Extended Electronic Signature Policies

Jorge L. Hernandez-Ardieta
SeTI Research Group
University Carlos III of Madrid
Av. Universidad 30
28911 Leganes Madrid, Spain
jlopez.ha@gmail.com

Ana I. Gonzalez-Tablas
SeTI Research Group
University Carlos III of Madrid
Av. Universidad 30
28911 Leganes Madrid, Spain
aigonzal@inf.uc3m.es

Benjamin Ramos
SeTI Research Group
University Carlos III of Madrid
Av. Universidad 30
28911 Leganes Madrid, Spain
benja1@inf.uc3m.es

Arturo Ribagorda
SeTI Research Group
University Carlos III of Madrid
Av. Universidad 30
28911 Leganes Madrid, Spain
arturo@inf.uc3m.es

ABSTRACT

A signature policy collects the rules to create and validate electronic signatures under which they become binding in a particular transactional context. These policies have been widely adopted to enforce the binding property of signatures in business scenarios. However, current standards only cover the definition of the requirements to be fulfilled by a single signature. As a consequence, business models where more than one signature is required in order to make the transaction effective cannot adhere to the benefits of signature policies. This paper is the first to propose a solution where the dependences and relationships among the signatures generated in the same transaction can be established. In particular, the ASN.1 definition of an extended signature policy is presented along with the procedures to be followed by the transacting parties. This work will be submitted to the IETF PKIX Work Group to be considered as an Experimental Request For Comments document (RFC).

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development; E.1 [**Data Structures**]: [Trees]; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Security, Standardization

Keywords

Signature policy, electronic signature, public key infrastructure, extended business model, e-commerce, ASN.1

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIN'09, October 6–10, 2009, North Cyprus, Turkey.
Copyright 2009 ACM 978-1-60558-412-6/09/10 ...$10.00.

1. INTRODUCTION

Electronic signatures (signatures) play an important role in the field of electronic business. On one hand, a signature provides message authentication and integrity, protecting it from unauthorized modifications and forgeries. On the other hand, and when certain requirements are fulfilled, signatures are legally equivalent to hand-written signatures, being assigned the property of evidence in legal proceedings [2, 1, 3]. In addition, international standards consider the signature as the means to provide a non-repudiation service using asymmetric cryptography [4]. As a result, signatures become a key technology that allows the participants to make binding commitments in a transaction in a secure manner.

Several efforts have been made to standardize electronic signature and related technology. The International Engineering Task Force (IETF) and the European Telecommunications Standards Institute (ETSI) are among those organizations that have boosted the use of signatures. Dozens of standards and technical reports haven been produced since the late 90's, many of them regarding the Public Key Infrastructure (PKI) while others focusing on particular formats of signatures. The main benefit of this standardization process is that market players are able to develop interoperable implementations, offering the end users applications which are compliant to a well-known set of requirements.

A relevant deliverable made by IETF and ETSI is the signature policy document. The signature policy concept was introduced by ETSI in 2000 and later adopted by IETF in 2001. According to [6, 18], a signature policy is a document that collects a set of rules to create and validate electronic signatures under which an electronic signature is deemed valid or becomes binding in a specific business context. Therefore, a signature policy enforces the binding feature of signatures, once the requirements that must be fulfilled by the signatory and the verifier are clearly specified. If a signature has not been generated according to the policy established in the particular transaction, the commitment made by the signatory is not binding. Otherwise, the signatory cannot later repudiate her participation in the transaction. In transactions in which there is a legal requirement to use signatures and follow a specific procedure,

a signature policy can outline the outer limits respecting the application and consequences of the signatures. As an example, signatures created solely for data origin authentication purposes can be distinguished from those created for content approval or authorization.

The policy can be written both in an informal text form, provided the rules of the policy are clearly identified, and using a formal notation like ASN.1 [9] or XML [5]. In the former, there will be a requirement of human interaction at the time of deciding if the signature is valid or not, while in the latter it is possible to automatically validate it, by using suitable programs.

Signature policies can also be seen as a mechanism that enhances the level of trust and supports the verification of the identity of a signatory in a transaction, complying with the article 6 of the European Directive on electronic signatures [2]. Moreover, by specifying the conditions under which a signature is regarded as valid in a particular transactional context, a signature policy permits non-qualified signatures to be legally binding, as stated in article 5.1 of the European Directive.

In conclusion, signature policies have become an important element to orchestrate electronic business transactions while at the same time the legislation is enforced.

2. MOTIVATION AND CONTRIBUTION

To date, current technical documents on signature policy address issues relating to single signatures. The signature policy defined by ETSI and IETF supports the creation and validation of a single signature. However, there are business models where more than one signature is required in order to give the transaction legal validity or to make it effective. For instance, a transaction where a contract of sale is to be signed may be considered complete if and only if the signature of both buyer and seller is present. Sometimes, the signature of an e-notary may be necessary as well. As a consequence, no signatory should be held liable until every player has made the corresponding commitment. In this case, a single signature is useless unless every signature is generated.

The increase of paper-based processes being transposed into the digital realm makes current signature policy definition insufficient to cope with the new needs that arise. This limitation was pointed out by ETSI in a technical report published in 2003 [8]. In particular, a transaction may need certain types of multiple signatures, that can be classified in the next three groups:

- Parallel signatures, which are applied on the same piece of information. They are mutually independent signatures where the ordering of the signatures is not important. For instance, a document may be electronically signed by two or more parties, like in the contract sale example given above.

- Sequential signatures, which differ from parallel signatures in that the ordering is significant (e.g. a data flow or transaction chain).

- Embedded signatures, where one signature is applied to another. The sequence in which the signatures are applied is important and there is a strong interrelationship. An example is a process where an electronic signature must be signed (authorized) by another (e.g. an e-notary signature applied to another).

By combining these three types of signatures, all needs related to electronic signatures can be covered.

It is obvious that the current signature policy definition must be extended to include the management of multiple signatures. The purpose of this paper is straightforward. We propose an extended signature policy definition that covers the needs respecting the management of multiple signatures in a single transaction. Thereby, the presence of every signature as a binding requirement can be stipulated as well as the dependences and relationships among them. The procedures to generate and validate multiple signatures according to our extended signature policy are also specified. This work will be submitted to the IETF PKIX Work Group to be considered as an Experimental Request For Comments document (RFC). To the best of our knowledge, this is the first proposal on this issue.

The next section 3 provides the extended signature policy definition. Generation and validation procedures to enforce our policy are given in section 4. Finally, we conclude the article in section 5. The validation algorithm is defined in pseudo-code in the Appendix.

3. AN EXTENDED SIGNATURE POLICY

This section proposes an extended signature policy (ext-SP) defined in ASN.1 [7] that allows the management of a set of signatures generated in a single transaction. We have taken the ETSI technical report on signature policy for extended business model [8] as the reference document that collects the high level requirements.

Though our proposal covers the most important aspects contained in [8], we consider that some of them cannot be transposed to an automatically processable document (e.g. ASN.1, XML). For instance, the umbrella type of approach outlined in section 10.4 of [8] describes the signature policy and technical rules (section 10.4.2) that are near impossible to define in generic data structures. They cover so many different business and application domains that their specification can be done in free text form documents rather than in formal languages.

The ext-SP has been designed taking into account three different levels of abstraction:

Business Level. The first level defines the business and transactional contexts that apply to the signatures generated according to this policy.

Inter-relationships Level. The second level establishes the signatures that must be present in order to give legal effectiveness to the transaction as well as the relationships and dependences that are accepted among those signatures.

Atomic Level. In the third level, the requirements to be fulfilled by each signature on its own are defined. In practice, this level is implemented by current signature policies.

The ASN.1 definition of the ext-SP is explained in next subsections.

3.1 Base structure

The next ASN.1 type defines the base of the ext-SP, and consists of the field *extSignPolicyInfo* and the field *extSignPolicyProtection*.

```
ExtSignaturePolicy ::= SEQUENCE {
  extSignPolicyInfo       ExtSignPolicyInfo,
  extSignPolicyProtection ExtSignPolicyProtection
                          OPTIONAL
}
```

The whole information about the ext-SP is collected in the *extSignPolicyInfo* field, which ASN.1 type is the next:

```
ExtSignPolicyInfo ::= SEQUENCE {
  extSignPolicyIdentifier ExtSignPolicyIdentifier,
  extSignValidationPolicy ExtSignValidationPolicy,
  extSignContext          [0] ExtSignContext
                              OPTIONAL,
  extSignPolExtensions    [1] SignPolExtensions
                              OPTIONAL
}
```

On the other hand, *extSignPolicyProtection* field includes the information about the cryptographic algorithm applied to protect the ext-SP. If this field is not included in the ext-SP, then an external protection mechanism should be used by the parties when transmitting the ext-SP through insecure means. More specifically, *ExtSignPolicyProtection* type is defined as follows:

```
ExtSignPolicyProtection ::= SEQUENCE {
  protectionAlg AlgorithmIdentifier,
  protection    BIT STRING
}
```

The protection algorithm, defined in *protectionAlg* field (*AlgorithmIdentifier* ASN.1 type, as defined in [12]) must be applied on the DER (Distinguished Encoding Rules) encoding of the *extSignPolicyInfo* field. Different cryptographic algorithms could be used, like hash functions or digital signature algorithms. The ext-SP shall be protected by other means if the applied protection algorithm does not suffice in certain circumstances - a hash function does not prevent an attacker to modify the ext-SP content. If a digital signature algorithm is used (e.g. sha1withRSAEncryption), then the digital signature value will be encoded in the *protection* field. In this case, the digital certificate that wraps the public key corresponding to the signing private key must be provided by other means. The *subjectDN* field of the certificate should correspond to the *policyIssuerName* further defined.

The next subsections describe the fields indicated in the *ExtSignPolicyInfo* ASN.1 type. It should be noted that, following current standards philosophy, an optional extension field named *extSignPolExtensions* of *SignPolExtensions* ASN.1 type (as defined in [9, 18]) is included in most of types herein defined for future needs.

3.2 Policy Identifier

The ext-SP must be uniquely identified by both signers and verifiers. The *extSignPolicyIdentifier* field of *ExtSignPolicyIdentifier* ASN.1 type is included for that purpose:

```
ExtSignPolicyIdentifier ::= SEQUENCE {
  extSignPolicyId        ExtSignPolicyId,
  dateOfIssue            GeneralizedTime,
  policyIssuerName       GeneralNames,
  extSigPolicyQualifiers [0] SEQUENCE SIZE (1..MAX)
                OF SigPolicyQualifierInfo OPTIONAL,
  extSignPolExtensions   [1] SignPolExtensions
                             OPTIONAL
}
```

The *extSignPolicyId* field is an Object Identifier (OID) that uniquely identifies this ext-SP among all policies issued by the issuer identified by *policyIssuerName* field. The *dateOfIssue* field indicates the date when this policy was issued. Finally, the *extSigPolicyQualifiers* field includes additional qualifying information, like the location where the ext-SP can be retrieved from. Its *SigPolicyQualifierInfo* type is defined in [9, 18].

3.3 Validation Policy

The field *extSignValidationPolicy* of *ExtSignValidationPolicy* ASN.1 type is the core of the ext-SP, and describes the rules and conditions to be fulfilled by the set of signatures in order to give effectiveness to the transaction.

```
ExtSignValidationPolicy ::= SEQUENCE {
  signingPeriod        [0] SigningPeriod,
  treesOfSolutions     [1] TreesOfSolutions,
  extSignPolExtensions [2] SignPolExtensions
                           OPTIONAL
}
```

The *signingPeriod* field is of SigningPeriod ASN.1 type (as defined in [9, 18]), and identifies the period of time before and after which the ext-SP should not be used for creating signatures under this policy.

The *treesOfSolutions* field, further detailed, contains a set of graphs where each one represents a tree of signatures that defines the dependences and relationships among them. This field implements the **Inter-relationships Level** mentioned above.

Trees of Solutions

Taking into account the three types of signatures defined in section 2, it is obvious that a tree can be derived from the generated set of signatures. A tree is a connected graph with n vertices (nodes) and $n-1$ edges, and thus where there are no cycles. In particular, the tree used to represent the set of signatures has the next specific properties as well:

- The tree is a rooted tree in which the root node (level 0) represents the signed document and the rest of nodes correspond to signatures.
- The edges have a natural orientation away from the root. The tree expands from the root towards the leaf nodes, which are nodes with no child.
- The graph is unweighted, that is, there are no edge weights.
- The tree is irregular: each node (signature) not being a leaf node can have a different positive grade, that is, it can have as many children as needed. Leaf nodes have positive grade 0.

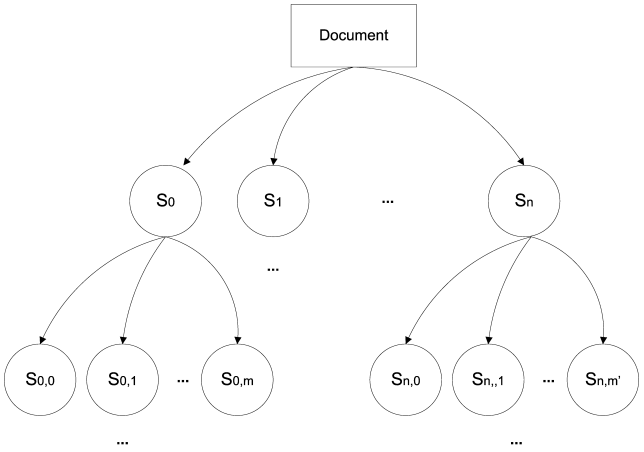

Figure 1: A generic tree of signatures.

- Every node has a negative grade 1 (number of parent nodes), except the root node which has negative grade 0.

Figure 1 depicts a generic tree of signatures. Signatures in level 1 of the tree correspond to Primary Signatures (PS), which are either parallel or sequential signatures. The rest of signatures correspond to CounterSignatures (CS), which are embedded signatures that can be applied to either a PS or another CS. Besides, signatures which are children of the same parent behave as PS among them. The difference is that the signatures are applied to another signature instead of the document. Though the arrows follow a top-down direction (for search purposes), a signature in level n is applied to the parent signature in level $n-1$.

This first approach would seem valid to implement any multisignature based transaction. However, we have noticed that, in certain scenarios, more than one tree of signatures can make the same transaction be legally effective. It is the case of fair exchange [17, 14] and fair non-repudiation [15] protocols, where the transaction can be finished (become effective) either by completing the main protocol or one of the specified subprotocols. Many other scenarios can be customized to follow this approach. For instance, an e-commerce protocol where two parties must sign a document can decide to chose either countersigning each others signature or let an authorized e-notary to do it.

In order to support this type of transactions, the Trees of Solutions (TSo) consists of a sequence of trees, each of which (named TSi) as represented in figure 1. During the validation stage (see section 4.2), the verifier must check if the multisignatures evaluated match one of the trees defined in TSo. The transaction is made effective providing that one tree is completely satisfied.

```
TreesOfSolutions ::= SEQUENCE OF treeOfSignatures
                     TreeOfSignatures

TreeOfSignatures ::= SEQUENCE OF signature
                     Signature
```

The *Signature* ASN.1 type defines the information of a particular node (signature) in a tree of signatures:

```
Signature ::= SEQUENCE {
  identifier             INTEGER (0..MAX),
  signer                 INTEGER (0..MAX),
  acceptableSignPolicies AcceptableSignPolicies,
  allowedCommitmentTypes SelectedCommitmentTypes,
  counterSignatures      [0] TreeOfSignatures
                             OPTIONAL,
  timingAndSequence      [1] TimingAndSequence
                             OPTIONAL,
  extSignPolExtensions   [2] SignPolExtensions
                             OPTIONAL
}
```

Each node (signature) is uniquely identified by the *identifier* field. This information is used to specify timing and sequence dependences, as shown further. The signer that must perform this signature is uniquely represented in a figurative sense by the *signer* field. No signer specific information can be used (e.g. subject distinguished name or subject alternative name) as the ext-SP issuer does not know a priori which signer will actually perform the signature. A signer can appear in as many signatures of the tree as needed, but signatures to be generated by different signers must contain different *signer* values.

The *acceptableSignPolicies* field contains the signature policies (SP) OIDs [9, 18] that can be used by the signer when creating this signature. This field implements the **Atomic Level** mentioned above.

```
AcceptableSignPolicies ::= SEQUENCE OF signPolicyId
                                       SignPolicyId

SignPolicyId ::= OBJECT IDENTIFIER
```

The *allowedCommitmentTypes* field restricts the commitment types that can be assumed by the signer when producing this specific signature. This field is of *SelectedCommitmentTypes* ASN.1 type (as defined in [9, 18]). These commitment types must be consistent with those included in the acceptable signature policies herein indicated.

Each signature can have a finite number of children nodes, which are represented by a *TreeOfSignatures* in the *counterSignatures* field. As a result, the tree is represented by following a recursive method, in which the leaf nodes of the tree will not have the *counterSignatures* field.

The time frame during which a signature must be generated and the sequential relationships with other signatures are described in *timingAndSequence* field.

```
TimingAndSequence ::= CHOICE {
  absoluteTimingAndSequence [0] SigningPeriod,
  relativeTimingAndSequence [1] SEQUENCE OF
                          RelativeTimingAndSequence
}
```

The *TimingAndSequence* type supports the specification of sequential signatures, as defined in section 2. It allows any signature to have as many timing and sequence dependences on other signatures as needed. There are three possibilities:

- A signature has no actual dependence on any other signature (e.g. primary signatures).
- A signature has no dependence on other signatures but it must be performed within a period of time (e.g. a

primary signature to be performed not before 17/07/1997 00:00:00 GMT and not after 17/07/2007 00:00:00 GMT). We define this dependence as an absolute dependence.

- A signature has certain dependences on other signatures, either sequential or embedded. These are considered as relative dependences.

The first case is achieved by omitting the *timingAndSequence* field of *Signature* type above. The second case is implemented by selecting *absoluteTimingAndSequence* field in *TimingAndSequence* type. To define one or more relative dependences, the *relativeTimingAndSequence* field must be selected, which ASN.1 type is the next:

```
RelativeTimingAndSequence ::= SEQUENCE {
  pathToRefSignature SEQUENCE OF INTEGER,
  maxDelta           DeltaTime OPTIONAL
}
```

The *pathToRefSignature* field indicates the path of node *identifier* field values from a signature located in level one of the tree to the signature with which there is a timing and sequence dependence. The *maxDelta* field indicates the maximum time delay allowed from the referenced signature's signing time during which this signature can be performed. That is, this signature must be performed in a period of time defined by $[t0, t0 + maxDelta]$ where $t0$ is the referenced signature's signing time. If this field is omitted, it means that this signature must be generated after the referenced signature but with no time limit.

In order to obtain accurate and reliable time references, signatures should be time stamped, following the requirements specified in the *TimestampTrustCondition* ASN.1 type [9, 18].

3.4 Business and Transactional Domains

The context in which the extended signature policy applies is defined in the field *extSignContext* of *ExtSignPolicyInfo* ASN.1 type. This field is of *ExtSignContext* type, and implements the **Business Level** mentioned above:

```
ExtSignContext ::= SEQUENCE {
  businessApplicationDomain [0] SigPolicyQualifier-
                                Info OPTIONAL,
  transactionalContext      [1] SigPolicyQualifier-
                                Info OPTIONAL,
  disputeResolution         [2] SigPolicyQualifier-
                                Info OPTIONAL,
  audienceConditions        [3] SigPolicyQualifier-
                                Info OPTIONAL,
  extSignPolExtensions      [4] SignPolExtensions
                                OPTIONAL
}
```

The *businessApplicationDomain* field outlines the business domain in which the ext-SP is suitable for use, e.g. sale of goods/international trade transactions, e-Government transactions between citizens and e-Administration, e-health services, etc. It covers high-level and sector-oriented domains. On the contrary, the *transactionalContext* field provides additional information about the transactional context, e.g.: draft of a contract, purchase by means of online service, exchange of design documents, etc. This information should match with the *fieldOfApplication* field of each signature policy, described in *SignPolicyInfo* ASN.1 type [9, 18].

Disputes on a specific event or action taken by any party in a transaction may arise in a future. A dispute must be resolved by a third party with authority to do so, taking as information for the resolution the evidence generated in the transaction. Electronic signatures may act as non-repudiation evidence if adequate policies used by the parties enforce them. In that case, the third party must consider if the conditions established for the transactions have been fulfilled by the parties. The dispute resolution procedures are contained in the *disputeResolution* field. It allows the ext-SP issuer to specify a binding text to be considered by the parties when using this policy for generating and validating signatures, and by the third party for resolving a dispute.

Finally, *audienceConditions* states the conditions under which a signature may be relied upon. e.g. the signature only valid in a specified jurisdiction, where laws exist which recognize the legal validity of signatures created under conditions as specified in the policy. This field may include provisions relating to the intended effectiveness of signatures, where multiple signatures are required, e.g. the signature must be countersigned to be relied upon.

4. USING THE POLICY

Section 3 above described the ASN.1 definition of the extended signature policy (ext-SP). This section deals with the steps a signer and a verifier must follow in order to adhere to this policy. We suppose that a signature application that supports the ext-SP is available to both signers and verifiers, the ext-SP has been retrieved by the application, and its integrity, authenticity and validity period verified. Also, the acceptable signature policies indicated in the *acceptableSignPolicies* field of each signature can be retrieved and verified as well.

4.1 The Signature Generation Process

The steps a signer must follow in order to generate a signature according to the ext-SP are the following:

1 - Policy information visualization. As a first step, the signer should be shown the tree of signatures information, differentiating between the signatures present in that moment and the signatures that are left to complete the transaction. The application should permit the signer to visualize all the information of each signature (node) of the tree, including the acceptable signature policies, the allowed commitment types and the timing and sequence dependences. Furthermore, the business and transactional domains information contained in the *extSignContext* field of the policy should be displayed to the signer.

2 - Certificate selection. The signer should then be asked to select the digital certificate to use. The certificate must be selected among available certificates with associated signing private key.

3 - Candidate nodes. Once chosen, the application must select the nodes that potentially represent the signature to be generated by the signer. These candidate

nodes will be among the signatures that are still left to be performed. The application must extract the *subjectDN* information from the certificate. This information represents the digital identity of the signer. Afterwards, the application must detect if the signer has already performed any signature in the tree. In that case, the association *subjectDN* and *signer* identifier can be done, and the candidate nodes be easily highlighted. Otherwise, there may be several possible *signers* to which the *subjectDN* can be associated. The candidate nodes are highlighted anyway. Candidate nodes that do not comply with the timing and sequence dependences must be discarded.

4 - Node selection. In this step the signer must select the candidate node over which she is going to apply the signature. As a result, the information to be signed can be the document (root node) or another signature. The signer must also select the type of commitment to make among those available in the selected node. The application will then allow the signer to select a signature policy among the policies in the node that support the selected commitment type.

5 - Signature computation. Finally, the signer performs the signature. The signature generation process is carried out according to the signer rules of the selected signature policy, as stated in the corresponding *signerRules* field (see [9, 18]).

The signer can abort the generation process at any time before step 5. If, at step 3, the application obtains a void set of candidate nodes, the generation process must be automatically aborted. This situation implies that either the set of signatures does not comply with the ext-SP requirements or the signer cannot generate a compliant signature at that moment basing on the selected certificate.

It is important to emphasize that a signer may also act as a verifier. Every signature that is already present must be verified by the signer before computing her signature. The verification procedure could be delegated to a trusted party, if possible. In those cases, the signer (or the third party) must verify previous signatures and collect the required validation data as specified in the signature policy. Finally, the generation process could be automated for simplification purposes, if necessary, on a case-by-case basis.

4.2 The Validation Process

During the validation stage, the verifier will check if the set of signatures fulfill the requirements established in the referenced ext-SP, and if each signature is compliant with its corresponding referenced signature policy.

The next subsection 4.2.1 provides an overview of the designed validation strategy. Subsection 4.2.2 describes the pruning methods that have been incorporated. A refinement stage needed to complete the validation process is explained in subsection 4.2.3. Finally, the way the extended signature policy can be integrated into current standard signature formats is given in subsection 4.2.4.

4.2.1 Approach

Each Tree of Signatures (TSi) belonging to the Trees of Solutions (TSo) and the generated set of signatures (SSi) represent a tree graph with the properties explained above in subsection 3.3. Our validation process combines a Depth-First-Search (DFS) algorithm to explore the SSi and a modified Breadth-First-Search (BFS) algorithm to locate each signature of SSi in a TSi. Other strategies different than the proposed one may achieve the same aim, which is to evaluate if SSi can be mapped to at least one TSi, what would imply that SSi is compliant with the ext-SP. Furthermore, two different pruning methods that improve the process performance and effectiveness have been integrated.

Graph search algorithms like BFS or DFS follow an established strategy to explore a graph or search a node in a graph. In particular, BFS starts at the root node and explores all the neighboring nodes. Then, for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until the graph has been completely explored or the node is found. DFS starts at the root node and explores along each branch until the leaf nodes before backtracking. It continues until the whole graph has been explored or the goal node found.

BFS and DFS are combined and modified in order to cope with next particularities:

- We need to search as many nodes in TSi as signatures in SSi, not just one signature (node).
- Each signature in SSi can be found only in a specific level of TSi. Therefore, we can apply an important filter in the search strategy. As an example, signatures located in first level of SSi will be searched only in TSi nodes in level 1. Their children signatures (countersignatures) will be searched only in TSi nodes in level 2, and so forth.
- A signature can be matched (found) with more than one node in the specific level of TSi. Therefore, our algorithm does not stop when a signature is matched, but when all candidate nodes have been explored. The reason is that different signers can make the same commitment type and use the same signature policy when creating their signatures. As a consequence, there is no way of differentiating which node of TSi better matches with each signature, except when a pruning is performed (see subsection 4.2.2).
- There is a natural reduction in the number of searching possibilities for a particular signature. The reason is twofold: a signature can be found in a specific level of TSi but only in those nodes that are children of nodes matching the parent signature. Besides, a reduction can be applied to the potential search paths when a pruning has been done (see subsection 4.2.2). The nodes used by the algorithm in order to find a signature are called candidate nodes.

As a result, SSi is explored by using a DFS strategy that will visit every signature. Each time a signature is visited, the algorithm tries to match it with the corresponding candidate nodes of TSi. The candidate nodes are explored by following a BFS strategy, but with the particularities explained above.

Each signature is represented by a set of information that allows the algorithm to perform the search. These search parameters are the next:

- The **subject distinguished name**, which can be retrieved from the *SubjectDN* field of the signer's certificate.

- The **signature policy** (OID value) used by the signer when creating the signature. This value can be obtained from the policy identifier, included in the signature as a signed attribute. Please refer to Explicit Policy-based Electronic Signatures (EPES) formats in [11, 16] for further information.

- The **commitment type** (OID value) made by the signer respecting the signed information. This information is also included in the signature as a signed attribute. Please refer to EPES formats in [11, 16] for further information.

On the other hand, the TSi node information used by the algorithm for the search is the following:

- The **signer** field of a node, which is a numeric representation of the signer of the node.

- The **acceptable signature policies**, which OIDs are contained in the *acceptableSignPolicies* field of the node.

- The **allowed commitment types**, which OIDs are contained in the *allowedCommitmentTypes* field of the node.

Therefore, the algorithm will try to match a signature with the candidate nodes by using previous search parameters and node information.

Next three end conditions for the algorithm apply:

- If a TSi is completely satisfied after having processed SSi, then SSi represents a solution, and the transaction is made effective and considered complete.

- If after having completely processed SSi, there is at least one TSi that is partially satisfied (no deadlock is found), then SSi represents a partial solution that may become complete if the rest of required signatures are generated. We define a *deadlock* as the situation where a signature cannot be matched with any candidate node, and as a result the search algorithm cannot continue processing the SSi.

- Otherwise, SSi is not compliant with the ext-SP conditions, and the validation fails.

Definition 1. Providing that the number of signatures in SSi and nodes in TSi are the same, a TSi is completely satisfied if, once the validation algorithm is finished, each signature of SSi is matched with at least one node in TSi and every node of TSi is matched with at least one signature of SSi. In other words, the matching function between TSi and SSi is a surjective function.

Definition 2. SSi is compliant with ext-SP if at least one TSi is completely satisfied. The set of signatures can be complete - every signature needed to complete the transaction has been generated - or incomplete - there are still some required signatures left.

4.2.2 Pruning methods

Our combined algorithm supports backtracking, like a normal DFS algorithm, but applying two pruning methods herein defined.

The aim of the pruning methods is to dynamically reduce the number of possible nodes that can be matched with each signature while the search algorithm is working. As a result of this on-the-fly refinement, the number of paths to explore in further steps can be substantially reduced, improving the computational cost and memory consumption. Besides, pruning methods can detect a deadlock situation that may not be detected otherwise, or detected in a later step.

Signer-based pruning

The first time a *subjectDN* is used as a search parameter, its corresponding signature will be matched with one or more nodes of TSi in a certain level. As a result, the *subjectDN* will be associated to one or more *signer* identifiers. There are two possibilities providing that a deadlock is not found:

Non-definite signer assignment. If several matches have been made, the *subjectDN* cannot be assigned to a definite signer identifier. However, the algorithm can still use this list of assigned signer identifiers to filter candidate nodes in future steps.

Definite signer assignment. If there is only one match, then a definite signer identifier assignment is done. From that moment onwards, the *subjectDN* is linked to a unique signer identifier. This association is used by the algorithm to discard candidate nodes in future steps.

As an example of a non-definite assignment, suppose that a signature with *subjectDN* equals to *CN = Researcher, OU = Computer Science Department, O = University Carlos III of Madrid* is matched with two nodes in level 2 of TSi, the first node with signer's $id = 1$ and the second one with signer's $id = 2$. The pruning here consists in that future searches of signatures with the same *subjectDN* can only be matched with nodes with signer identifiers equal to 1 or 2.

On the other hand, suppose that a signature with a different *subjectDN* is matched, after having processed its candidate nodes, with just one node with signer's $id = 3$. In this case a definite signer assignment is done. Therefore, future processing of signatures with that *subjectDN* can only be matched with nodes with $id = 3$. Moreover, future processing of signatures with different *subjectDN* cannot be matched with nodes with $id = 3$.

Once a signature has been firstly matched with certain nodes at a specific level of TSi, the algorithm knows that the signer identifier to which that *subjectDN* will be finally assigned is one of those identifiers, and no other. Otherwise, the matching would imply a contradiction.

A signer-based pruning can also be applied while backtracking. For instance, say that a signature with a concrete *subjectDN* has been matched with two nodes. Node n_0 with signer's $id = 1$ and node n_1 with signer's $id = 2$. Suppose that, while processing the signatures below it (countersignatures), the signer identifier 1 is definitely assigned to a different *subjectDN*. As a result, while backtracking, the node n_0 with signer identifier 1 is deleted as a matched node for the signature. Moreover, by discarding this node, and as only

one matched node remains, another definite signer assignment ($id = 2$) is done. It should be noted that a deadlock situation would have occurred if signer identifier 2 had been also definitely assigned to a countersignature with different *subjectDN*. Due to the signer-based pruning, the signature in the example would end without matched nodes, and the validation would fail.

In order to be able to manage this pruning method, the algorithm must maintain an updated list of assigned signer identifiers (both definite and non-definite) that has to be looked up in each step.

Path-based pruning

As previously mentioned, the algorithm processes each signature focusing on its corresponding TSi search level. Once a signature has been matched with certain nodes, the countersignatures are processed in the next deeper level, but using as candidate nodes those that are children of nodes matching the parent signature, as explained in subsection 4.2.1 above. This pruning method consists in the following: Once the leaf signatures are reached, the algorithm backtracks, providing a list of the parent nodes that resulted in a matching. Parent nodes not included in the list are discarded as matched nodes for the parent signature. As a result, the space of possibilities, that is, actual nodes which requirements are fulfilled by that signature, is reduced and refined.

For example, let T be a TSi of depth n, and S be a SSi of depth n as well. Let s_{n-1} be a signature of S in level $n-1$, and s_n be its countersignature in level n. If s_{n-1} is matched with nodes $node1_{n-1}$, $node2_{n-1}$ and $node3_{n-1}$ of T, then the algorithm uses nodes children of $node1_{n-1}$, $node2_{n-1}$ and $node3_{n-1}$ as candidate nodes for countersignature s_n. However, suppose that only a candidate node child of $node2_{n-1}$ produces a matching with countersignature s_n. As a result, when the algorithm backtracks, nodes $node1_{n-1}$ and $node3_{n-1}$ are discarded as matched nodes for s_{n-1}. In this case we say that a path-based pruning has been done.

A path-based pruning can result in a definite signer identifier assignment if, after having applied the path-based pruning, only one matched node remains.

Figure 2 illustrates an example where both prunings are applied. Signer identifiers, acceptable signature policies and allowed commitment types of TSi nodes are shown. The subject distinguished name, signature policy used and commitment type made of each SSi signature are shown as well.

Though it would seem that the SSi is compliant with the TSi, a deadlock occurs during the validation. In the first step, signature *SDN0* is matched with nodes *id1* and *id2* in first level, as the signature policy used and the commitment type made are among those permitted by both TSi nodes. When processing the first countersignature (*SDN1*), the algorithm evaluates the four children nodes in level 2 as candidate nodes. This countersignature is only matched with the first candidate node (*id1*). Then, a definite signer identifier assignment is done. As it is a leaf signature, the algorithm backtracks. At this point, the algorithm detects that only the path of the node *id1* in first level resulted in a matching. As a consequence, node *id2* in first level is discarded as a matched node for signature *SDN0*. Besides, the algorithm detects that the signer identifier *id1* has been definitely assigned to a different subjectDN (*SDN1*). Therefore, the matched node *id1* is discarded as well, and a deadlock occurs.

As can be seen, both pruning methods feed each other, improving the overall performance and accuracy of the algorithm.

4.2.3 Refinement Stage

Due to the followed DFS strategy, SSi is processed in a top-down and left-right manner. As a result, prunings and signer identifier assignments produced during the tree evaluation have no effect on the already processed signatures. For that reason, and providing that a deadlock has not occurred during the search, a refinement stage has to be applied before concluding the validation algorithm.

At this stage, the information generated during the search is analyzed. The information mainly covers the matches between signatures and nodes, definite and non-definite signer assignments, and timing and sequence dependences, which have not been evaluated so far.

The aim of this stage is threefold:

- Detect possible deadlocks not detected yet. A signer identifier can be definitely assigned to a *subjectDN*, but previously processed signatures with different *subjectDN* still maintain the same assignment. On the other hand, signatures may be matched with nodes which parent nodes do not derive in a matching for their right-hand side siblings (path-based pruning), but their matchings are not updated during the search. For these reasons, during this stage both pruning methods are iteratively applied until no change is produced, that is, a stable version of the solution is obtained or a deadlock is found.

- Analyze if every node of TSi is matched with at least one signature of SSi (SSi satisfies this TSi).

- Evaluate if the dependences among the signatures are fulfilled. It is not possible to analyze these constraints until each signature has been matched with nodes. Therefore, this analysis must be done at this stage. Ideally, at this point each signature is matched with just one node, and the evaluation of the dependences is straightforward. However, sometimes there can still be multiple matches. In these cases, each combination must be considered for the timing and sequence analysis until either a solution is found or every combination has been tested without producing a solution. In this case, the validation fails. It is important to remark that an ext-SP wrongly defined can provoke a deadlock respecting the timing and sequence dependences. For instance, if a signature $s1$ must be generated after the signature $s2$, but $s2$ must be generated after $s1$ (impossible condition), then a deadlock occurs. The ext-SP issuer is responsible for defining a correct ext-SP.

As the very last step, and once a solution is found, each signature must be evaluated according to the requirements of the referenced signature policy (SP). Only when this final validation step is successfully completed, the set of signatures can be said to comply with the extended signature policy.

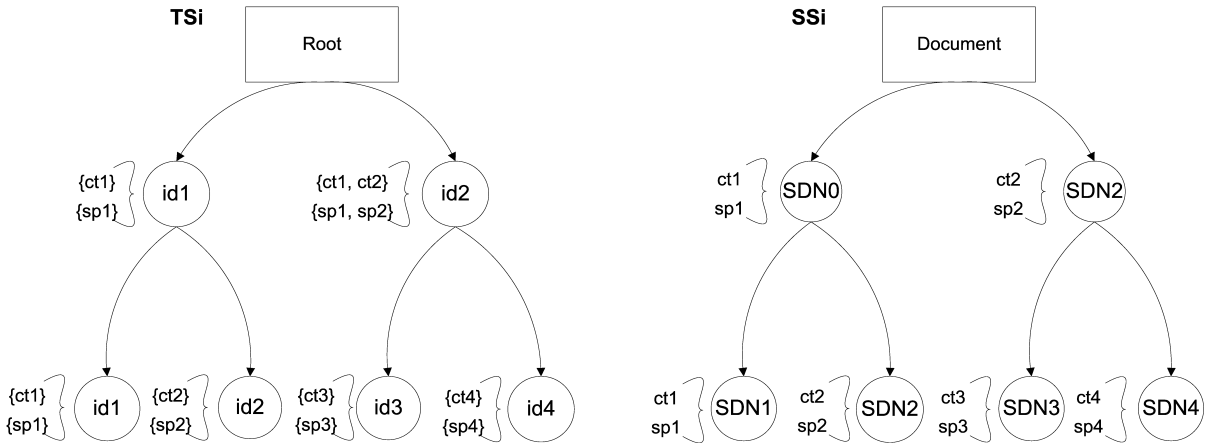

Figure 2: Deadlock example.

4.2.4 Integration in AdES formats

Advanced Electronic Signature Formats (AdES) like CAdES [11, 16] and XAdES [10, 13] have adopted the inclusion of the signature policy reference as a signed attribute in their EPES version. By signing over the signature policy identifier, the signer explicitly indicates that she has applied the signature policy in creating the signature. The verifier is also able to retrieve the referenced signature policy content and thus validate the signature accordingly. In order to unambiguously identify the referenced signature policy that is to be used to verify the signature, the signed attribute includes an identifier unique in the domain of the signature policy issuer and a hash of the signature policy document.

In order to support the usage of extended signature policies in the same way, a new signed attribute has to be defined. We propose the following object identifier to identify the new ext-signature-policy-identifier attribute (id-aa-ets-extSigPolicyId):

```
{iso(1) member-body(2) us(840) rsadsi(113549)
pkcs(1) pkcs9(9) smime(16) id-aa(2) 49}
```

The ext-signature-policy-identifier attribute values have *SignaturePolicyIdentifier* ASN.1 type, as defined in [11, 16]:

```
ExtSignaturePolicyIdentifier ::= SignaturePolicy-
                                 Identifier

SignaturePolicyIdentifier ::= CHOICE {
  SignaturePolicyId      SignaturePolicyId,
  SignaturePolicyImplied SignaturePolicyImplied
}
```

The signature would then include as signed attributes the signature policy and extended signature policy references along with the commitment type.

5. CONCLUSIONS

Signature policies are an important step forward since they customize the requirements an electronic signature must fulfill in a particular transactional context. However, current standardized signature policy definition is focused on the generation and validation rules for a single signature. The management of multiple signatures where relationships exist in a unique transaction is not possible. This scenario limits the usage of signature policies in business scenarios where the presence of more than one signature is a must, like e-commerce, contract signing protocols, e-Government applications or certified email systems.

In this paper we have proposed a complete framework to cover this need. In particular, the ASN.1 definition of an extended signature policy along with the generation and validation procedures to be followed by signers and verifiers have been presented. To the best of our knowledge, we are the first to propose a solution that resolves this issue. We have designed the solution taking into account current standards. As a result, the extended signature policy framework herein described can be easily integrated into existent signature applications and processes. Finally, our design allows a flexible definition of extended signature policies, supporting complex business models.

This work will be submitted to the IETF PKIX Work Group to be considered as an Experimental Request For Comments document (RFC).

6. ACKNOWLEDGMENTS

This research has been partially supported by the Ministry of Industry, Tourism and Trade of Spain, in the framework of the project CENIT-Segur@, reference CENIT-2007 2004. (https://www.cenitsegura.es)

7. REFERENCES

[1] Electronic Signatures in Global and National Commerce Act, Federal Trade Commission, Department of Commerce, United States of America, 2000.

[2] European Directive 1999/93/CE of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

[3] UNCITRAL Model Law on Electronic Signatures with Guide to Enactment, United Nations, 2001.

[4] ISO/IEC 13888-3 Information technology - Security techniques - Non repudiation - Part 3: Mechanisms Using Asymmetric Techniques. International Organization for Standardization, 1997.

[5] ETSI TR 102 038 - TC Security - Electronic Signatures and Infrastructures (ESI). XML format for

signature policies v1.1.1. European Telecommunications Standards Institute (ETSI), April 2002.

[6] ETSI TR 102 041 - Signature Policies Report v1.1.1. European Telecommunications Standards Institute (ETSI), February 2002.

[7] ITU-T Recommendation X.680. Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation. ITU-T, 2002.

[8] ETSI TR 102 045 - Electronic Signatures and Infrastructures (ESI); Signature policy for extended business model v1.1.1. European Telecommunications Standards Institute (ETSI), March 2003.

[9] ETSI TR 102 272 - Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies v1.1.1. European Telecommunications Standards Institute (ETSI), December 2003.

[10] ETSI TS 101 903 - XML Advanced Electronic Signatures (XAdES) v1.3.2. European Telecommunications Standards Institute (ETSI), March 2006.

[11] ETSI TS 101 733 - Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES) v1.7.4. European Telecommunications Standards Institute (ETSI), July 2008.

[12] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. *RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Internet Engineering Task Force (IETF), 2008.

[13] J. C. Cruellas, G. Karlinger, D. Pinkas, and J. Ross. *XML Advanced Electronic Signatures (XAdES)*. World Wide Web Consortium (W3C), 2003.

[14] J. L. Hernandez-Ardieta, A. I. Gonzalez-Tablas, B. R. Alvarez. An Optimistic Fair Exchange Protocol based on Signature Policies. *Computers & Security*, 27(7-8):309–322, December 2008.

[15] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25:1601–1621, April 2002.

[16] D. Pinkas, N. Pope, and J. Ross. *RFC 5126 - CMS Advanced Electronic Signatures (CAdES)*. Internet Engineering Task Force (IETF), 2008.

[17] I. Ray and I. Ray. Fair exchange in e-commerce. *ACM SIGecom Exchange*, 3(2):9–17, May 2002.

[18] J. Ross, D. Pinkas, and N. Pope. *RFC 3125 - Electronic Signature Policies*. Internet Engineering Task Force (IETF), 2001.