>
</tr>
</tbody>
</table>
Limitation of our approach in (El Uahhabi and El Bakkali, 2016a, 2016b) is presented from the fact that many CHs may not provide their appreciation for a specific CA. Then, we may get a low number of ratings or not get any rating from them. In such cases, it is not possible to obtain a precise trust evaluation for CA. Consequently, it is necessary to determine other factor or mechanism for improving the TLoCA calculation process. In fact, our suggested solution is that evaluating trust propagation along the constructed path from an issuer CA to subject CA in question. Our solution extends our previous approach (El Uahhabi and El Bakkali, 2016a) by adding a new trust assessment method for calculating a CA trust level, and aims to improve it. According to (El Bakkali and Idrissi, 2001), giving a certificate by a trusted CA for another one indicates that the issuer authority places certain trust in it. Then, an issuer CA should not issue a certificate if it does not trust a subject CA. In this paper, we will evaluate this implicit trust by analysing the issued certificate extensions fields. We note that these fields play an important role for managing relationships between CAs. In this sense, we assess a CA trustworthiness based on their certificates issued by the other trusted CAs. Our main concern is to provide an efficient tool which helps RPs to make decision about a received certificate.

4.2 Suggested trust framework architecture

Firstly, we define the main actors involved for evaluating trust in X.509 certificate. There are four entities: the RP, the CA, the CH and our system TrustCert. The sequence diagram (Figure 3) presents the interaction between these actors. When CH receives a certificate from a CA, he issues it to RP for accessing a service. In this case, the task of RP is huge and complex. He must decide whether accept a received certificate or not. Then, he issues it to our system for knowing its trust level value (TLoCERT) and helping him in making decision. The role of our suggested system is that evaluating the trustworthiness of the issuer CA, and that of the received certificate. For this reason, it is necessary to extract some information, as illustrated in Figure 3, from extern actors such as CA, other CHs, and sources, in order to compute the parameters used for calculating a TLoCERT value of certificate in question. The calculation process is explained in details in Figure 4 which defines the architecture of our proposed system.

The proposed architecture describes several steps for computing a TLoCERT. Noting this, we suppose that our suggested system is secure against an attacker that may forcefully inject satisfied value of trustworthiness and then sends a high trust level of a certificate to RP. In this paper, we do not consider security problems regarding an attacker ability to violate our proposed system and issue fraudulent trust level. In our work, we take into account evaluating security level of CA system. The policies, procedures, and controls measures against any malicious behaviour of an attacker such as trying to access the database, and injecting false information, will be enforced during deployment and operation of our system in a secure manner. Also, we will use a secure development for deploying it. We will give more details about a security of our system in a future paper when implementing a realistic proof of concept of the proposed framework TrustCert.

On the other hand, in our suggested architecture, the CHs provide their ratings about a CA, which has issued them a certificate, by accessing our framework through a link embedded into e-services applications. As mentioned previously, some CHs may be not interested in giving their appreciations about a specific CA. In this case, we focus on
improving a trust evaluation process of a CA based on their certificates issued by the other trusted CAs. In this sense, we add to our suggested architecture, which is presented in a previous paper (El Uahhabi and El Bakkali, 2016a), a new module called extractor module, as illustrated in Figure 4. Once RP accesses our system via a RP interface and sends a certificate that he wants to know its trust level, the TLoCERT computation process is started. Then, a received certificate is exploited by our architecture using several modules involved for calculating the TLoCERT. We explain these modules:

- **Decoder module:** It permits to convert a received X.509v3 certificate into XML format which is issued to the trust module.

- **Trust module:** It includes the trust calculator. It analyses a received certificate in XML using a parser DOM and extracts the needed information for computing a TLoCERT. This information is used for sending a request by the calculator to four modules; extractor module, reputation module, CPQ module, and SL evaluator asking them respectively a CA certificates, RepScore, CPQ value, and SL value. The two parameters RepScore and SL are used by the calculator for computing a CA trust level TLoCA. Also, the obtained CA certificates are analysed by a trust module for calculating a propagated trust level (PTL). This parameter is added to other parameters for calculating TLoCA. Moreover, this module calculates then a TLoCERT based on three parameters that are a calculated TLoCA, CPQ value, and a rating value assigned to certificate fields’ content.

- **Reputation module:** Once CHs provide via a CH interface their ratings about a CA that has issued them a certificate, they are stocked in feedback DB. These ratings are used by a reputation module for computing or updating the RepScore value, and then archived in feedback archive in order to release a feedback database space. We note that a reputation calculator retrieves the received ratings for computing a RepScore value if its number reaches a 10% of the calculated threshold that will be explained in next section.

- **Security level module:** This module includes two components: CA-SL Evaluator and SL DB. The evaluator component assesses a CA security level (SLoCA) on the basis of the retrieved information from the SL DB. This information is a security standard implemented by a CA and EAL of the software used by this authority.

- **CPQ module:** In this module, there are two components: CPQ database (CPQ DB) and CPQ calculator. CPQ value is computed by a calculator in applying an algorithm that takes a CP in XML format as input.

- **Translator module:** It permits to translate a CP document in XML format in order to use it for calculating a CPQ value. The CP translator retrieves from a repository a requested CP document and converts it into XML format. Then, the XML CP file is sent to CPQ calculator in order to use it for calculating a CPQ value. The translation process will be described in a future paper.

On the other hand, the proposed architecture uses different algorithms for calculating a TLoCERT value. These algorithms are described in details in the previous papers (El Uahhabi and El Bakkali, 2016a, 2016b, 2016c). In next subsection, we define in details a new algorithm for calculating a PTL.
Figure 3  Interaction sequence between the main actors involved for evaluating trust in X.509 certificate (see online version for colours)
4.3 Proposed calculation method of TLoCERT

To calculate a certificate trust level, it is needed to determine the factors that have an influence on its trustworthiness. As discussed in our previous paper (El Uahhabi and El Bakkali, 2016a), evaluating trust in X.509 certificate depends on the CA trustworthiness, the procedures which are announced in the CP/CPS, and certificate extension fields’ content. Also, there are different factors that influence CA trustworthiness, and then a certificate trust assessment such as CA reputation, and its security risks. In this work, we define other factor which permits to evaluate CA trustworthiness. It is a trust propagation evaluation that is determined from the certificates issued by trusted CAs for a CA in question.

Moreover, we use the weighted average method for calculating TLoCERT value on the basis of three parameters that are the TLoCA, CPQ, and RoCERT. Then, we suggest the following formula:

\[
T\text{LoCERT} = \frac{\sum_{i=1}^{3} w_i \cdot \text{RoCERT} + w_2 \cdot \text{TLoCA} + w_3 \cdot \text{CPQ}}{\sum_{i=1}^{3} w_i}
\]

(1)

where

\[
a \quad \text{CPQ} \in [0, 1]
\]

represents a quality of the procedures followed by CA and announced in its CP document.
b TLoCA ∈ [0, 1] represents a trust level accorded to CA for issuing a correct certificate.

c RoCERT ∈ [0, 1] represents the rating assigned to the certificate fields’ content (validity and key usage fields).

d w1 = 0.2, w2 = 0.4, and w3 = 0.4 are the assigned weights and \( \sum wi \) equals to 1.

These weights present the impact of these parameters on the TLoCERT computation. The two parameters TLoCA and CPQ have a more impact on the determination of certificate trustworthiness than RoCERT. For this reason, their weight value is greater than that of RoCERT. Moreover, these weights may be determined by RP through a RP interface when he asks for TLoCERT value of the received certificate. Then, we propose him to define the weights values where \( \sum wi \) equals to 1, which he thinks they are suitable for assign them to the parameters used for computing a TLoCERT value, or select default weights which we have defined previously. The main reason for this option is that we permit RPs to participate in calculation process and define their preferences. Besides, the aforementioned parameters are taken as input of the TLoCERT computation algorithm illustrated in Algorithm 1 as follows:

**Algorithm 1** Basic algorithm

---

**Input**: XML FILE as an XML Dom tree D

idCA id of a CA  
RepScore is a reputation score of a CA  
CPQ is a certificate policy quality value  
RoCERT is assigned rating to certificate  
SLoCA is a security maturity level  
PTL is a propagated trust level  
\( n \) is a threshold value  
\( m \) CHs number that leave their rating

**Output**: TLoCERT: trust level of a received certificate

1 Define the following variable:  
TLoCA: calculated trust level of a CA  
// Call a function trustlevelCA for getting a CA trust level

2 \( TLoCA = \text{trustlevelCA}(idCA, \text{RepScore}, SLoCA, PTL, m, n) \)

3 \( TLoCERT = 0.2 \times \text{RoCERT} + 0.4 \times TLoCA + 0.4 \times CPQ \)

4 Return TLoCERT

5 End

The calculation methods of these aforementioned parameters will be explained in details in next subsections. On the other hand, the calculated TLoCERT belongs to [0, 1]. There are four trust levels: high trust, medium trust, low trust, and no trust. Then, we divide the interval [0, 1] into four subintervals of equal length. Four levels are enough to quantify the trust levels. We could add another level but not necessary. Our idea is that helping a RP to decide whether accept or not a received certificate. If its trust level is medium or
high \((T_L \geq 0.5)\), we send him this level and recommend him to accept it. Otherwise, we advise him to reject it.

As a result, the first level is included in \([0.75, 1]\), whereas the second belongs to \([0.5, 0.75)\). 'No trust' is explained by a level included in \([0, 0.25)\). At last, concerning the 'low trust', its related level belongs to \([0.25, 0.5)\).

**Figure 5** An initiated evaluation of certificate correctness

![Flow Diagram of Certificate Evaluation](image)

### 4.3.1 Calculating the RoCERT

At the beginning, our system assesses certificate correctness by checking the contents of two fields: validity and key usage. We evaluate them by using a rating mechanism. There are three ratings (RoCERT) 1, 0.5, 0. One of them is assigned to the certificate for representing its initial correctness evaluation. Moreover, the assessment process of the certificate correctness is illustrated in the flow diagram on Figure 5. As shown in this
figure, initially, we check the key usage field’ content. If it is equivalent to the application usage requirements, we verify a validity field’ content. If a certificate is valid, we check its validity period. When we get that its validity will be expired soon, the assigned rating is \( \text{RoCERT} = 0.5 \). Otherwise, \( \text{RoCERT} \) takes value 1 which is assigned to the certificate in question. In this case, the evaluation process is continued. On the other hand, we consider that if a key usage field’ content is not equivalent to the application usage requirements and/or its validity period is not respected; such certificate is untrustworthy and must be disallowed. Then, our system will stop the trust evaluation process and the assigned rating is \( \text{RoCERT} = 0 \). In such case, we advise RP that the requested certificate is untrustworthy by sending him an e-mail message.

4.3.2 Calculating the TLoCA

As indicated previously, calculating the TLoCA depends on three parameters PTL, RepScore and SLoCA. The calculation of the PTL and SLoCA depends on the received information about a CA, but the determination of RepScore is based on the feedback given by CHs explaining their appreciation for this authority. Firstly, we compute an initial trust level of CA (TLoCA\(_0\)) based on the real information related to this authority concerning its adopted security measures and its obtained certificates. This information may promote trust in its issued certificate. Then, TLoCA\(_0\) indicates the trustworthiness of this authority’s technology, practices and other relevant characteristics. It depends on PTL and SLoCA. As a result, we calculate a TLoCA using a weighted average method as follows:

\[
\text{TLoCA} = \frac{p_1 \cdot \text{RepScore} + p_2 \cdot \text{TLoCA}_0}{\sum_{i=1}^{2} p_i}
\]

(2)

where

a TLoCA\(_0\) \( \in [0, 1] \) represents an initial trust level which is explained above.

b RepScore \( \in [0, 1] \) represents a score of the CA reputation which is determined from the ratings provided by CHs.

c \( p_1, p_2 \) are the assigned weights and \( \sum p_i \) equals to 1.

These weights present the impact of these parameters on the TLoCA computation. RepScore is updated when receiving new ratings or the RP request about a TLoCA.

TLoCA\(_0\) is calculated using a following formula:

\[
\text{TLoCA}_0 = \alpha_1 \cdot \text{SLoCA} + \alpha_2 \cdot \text{PTL}
\]

(3)

where

a SLoCA \( \in [0, 1] \) represents a security maturity level which evaluates CA system security.

b PTL \( \in [0, 1] \) represents a PTL which is determined from the certificates issued by trusted CAs for CA in question.

c \( \alpha_1, \alpha_2 \) are the assigned weights and \( \alpha_1 + \alpha_2 = 1 \).
TLoCA₀ changes if SLoCA and PTL are updated. In addition, \( \alpha_1 \) and \( \alpha_2 \) represent the weights which are determined depending on the impact of PTL and SLoCA on TLoCA₀ calculation. They may take one of these discrete values 0.5, 0.25, and 0.75 which present respectively medium, low, and high impact. We choose discrete values rather than continuous one because PTL and SLoCA value change over time depending on the received information on which we are based for computing them. The use of a continuous scale makes it difficult to understand the meaning of this change. The weights value should reflect this change and encompass the reality. Then, their adjustment can be determined in two ways as follows:

\[ \alpha_1 = 0.75, \text{ and } \alpha_2 = 0.25 \]

CA, that we want know its trust level, may obtain a low number of certificates which are issued by other CAs for it. Consequently, PTL value is computed on the basis of the analysis of low number of certificates. In this case, we do not obtain an accurate result. For this reason, we assign to PTL a weight value which should be lower than that of SLoCA. Then, the impact of SLoCA is more than that of PTL on CA trustworthiness. As a result, \( \alpha_1 \) value must be greater than \( \alpha_2 \).

\[ \alpha_1 = 0.5, \text{ and } \alpha_2 = 0.5 \]

In this case, we have a high number of certificates which are issued to CA in question, and used for evaluating an implicit trust placed by the issuers CA in this authority. We will discuss this in more details in Section 4.3.2.3. Moreover, getting a high number of these certificates permit us to accurately judge a CA trustworthiness based on their analysis. In this case, \( \alpha_1 \) and \( \alpha_2 \) take a same value 0.5.

As presented in equation (2), the TLoCA is calculated on the basis of two parameters TLoCA₀ and RepScore. The adjustment of the assigned weights \( p_1 \) and \( p_2 \) is determined using a same process that is defined for precising the values of \( \alpha_1 \) and \( \alpha_2 \). They take one of these values 0.5, 0.25, and 0.75. That is described in the following steps, as defined in Algorithm 2:

**Step 1** At initial, we may not obtain any rating from CHs. In this case, the TLoCA is equal to TLoCA₀. Then, we use the following formula for computing a TLoCA value:

\[ TLoCA = TLoCA₀ + \alpha_1 \cdot SLoCA + \alpha_2 \cdot PTL \]

**Step 2** In this step, there are a few CHs that access to our framework and leave their ratings for a specific CA. In this case, a CA reputation must not be judged on the basis of the low number of CHs. Consequently, we assign to RepScore a weight value \( p_1 \) which should be lower than \( p_2 \) of TLoCA₀. It takes a value 0.25.

\[ TLoCA = 0.75 \cdot \text{RepScore} + 0.25 \cdot TLoCA₀ \]

**Step 3** The number of CHs growths and achieves a threshold value which will be defined in next, we increment the weight value of RepScore (\( p_1 = 0.5 \)),

\[ TLoCA = 0.5 \cdot \text{RepScore} + 0.5 \cdot TLoCA₀ \]
Algorithm 2 CA trust level calculation

Function trustlevelCA(idCA, RepScore, SLoCA, PTL, m, n)

Input: idCA id of a CA
   RepScore is a CA reputation score
   SLoCA is a CA security level
   n is a threshold value
   m is CHs number that provide their feedback
   PTL is a propagated trust level
   $\alpha_1$ is a weight assigned to SLoCA
   $\alpha_2$ is a weight assigned to PTL

Output: TLoCA is a CA trust level

1. If Repscore value does not exist then
2. TLoCA ← $\alpha_1*SLoCA + \alpha_2*PTL$
3. Else
4. If (m<n) then
5. TLoCA ← 0.25* RepScore + 0.75*(\(\alpha_1*SLoCA + \alpha_2*PTL\))
6. Else TLoCA ← 0.5* RepScore + 0.5*(\(\alpha_1*SLoCA + \alpha_2*PTL\))
7. EndIf
8. EndIf
9. Return TLoCA
10. End

In next subsections, we will present the calculation method of the aforementioned parameters.

4.3.2.1 RepScore calculation

To calculate a RepScore value, which depends on the feedback ratings given by CHs, we utilise the exponentially weighted moving average (EWMA) method (James and Michael, 1990). This method permits to weigh recent observations more and do not ignore old ones (El Uahhabi and El Bakkali, 2016a). In this way, RepScore value is updated, and its older value is not ignored and reduces over time. Generally, the EWMA control chart is control limits used for verifying whether a process is in statistical control (Rajendran and Swamynathan, 2015). In this sense, it is usually utilised for detecting the malicious feedback (Rajendran and Swamynathan, 2015; Satyadeep and Vipul, 2015).

Consequently, we use the following equation for computing a RepScore value:

$$RepScore = \alpha * rtg + (1-\alpha)OldRepScore$$

(7)

where,

- RepScore: computed reputation score
- OldRepScore: old reputation score
0 < \alpha \leq 1 represents the weight assigned to the previous data. Determining its value is based on the personal experience and preference. In our case, choosing a lower value of \alpha may give more weight to the old reputation value, while the higher value may have an influence on the reputation score calculation by giving more weight to new ratings. Then, in order to control the strictness of our system, it is necessary to choose an appropriate \alpha. Its value must be not higher and not lower. In this sense, we propose that \alpha value belongs to [0.5, 0.7].

rtg: new rating leaved by a CH. It represents his appreciation for a specific CA. It is expressed as a percentage included in [0, 100]. More details about these ratings are presented in (El Uahhabi and El Bakkali, 2016c).

Additionally, to detect the malicious ratings, we use the control chart methodology. Then, the upper and lower control limits are computed as follow:

\[
LCL = \mu_0 - L \cdot \sigma \sqrt{\frac{\alpha}{2 - \alpha}}
\]

(8)

\[
UCL = \mu_0 + L \cdot \sigma \sqrt{\frac{\alpha}{2 - \alpha}}
\]

(9)

where

- L takes a value 3 (the 3-sigma control limits) or is determined by utilising the Saccucci and Lucas tables (ARL = 370)
- \sigma represents the estimated variance calculated from historical data (ratings received from CHs).
- \mu_0 represents the historical data mean.

Generally, a statistical anomaly is detected when the values are greater than UCL or less than LCL (El Uahhabi and El Bakkali, 2016a). Consequently, if RepScore value falls outside the LCL or UCL, the last user rating will be ignored and not be taken into account in calculation process of reputation score because it is considered malicious. Moreover, UCL and LCL values are updated according to the new received ratings.

It is important to note that RepScore calculation process is described in our proposed algorithm which is defined in our previous paper (El Uahhabi and El Bakkali, 2016a). Also, RepScore value is computed when a number of the participated CHs is representative, meaning that it is equal or greater than k which represents 10% of the threshold n. This threshold represents a sample size n which presents the total user subpopulation certified by a same CA. It is calculated by using a Statistical Sampling Technique (stratified sampling). Additionally, our population is the total number of the users that access and utilise e-services applications, into which our platform link is integrated. We classify this population into different subgroups (subpopulation) in accordance with a CA name that issues them certificates.

To compute a sample size n that defines our user subpopulation, we use the following formula (Glen, 1992):

\[
n = \frac{N}{1 + N \cdot e^2}
\]

(10)
where \( N \) is a size of the user subpopulation, \( n \) is the sample size, \( e \) is the precision level (\( e = 5\% \)).

### 4.3.2.2 Evaluating the SLoCA

CA Security is a complex property that cannot be easily measured. The attackers can violate a CA system security and issue fraudulent certificate. In order to protect it against security breaches, it should implement the highest security standard and is controlled by a regular audit for ensuring that its compliance with the security requirements. In this paper, we do not treat security problems of CA system. We only take into account its implemented security standard that we use it for evaluating its security level.

In addition, each CA uses software for issuing certificates, managing the CRLs, and generating public/private key pairs (El Uahhabi and El Bakkali, 2016a). When this software is certified by the common criteria (CC) standard, its security features are evaluated by establishing the EAL level (El Uahhabi and El Bakkali, 2016a). The CC presents seven levels: EAL1, EAL2, EAL3, EAL4, EAL5, EAL6, and EAL7. These levels EAL1-2 describe low assurance. Thus, the levels EAL3-4 present medium assurance. The highest level of assurance is between [EAL5-EAL7]. More details about these levels are presented in (Common Methodology for Information Technology Security Evaluation, 2005; Ruben, 2004).

In our approach, we propose to combine between a security standard implemented by a CA system, and the EAL value of software used by this authority for assessing its security level. We consider that if a CA implements a recognised security standard such as ISO/IEC 27001 covering the trustworthiness of this authority’s technology, practices or other relevant characteristics, and EAL of its software is high or medium, it is then more trusted than CA that does not adopt any standard. We explain in Table 2 how we determine the SLoCA value:

<table>
<thead>
<tr>
<th>EAL</th>
<th>Security standard</th>
<th>SLoCA</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[EAL1-EAL2]</td>
<td></td>
<td>0</td>
<td>Weak</td>
</tr>
<tr>
<td>[EAL1-EAL2]</td>
<td>ISO/IEC 27001</td>
<td>0.5</td>
<td>Medium</td>
</tr>
<tr>
<td>[EAL3-EAL4]</td>
<td></td>
<td>0.5</td>
<td>Medium</td>
</tr>
<tr>
<td>[EAL5-EAL7]</td>
<td>ISO/IEC 27001</td>
<td>1</td>
<td>Strong</td>
</tr>
<tr>
<td>[EAL5-EAL7]</td>
<td>ISO/IEC 27001</td>
<td>1</td>
<td>Strong</td>
</tr>
</tbody>
</table>

The SLoCA values are 0, 0.5 and 1 that explain respectively weak, medium, and strong. For instance, we assign SLoCA = 0.5 to a CA that adopts the security standard ISO/IEC 27001 and the EAL value of its software is one of these levels EAL1, EAL1+, or EAL2. Furthermore, updating of the SLoCA value depends to the update of an implemented security standard or EAL value.

### 4.3.2.3 Calculating the PTL

As mentioned previously, issuing certificates by a CA to other CA implicates that the issuer CA places certain trust in it (El Bakkali and Idrissi, 2001). In this paper, we try to
measure this implicit trust which is explained with PTL value. We calculate this value using an Algorithm 3.

As explained previously, upon reception of a certificate issued by a RP, our system analyses it, and extracts from it the information about its issuer CA. Then, it searches certificates which are issued by other trusted CAs for this issuer authority. These obtained certificates will be translated into XML format, parsed and analysed by our system. The XML format of a certificate is shown in Figure 6. It presents an output file format of the ASN decoder that we use it for converting an X.509 certificate into XML file. Noting this, the structure of an X.509 certificate is specified in RFC 5280 (Cooper et al., 2008).

Figure 6 X.509 certificate in XML format

```
<Certificate>
  <TBSCertificate>
    <Version>v3</Version>
    <CertificateSerialNumber>1120683489</CertificateSerialNumber>
    <AlgorithmIdentifier></AlgorithmIdentifier>
    <validity>
      <notBefore>
        <utcTime>061130000000Z</utcTime>
      </notBefore>
      <notAfter>
        <utcTime>311130000000Z</utcTime>
      </notAfter>
    </validity>
    <extensions>
      <extension name=PolicyMappings>
        <extnId id=ce 33 ></extnId>
        <critical>true</critical>
        <extnValue>
          <issuerDomainPolicy/>
          <subjectDomainPolicy/>
        </extnValue>
      </extension>
    </extensions>
  </TBSCertificate>
  <AlgorithmIdentifier>
    </AlgorithmIdentifier>
  </Certificate>
```
Firstly, our system verifies that if these obtained certificates are valid and their keys are used for a certification. If they are valid, it checks a trustworthiness of their issuer CAs by computing and evaluating their trust level using a function ‘TLIsCA’ which is illustrated in Algorithm 7. If they are trusted, it verifies the content of three certificate extensions fields: basic constraints, policy constraints, and policy mappings. These extensions are necessary for managing a trust between CAs.

Moreover, they explain implicitly a trust degree which is placed by a CA issuer in subject CA. Then, they are important for evaluating this implicit trust. For example, when a trusted issuer CA specifies in the policy mappings field that one of its CPs can be considered equivalent to a policy of the subject CA being certified, this indicates that it places certain trust in this subject. This implicit trust is presented by PTL value that we will compute for each issued certificate.

As presented in Algorithm 3, we define a calculation process of PTL value. We assign to each extension field a score which explains the importance of its content for measuring a trust placed in a CA subject. For instance, when we found that if a CP of an issuer CA is mapped to that of CA subject, the assigned score is 1 (scoremap = 1/1). Otherwise, we assign to such field a score 0 (scoremap = 0/1). See Algorithm 4.

On the other hand, the basic constraints extension is used only in CA certificate when it is marked critical (see Algorithm 5). It specifies whether the subject of the certificate is a CA and defines a parameter pathLenConstraint (path length constraint). This parameter presents a depth of the certification path which may be through a subject CA. Then, if takes a value 0, this means a subject CA can not issue any subordinate CA certificate, but can only issue end entity certificate. In this case, the assigned score is 0 (scorepath = 0/3) because a subject CA has not the necessary requirements to certify the subordinates CAs, and then the implicit trust placed by an issuer CA in it is low. Moreover, if the pathLenConstraint value is greater than 2, then this means the presence of an intermediate CA number in a path built from the CA certificate to an end entity certificate. This indicates a subject CA can issue a subordinate CA certificate, it is more trusted by an issuer CA, and then it may be principal authority. For this reason, we assign to this field a score 3 (scorepath = 3/3).

Concerning the policy constraints (see Algorithm 6), it is used for controlling the CP processing through a trust path in two ways. This extension permits to inhibit policy mapping or require a specific policy. The first one is indicated in field ‘inhibitPolicyMapping’ by specifying a number of intermediates certificates which may be present in the path before policy mapping is prohibited. If this number is equal to 1, this means certificates issued by the CA subject are only allowed to have a policy mapping. This indicates an issuer CA trusts more in subject CA. As a result, we assign to this field a score value 1 (scoreinhibitpm = 1/1) when a number of intermediates certificates, which appear in the path before prohibiting a policy mapping, is equal or greater than 1. In addition, requiring a specific policy is specified in field ‘requireExplicitPolicy’. This field indicates a number of additional certificates appearing in the path before requiring an acceptable policy. Then, when an issuer CA allows a subject CA to issue certificate containing an acceptable policy, this indicates it trusts it implicitly. For this reason, we assign to this field a score value 1 (scorerequireExpP = 1/1) when it contains a value 0 (see Algorithm 6).
Algorithm 3 Calculating a PTL on the basis of certificate extensions fields

Function PTLcert (D)

Input : XML Dom tree D

Output: PTL is a propagated trust level

1 Define the following variables:
   - scorepath : score assigned according to the path length value
   - scoremap : score assigned according to parameters of the policy mapping
   - scorepolicyconst: score assigned according to the policy constraints

2 root ← (D ←root()) // get the root node of XML file

3 listsequence ← getchildren(root) // getting children of parent node(root) using a function getchildren

4 //TBSCertificate is one of the basic fields of X.509 certificate. See Figures 2 and 6
   listTBSCertificate← getchildren(listsequence[0])

5 listchild← getchildren(listTBSCertificate[4]) // getting children of listTBSCertificate[4]

//Verifying a certificate validity
6 If listTBSCertificate[4] is equal to validity then
   listbefore← getchildren(listchild[0])

7 listafter← getchildren(listchild[1])

8 before ← listbefore[0]

9 after ← listafter[1]

10 If after<today then
   Return -1

11 Else

//Verifying a field Key Usage (the keys should be used for a certification)
12 For i← 4 to size of listTBSCertificate
   If listTBSCertificate[i] is equal to extensions then
      listchildren← getchildren(listTBSCertificate[i])

13 listextension ← getchildren(listchildren[2])

14 //Verifying a trustworthiness of the issuer CA (checking its trust level)
15 If listextension[0] is equal to id-ce 15 then
   listusage ← getchildren(listextension[2])

16 If listusage[4]=1 then
   TLoIsCA=TLIsCA(D)

17 If 0.5 ≤ TLoIsCA ≤ 1 then
   // in this case, trust level of CA is medium or high

18 For j← 2 to size of listchildren
   listtext ← getchildren(listchildren[j])

//If CA is trusted, we parse the extension fields (policy mapping, basic constraints, policy constrainst) for determining a PTL value
Trust assessment of X.509 certificate based on certificate authority

27 Select case content of listext [0]
28   case : id-ce 33
29       scoremap← policymap(listext) // Call a function policymap for getting a score assigned to policy mapping field. See Algorithm 4.
30   case : id-ce 19
31       scorepath← basicConst(listext) // Call a function basicConst for getting a score assigned to basicconstraints field. See Algorithm 3.
32   case : id-ce 36
33       scorepolicyconst← policyconst(listext) // Call a function policyconst for getting a score assigned to policy constraints field. (Algorithm 6).
34 EndSelect
35 EndFor
36 PTL ← (scoremap + scorepath + scorepolicyconst)/6
37 Return PTL
38 EndIf
39 Else Return -1
40 EndIf
41 Else Return -1
42 EndIf
43 EndIf
44 EndIf
45 EndFor
46 EndIf
47 End

Algorithm 4  Determining a score of a policy mapping field

Function policymap (listext)
   Input : listext is list of extension fields
   Output: scoremap : score assigned according to parameters of the policy mapping
1   If listext [1] = true then
2     listpolicymap← getchildren(listext[2]) // get children node of listext[2]
3     If content of listpolicymap[0] and listpolicymap[1] is not null then
4       // In this case, an issuer CA considers that one of its CPs is equivalent to a policy of the CA subject
5       scoremap = 1/1
6     EndIf
7     Else scoremap = 0/1
8     Return scoremap
9   EndIf
10 Else Return -1
11 End
Algorithm 5  Determining a score of basic constraints field

Function basicConst(listext)
   Input : listext is list of extension fields
   Output: scorepath : score assigned according to the path length value
1    If listext[1]=true
2       listbasiconst← getchildren(listext[2])
3       If listbasiconst[1]=1
4          pathLenConstraint← listbasiconst[2]
5       Select case pathLenConstraint
6          case: 0
7              scorepath=0/3
8          case: 1
9              scorepath=1/3
10         case: 2
11              scorepath=2/3
12          case: pathLenConstraint >2 or its value is omitted
13              scorepath=3/3
14     EndSelect
15    EndIf
16    Return scorepath
17 End

Algorithm 6  Determining a score of policy constraints field

Function policyconst(listext)
   Input : listext is list of extension fields
   Output: scorepolicyconst: score assigned according to the policy constraints
1    If listext[1]=true
2       listpolicyconst← getchildren(listext[2])
3    Select case size of listpolicyconst
4       case: 2
5          If listpolicyconst[0] ≥ 1 then
6             scorerequireExpP=0/1
7          Else If listpolicyconst[1] =0
8             scorerequireExpP=1/1
9          EndIf
10     EndIf
11    If listpolicyconst[1] ≥ 1 then
12       scoreinhibitpm=1/1
13    Else If listpolicyconst[1]=0 then
Therefore, we propose the following formula for computing a PTL value:

\[
PTL = \frac{\text{Scoremap} + \text{scorepath} + \text{scorepolicyconst}}{6}
\]  

(11)

where \text{scoremap} represents a score assigned to the field policy mapping, \text{scorepath} indicates a score assigned to the field basic constraints. Finally, \text{scorepolicyconst} is a score assigned to the field policy constraints.
Algorithm 7  Calculating a trust level of an issuer CA

Function TLIsCA (D)
Input : XML FILE
Output: TLoCERT: trust level of a received certificate

1 Define the following variable:
   TLoIsCA : trust level of an issuer CA
   idIsCA : identifier of an issuer CA
   n is a threshold value
   m is CHs number that provide their feedback
   K: size of listTBSCertificate
   S: size of listRDN

2 val ← array() // create an empty table
3 root ← (D ← root()) // get the root node
4 listsequence ← getchildren(root) // get its children. Function getchildren returns an array of children, given a parent node (root)
5 listTBSCertificate ← getchildren(listsequence[0]) // Extracting information about an issuer CA

6 For i ← 0 to K-1
7    If listTBSCertificate[i] = issuer then
8       listchildren ← getchildren(listTBSCertificate[i])
9       listRDN ← getchildren(listchildren[0])
10      For j ← 0 to S-1
11         listattributetpval ← getchildren(listRDN[j])
12         attribute ← getchildren(listattributetpval[0])
13         val ← content of attribute[1]
14     EndIf
15   EndFor
17 // Search a TLoIsCA value in database
18 TL0lsCA ← search (idlsCA )
19 If TL0lsCA does not exist in database Then
   RepScoreIsCA ← RepScoreRequest(idlsCA ) // sending RepScore query
   // Sending a request for m and n value
20   m ← mRequest(idlsCA )
21   n ← nRequest(idlsCA )
22   SIsCA ← SIsCAREquest(idlsCA ) // sending SL query
23   PTL ← PTLcert (D)
24   TL0lsCA = trustLevelCA (idlsCA, RepScoreIsCA, SIsCA, PTL, m, n)
25 EndIf
26 Return TL0lsCA
27 End
Furthermore, as mentioned previously, the Algorithm 7 presents a TLoIsCA calculation. This parameter defines a trust level of a CA that issues a certificate to a CA in question. Firstly, we parse the obtained certificate, and extract information about its issuer CA. Then, we verify if a trust level of this issuer authority is already calculated and exists in database. Otherwise, we compute it by applying a function ‘trustlevelCA’.

It is important to note that it often happens that we can get several certificates issued by trusted CAs for a CA in question. Then, we calculate PTL value for each one of them. In this case, we require a mechanism for combining several PTL values calculated from different certificates. This process is named trust aggregation. Algorithm 8 presents an aggregation process of PTL value.

In this context, we use a mean aggregation method (Lesani and Montazeri, 2009; Victor et al., 2011) for computing the aggregated PTL value. We consider that we have n different PTL values calculated from n certificates. As indicated previously, this PTL value evaluates the implicit trust placed by an issuer CA which its trust level is TLoIsCA.

Algorithm 8  Calculating a PTL value based on number of the obtained certificates

<table>
<thead>
<tr>
<th>Function PTLnbcert(listfilexml)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: listfilexml: list of the issued XML certificates files</td>
</tr>
<tr>
<td><strong>Output</strong>: PTL is a propagated trust level</td>
</tr>
<tr>
<td>1   PTL ← array()  // create an empty table</td>
</tr>
<tr>
<td>2   i=0  // initialise i to 0</td>
</tr>
<tr>
<td>3   While i &lt; size(listfilexml)</td>
</tr>
<tr>
<td>4   file← listfilexml[i]</td>
</tr>
<tr>
<td>5   D ← xmldocfile('file')  // Parse an XML certificate file and return an object of class ‘dom document’</td>
</tr>
<tr>
<td>6   p ← PTLcert (D)  // Call function PTLcert for getting a PTL value of certificate</td>
</tr>
<tr>
<td>7   PTL[i]=p</td>
</tr>
<tr>
<td>8   TLoIsCA[i]=TLIsCA[D]  // Call a function TLIsCA for getting a trust level of an issuer CA</td>
</tr>
<tr>
<td>9   i=i+1</td>
</tr>
<tr>
<td>10  EndWhile</td>
</tr>
</tbody>
</table>
| 11  AgPTL = \[
\frac{\sum_{i=0}^{n} TLoIsCA[i] \cdot PTL[i]}{\sum_{i=0}^{n} TLoIsCA[i]}
\] |
| 12  Return AgPTL |
| 13  End |

Then, the result of aggregation AgPTL is explained by the following formula:

\[
AgPTL = \frac{\sum_{i=0}^{n} TLoIsCA[i] \cdot PTL[i]}{\sum_{i=0}^{n} TLoIsCA[i]}
\] (12)
Figure 7  Calculation method of CPQ value
4.3.3 Calculating the CPQ

As mentioned previously, CP defines the set of rules followed by a CA for issuing a certificate. It represents an important document for assessing a certificate trust level. Practically, it is difficult for RP to judge it because it is long and technical. Then, it is necessary to automate the CP interpretation process in order to help him in making decision about a received certificate. In this context, we have proposed in our previous paper (El Uahhabi and El Bakkali, 2016a) an automated process for assessing the procedures defined in the published CP. In fact, we have suggested an algorithm for computing a CPQ which indicates that these described procedures are weak or rigorous (El Uahhabi and El Bakkali, 2016a). Firstly, we formalise and translate a CP document into XML format for representing it as tree structure that is defined in RFC 3647 (Chokhani et al., 2003), manipulating it easily, and parsing it for computing a CPQ value (El Uahhabi and El Bakkali, 2016a).

Moreover, the formal presentation of CP is detailed and described in the Internet RFC3647 (Chokhani et al., 2003). We aim to produce an identical representation for CPs which does not comprise all their announced procedures, but contains only that are important for defining and evaluating their quality. We find that identical representations could be defined by determining criteria which differ in their score value from one CP to other but have a same name in all the formalisation of CPs. These criteria are determined according to the requirements for certification service providers which are defined in the international laws related to electronic transactions and trust services, electronic signature and certification services. We give more details about these criteria in a future paper. Within the RFC3647 framework, a CP comprises several components which can consist of subcomponents, and a subcomponent may include multiple elements. For example, the component ‘Technical Security Controls’ consists of the eight subcomponents: ‘time-stamping’, ‘private key protection and cryptographic module engineering controls’, ‘key pair generation and installation’, etc. The ‘key pair generation and installation’ composes of several elements. Each element may contain multiple contents. In this context, we choose and define the components which are important comprising the subcomponents that their elements corresponding to the defined criteria.

On the other hand, we compute a score of each component by utilising a weighted average method. Wazan et al. (2016) assign a weight value to all aspects of the policy. This assigned value presents the importance of that aspect comparing to other aspects (Wazan et al., 2016). In our approach, we assign to each defined component a weight (ki) representing its importance for computing a TLoCERT. Also, we assign a weight to each subcomponent (cj) according to its priority and its importance comparing to other subcomponents for determining a TLoCERT value. These weights are defined according to the rules which will be discussed in a future paper.

As explained in Figure 7, we assign a score to each element’ content (cont_score) in accordance with its importance in the calculating of TLoCERT. The assigned score takes one of these values 0, 0.5 or 1 which presents respectively low, medium, and high. As mentioned previously, it explains the importance of each content for calculating a TLoCERT. For instance, for the element ‘key pair generation’ of the subcomponent ‘key pair generation and installation’, if its content indicates that a key pair is generated in hardware, we assign a score value 1 to this content (El Uahhabi and El Bakkali, 2016c). Generating of key pair in hardware is more trustworthy than generating it in software. Then, we sum all the scores values assigned to the contents of an element, and divide the
result on the number of these scores (n). Next, we use a same process for determining a score of each subcomponent (subcomp_score). We sum all the calculated scores values of their elements, and divide the result on the number of these scores (m). For calculating a score of each component, we sum the scores values of subcomponents multiplied by their respective assigned weight and divide the result on sum of these weights.

Finally, we obtain a CPQ value by using a same process; we sum the scores values of components (scorei) multiplied by their respective weight (ki) and divide the result on sum of these weights, as described in the following equation:

\[ CPQ = \frac{\sum_{i=1}^{n} \text{scorei} \times k_i}{\sum_{i=1}^{n} k_i} \]  

(13)

For more details about the algorithms for calculating a CPQ value, we refer to our previous papers (El Uahhabi and El Bakkali, 2016a).

5 Case study

In this section, we explain through an example how certificate trust level TLoCERT is computed in order to help RP for making a trust decision about a received certificate. In this context, we consider the following scenario:

- When a CH obtains a certificate issued by a specific CA, he uses it to confirm its identity. Accepting or rejecting this certificate by RP depends on his believe about its trustworthiness. So, he needs to make a good decision. For this reason, he accesses our system for requesting its TLoCERT value.

<table>
<thead>
<tr>
<th>CHs</th>
<th>Given rating</th>
<th>CHs</th>
<th>Given rating</th>
<th>CHs</th>
<th>Given rating</th>
<th>CHs</th>
<th>Given rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH1</td>
<td>50</td>
<td>CH14</td>
<td>73</td>
<td>CH27</td>
<td>30</td>
<td>CH40</td>
<td>50</td>
</tr>
<tr>
<td>CH2</td>
<td>40</td>
<td>CH15</td>
<td>50</td>
<td>CH28</td>
<td>20</td>
<td>CH41</td>
<td>35</td>
</tr>
<tr>
<td>CH3</td>
<td>70</td>
<td>CH16</td>
<td>45</td>
<td>CH29</td>
<td>60</td>
<td>CH42</td>
<td>45</td>
</tr>
<tr>
<td>CH4</td>
<td>20</td>
<td>CH17</td>
<td>65</td>
<td>CH30</td>
<td>10</td>
<td>CH43</td>
<td>70</td>
</tr>
<tr>
<td>CH5</td>
<td>65</td>
<td>CH18</td>
<td>60</td>
<td>CH31</td>
<td>70</td>
<td>CH44</td>
<td>60</td>
</tr>
<tr>
<td>CH6</td>
<td>30</td>
<td>CH19</td>
<td>30</td>
<td>CH32</td>
<td>55</td>
<td>CH45</td>
<td>75</td>
</tr>
<tr>
<td>CH7</td>
<td>25</td>
<td>CH20</td>
<td>50</td>
<td>CH33</td>
<td>50</td>
<td>CH46</td>
<td>50</td>
</tr>
<tr>
<td>CH8</td>
<td>10</td>
<td>CH21</td>
<td>65</td>
<td>CH34</td>
<td>55</td>
<td>CH47</td>
<td>55</td>
</tr>
<tr>
<td>CH9</td>
<td>60</td>
<td>CH22</td>
<td>75</td>
<td>CH35</td>
<td>60</td>
<td>CH48</td>
<td>30</td>
</tr>
<tr>
<td>CH10</td>
<td>50</td>
<td>CH23</td>
<td>70</td>
<td>CH36</td>
<td>50</td>
<td>CH49</td>
<td>65</td>
</tr>
<tr>
<td>CH11</td>
<td>45</td>
<td>CH24</td>
<td>40</td>
<td>CH37</td>
<td>65</td>
<td>CH50</td>
<td>55</td>
</tr>
<tr>
<td>CH12</td>
<td>75</td>
<td>CH25</td>
<td>50</td>
<td>CH38</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH13</td>
<td>55</td>
<td>CH26</td>
<td>60</td>
<td>CH39</td>
<td>45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initially, we suggest that the obtained CPQ of the CP, which is followed by a CA in issuing the certificate in question, is 0.7. Also, this authority implements a security
standard ISO/IEC 27001 and the CC level of its used software is EAL4. Then, in accordance with Table 2, the SLoCA takes a value 0.5. We suggest that a number of the certificates issued by other CAs for a CA in question is low. Then, the assigned weight to PTL is 0.25 and its obtained value is 0.6. RoCERT takes value 1. In this example, we take $\alpha = 0.6$ (weighting factor) in calculation process of RepScore.

Moreover, we propose that CHs number, that gives their appreciation for an issuer CA, is 50. Then, there are 50 ratings as illustrated in Table 3.

Firstly, we check if a CHs number achieves a threshold value or not. As signaled previously, this threshold represents a sample size $n$ which presents the total user subpopulation certified by a same CA. In this scenario, we assume that the users number that use e-services application, into which our framework link is integrated, is 10,000 users. We class this population into subgroups depending on the name of CA. In this example, we take into account only a subgroup which their members certified by the CA in question. Their number is 6,000. Then, the threshold value is calculated using an equation (10), as follows:

$$n = \frac{N}{1 + N \cdot e^{\frac{N}{10000}} \cdot 0.05^2} = 385$$

As a result, the participated CHs number is lower than $n$ (385). Likewise, it is greater than $k = 38$ which represents a 10% of the threshold $n$. In this case, we calculate a RepScore value from the received ratings and we assign to it a weight value 0.25. Thus, LCL and UCL are computed from 50 received ratings by utilising equations (8) and (9). At first, we calculate the estimated variance $\sigma$ and value $\mu_0$ from the received ratings. They are equal respectively 0.04 and 0.53. These two parameters are used for calculating LCL and UCL as follows:

$$\text{UCL} = 0.53 + 3 \cdot 0.04 \cdot \frac{0.6}{\sqrt{2 + 0.6}} = 0.59$$
$$\text{LCL} = 0.53 - 3 \cdot 0.04 \cdot \frac{0.6}{\sqrt{2 - 0.6}} = 0.45$$

As mentioned previously, if RepScore value falls outside the LCL or UCL, the last user rating will be ignored and not be taken into consideration in computation process of reputation score because it is considered malicious. Then, we get RepScore = 0.59.

By using equation (2), we compute the TLoCA:

$$\text{TLoCA} = \frac{p_1 \cdot \text{RepScore} + p_2 \cdot (\alpha_1 \cdot \text{SLoCA} + \alpha_2 \cdot \text{PTL})}{\sum_{i=1}^{3} p_i}$$

$$= 0.25 \cdot 0.59 + 0.75 \cdot (0.75 \cdot 0.5 + 0.25 \cdot 0.6) = 0.67$$

(where $\sum_{i=1}^{3} p_i = 1$, and $\alpha_1 + \alpha_2 = 1$)

We suppose that RP select default weights used for computing TLoCERT ($w_1 = 0.2$, $w_2 = w_3 = 0.4$).

Then, we compute the TLoCERT by using the equation (1):
\[ TLoCERT = \frac{\sum_{i=1}^{3} w_i \cdot R_{oCERT} + w_2 \cdot TLoCA + w_3 \cdot CPQ}{\sum_{i=1}^{3} w_i} = 0.2 \cdot 1 + 0.4 \cdot 0.67 + 0.4 \cdot 0.7 = 0.75 \]

(\text{where} \sum_{i=1}^{3} w_i = 1)

0.75 is included in [0.75, 1]. Consequently, the trust level of the received certificate is high.

6 Conclusions

In this paper, we propose an algorithm for evaluating CA trustworthiness on the basis of its certificates issued by others trusted CAs. Then, we ameliorate the trustworthiness evaluation of a certificate. In order to illustrate this new method, we present an example explaining how our solution can be used for more accurately evaluating certificate trustworthiness.

As a future work, we intend to develop a calculation method of the aggregated PTL value. In this sense, we will ask RP to determine his preference in the context of PTL aggregation. For example, if RP indicates that he wants a strict calculus, we take the minimum of PTL values computed from several certificates. On the other hand, we will specify with details a formalisation process of CP document according to the defined criteria. Finally, we aim to implement a realistic proof of concept of the proposed framework TrustCert.

References


