Towards Extensible Signature Policies in Brazil: A Case Study

Maurício de Oliveira, Martín Vigil, Marcelo Carlomagno Carlos, and Ricardo Custódio
Federal University of Santa Catarina
Florianópolis 88040-900, Brazil
Email: mauricio.so@posgrad.ufsc.br, martin.vigil@ufsc.br, me@marcelocarlos.com, custodio@inf.ufsc.br

***Abstract*—Signature policies are a set of rules to create and verify signatures. For example, they specify the signature algorithm that a signer should employ and the evidence a verifier must use to verify a signature. Brazil has adopted signature policies to regulate legally binding signatures. Our contribution is to analyze and improve the use of signature policies in Brazil. Our analysis shows that the current policies present a serious issue in situations where the requirements of a signature change. A practical example is when the validity of a signature needs to be extended, e.g., to guarantee non-repudiation time-stamps become required. To address this issue, we propose the *extensible signature policies* which, in addition to the definition of how a signature is created and verified, specifies which further policies can be applied to the signature. We demonstrate the efficacy of our solution by performing new signature policies and developing a prototype. Furthermore, we argue that our *extensible signature policies* solution does not require significant changes on existing signature methods and infrastructure.**

Keywords–Signature Policy; Digital Signature; Public Key Infrastructure; Time-Stamp.

I. INTRODUCTION

Digital tools and solutions are becoming a more constant part of our routines. For instance, people are increasingly engaging in e-commerce, whose size has doubled from 2011 to 2014 in the world [1]. However, despite the evolution of security mechanisms, frauds are still increasing. For instance, retailers and buyers have to deal with the risk of fraudulent e-commerce transactions, which has increased by 30% in 2015 in the United States [2].

To address these issues, digital signatures [3] pose an interesting solution. The reason is that signatures can provide integrity, authenticity, and non-repudiation. Integrity means one can check whether a piece of data has been unexpectedly altered. Authenticity allows identifying who originated a signature on the data. Non-repudiation prevents the originator of this signature from denying that he or she is the originator. Therefore, digital signatures are useful to prevent criminals from manipulating data or transactions without being noticed. Moreover, signatures are legally accepted to establish commitments in several countries, e.g., Brazil [4].

When using signatures, the involved parties need to agree beforehand how signatures are created and verified. This is needed because there are several options for creating and verifying signatures. For example, a party may prefer a specific signature algorithm to others, e.g., the *Elliptic Curve Digital Signature Algorithm*[5] rather than RSA[5]. Moreover, when signing a document, further data can be signed together with the document. For instance, the certificate containing the signer's public key. This certificate and the corresponding revocation status can be also attached to the signature to help verifiers to check the signature. Still, without knowing the parameters used for generating the signature, verifiers may not be able to verify it.

A solution for the issue above is the adoption of signature policies. They have been proposed by the European Telecommunications Standards Institute [6][7] and are a set of rules for creating and verifying signatures. Brazil has adopted signature policies to regulate the use of legally binding signatures. The adopted policies can be used to generate the *basic*, *dated*, *complete*, and *archival* signatures [8]. The basic signatures are for authenticating data and no additional information is attached to the signatures. The dated signatures have a time reference provided by a time-stamp. This time-stamp is attached to the signatures. The complete signatures have the signer's complete certificate chain, the corresponding revocation statuses, and a time-stamp. The archival signatures are used for archiving and have the signer's chain, the corresponding revocation statuses, and one or more time-stamps. After signing a document, the used policy cannot be changed because signing parties sign the document and the used policy together.

In this paper, we analyze the use of signature policies in Brazil and propose improvements. Our analysis identified a serious issue when using signature policies. Consider that one party created a basic signature to indicate his or her commitment to a transaction with a second party. Some time later, the second party opens a dispute in court providing the signature as a proof of the commitment. The problem is that the signature can become invalid before the dispute starts. This can happen because the validity of a signature ends when the signer's certificate expires or is revoked, e.g., due to private key compromise. In this case, the non-repudiation property of the signature is lost because one cannot prove that the signature was created *while* the signer's certificate was valid. A straightforward solution would be attaching a time-stamp to prove that the signature existed *before* the signer's certificate became invalid. However, the basic signature policy does not allow the use of time-stamps, preventing the addition of new times-stamps. Similar problem also affects dated and complete signatures. Although they allow using a time-stamp to extend the validity of a signature, the time-stamp itself has limited validity since it also relies on a signature. Note that one could apply a second time-stamp on the first time-stamp to extend the validity of the first time-stamp, but this is only allowed by the archival signature policy.

Solutions for this issue would be to identify the required validity time of signatures before creating them or using the

 ISBN: 978-1-61208-493-0

signatures with the longest validity, i.e., archival signatures. However, these are not good solutions. Parties may not always be able to identify the required validity time of their signatures in advance. If an archival signature is used, the policy requires certificates, revocation statuses, and time-stamps to be added even when they are not necessary, e.g., at the beginning of the signature validity. These initially unnecessary data require approximately 6 times as much space as a basic signature. Moreover, this overhead is magnified in repositories containing several signatures, because the Brazilian signature policies require that each signature contains its own copy of certificates, revocation statuses, and time-stamps.

To solve this issue properly, we propose the *extensible signature policies*. Similarly to the existing state-of-the-art policies, extensible policies are a set of rules for creating and verifying signatures. However, extensible policies allow for applying additional and more evolved policies to a signature. For example, a *basic extensible signature policy* could permit users to add a time-stamp to the signature by applying a *dated* or *archival signature policy* if necessary. Hence, parties need not know in advance the expected validity of their signatures. Also, parties avoid the overhead of using a time-stamp when it is not necessary.

Our proposal is compatible with the signature policy standard, since we use the extensions feature of this standard. We demonstrate the efficacy of our solution by implementing signature policies and developing a prototype. This prototype is based on a signature software provided by the Brazilian Public Key Infrastructure. Moreover, we demonstrate that Brazil could employ our solution without significant impact on signature users and existing infrastructure.

The remainder of this work is organized as follows. We introduce signatures, time-stamps, and policies in Section II. In Section III, we analyze the case of signature policies in Brazil. Section IV presents our solution, the extensible signature policies. Section V describes how we implemented the extensible policies and developed a prototype. In Section VI, we evaluate our solution. Finally, we draw our conclusions and plan future work in Section VII.

II. Digital Signatures, Time-stamps, & Signature Policies

This section presents the background necessary for this work. We first introduce the parties involved when digital signatures, time-stamps, and signature policies are used. Then, we explain signatures, time-stamps, and policies.

The parties that can be involved in the use of digital signatures, time-stamps, and signature policies are called *signers*, *verifiers*, *signature policy issuers*, and *judges*. Assuming a scenario where a *signer* and a *verifier* are participating in a transaction involving digital signatures, e.g., a seller signing a receipt stating that the payment from a buyer has been received and the purchase will be delivered, the steps to create the digital signature are: i) the two parties agree on a signature policy, issued by a signature policy issuer, establishing how this signature should be created and verified; ii) the signer creates a signature according to the chosen policy; and iii) if the signature may be needed after its validity ends, the signer or verifier can apply time-stamps to it.

Considering the same scenario mentioned before, when verifier needs to check the digital signature, the verification is performed using the policy initially defined. If there are time-stamps, they are also checked. The verifier performs the verification to be convinced that the signature is a proof of the signer's commitment. When there is a dispute between a signer and a verifier, a judge may be necessary. For example, when the signer wants to repudiate his or her commitment by claiming that he or she has not created that particular signature. In this case, the judge checks whether the signature is indeed a valid signature from the repudiating party in order to decide in favor of the signer or verifier. To do this, the judge also uses the policy and time-stamps.

Digital signatures guarantee integrity, authenticity, and non-repudiation of data [9]. Integrity allows to check whether the signed data has been modified. Authenticity allows identifying who created the signature on the data. Non-repudiation prevents the originator of the signature from claiming that he or she has not generated the signature. Because of these guarantees, signatures are useful to prevent frauds and to enforce commitments. Frauds are prevented because signatures allow users to notice when data has been modified (integrity) or forged (authenticity). Commitments can be enforced because signatures prevent a user from denying that he or she has acknowledged some data by signing it (authenticity and non-repudiation).

We now explain how digital signatures work. They can be generated using asymmetric cryptography [3], which provides three algorithms. First, a signer uses the *key generation algorithm* to generate his or her key pair (k_s, k_p), where k_s is the secret signing key and k_p is the public verification key. Next, given a piece of data d, the signer uses the signing key k_s and the *signature generation algorithm* to create a signature σ on d. Finally, given the public key k_p, the signature σ, and the signed data d, a verifier uses the *signature verification algorithm* to check whether σ is a valid signature.

The public key k_p is usually distributed in the form of a public key certificate [9]. This certificate is issued by a certificate authority (CA), which authenticates the signer's identity. Moreover, a certificate is only valid for a limited period of time. During this period, the CA can revoke the certificate if needed, e.g., when the corresponding private key is compromised. After a certificate expires or is revoked, a digital signature no longer provides the non-repudiation property. This is because the verifier cannot check whether the signature was generated by the signer *before* the certificate expired or was revoked, or by a forger *after* the expiration or revocation.

Time-stamps are a well-known solution for the above issue. They are issued by trusted time-stamp authorities (TSAs) as follows. Assume a piece of data d is to be time-stamped. A TSA time-stamps d by creating signature σ on d together with the current date and time τ, i.e., τ is the moment when σ is created. The signature σ is stored together with the date and time τ in a time-stamp t.

A party can use a time-stamp to guarantee the non-repudiation of a signature even after the public certificate necessary to verify this signature is no longer valid. More precisely, *before* this certificate expires or is revoked, the party requests a time-stamp on that particular signature together with the certificate and revocation status showing that the certificate is still valid. Later, if the certificate expires or is

 ISBN: 978-1-61208-493-0

revoked, the time-stamp can be used to demonstrate that the signature *already existed* when the certificate was valid and, therefore, that the signature was created at a time when only the legitimate signer could have produced it [10]. Note that this time-stamp is not intended to demonstrate the exactly time when the signature was created, but rather to show that the signature was created at some point in time *before* the certificate containing the verification public key expired or was revoked.

However, since time-stamps also rely on signatures, they have a limited validity period. Consequently, further time-stamps are needed, yielding the time-stamp sequence found in Figure 1. In this sequence, the first time-stamp extends the validity of the document signature. The subsequent time-stamp extends the validity of the previous time-stamp signature and so on. The last time-stamp should have a valid signature at the moment the verifier checks the document signature [11].

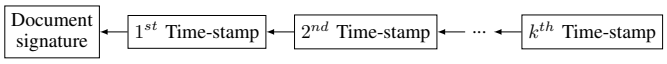

Figure 1. A document signature and a sequence of time-stamps.

Digital signatures are often distributed in a single file which includes their corresponding time-stamps. More precisely, this file contains the digital signature and the so-called *signed* and *unsigned attributes*. Signed attributes are additional data that the signer signs along with the document. For example, the certificate that verifiers must use to verify the digital signature. By contrast, unsigned attributes are not signed together with the document. Therefore, any party can append to or remove them from the digital signature without corrupting it. Examples of unsigned attributes are time-stamps and evidence showing that the certificate needed to verify the signature was not revoked (e.g., certificate revocation lists). The definition of the file containing a digital signature and the corresponding signed and unsigned attributes is provided by the so-called *Advanced Electronic Signatures* [12][13]. We will refer to this file simply as a *signature*.

Because signatures can be created using distinct signature algorithms and containing several attributes, signers and verifiers should agree beforehand how these signatures must be. To this end, signature policies have been proposed and standardized [6][7]. They are a set of rules for creating and verifying signatures. A signature policy has a unique identifier provided by the signature policy issuer. When signing a document d using a policy identified by i, signers create a digital signature σ on d together with i. Moreover, they provide σ together with i in the form of a signature (i is a signed attribute). Thus, verifiers and judges can identify the used policy and verify the signature accordingly.

III. Signature Policies in Brazil

This section describes how signature policies have been implemented by the Brazilian Public Key Infrastructure (PKI). We start by detailing the implemented signature policies. Next, we explain how they are distributed. Then, we analyze their limitations and implications.

In Brazil, people can use signatures to establish a legally binding commitment only if these signatures fulfill specific signature policies. More precisely, the Brazilian PKI created four signature policies. The policies are defined as follows.

TABLE I. Signature policies created by the Brazilian PKI.

Features	Signature Policies			
	Basic	Dated	Complete	Archival
Policy identifier	✓	✓	✓	✓
Time-stamp on the digital signature	x	✓	x	x
Complete certificate chain & revocation statuses	x	x	✓	✓
Time-stamp on the dig. signature, chain & rev. statuses	x	x	✓	x
Archiving time-stamp	x	x	x	✓
Maximal validity (years)	6	6	6	x

1) The *basic signature policy* can be used for short-term authentications, e.g., to authenticate wire transfers.
2) The *dated signature policy* can be used when the creation time of a signature is needed, e.g., to authenticate auction bids. It contains a time-stamp computed on the digital signature as an unsigned attribute.
3) The *complete signature* can be used when the signer's certificate chain and revocation statuses may not be available for verifiers. It also provides a time-stamp on the digital signature together with the chain and revocation statuses. This time-stamp prevents that the signature becomes invalid if any certificate in the chain is revoked or expires. This policy can be used, for example, to authenticate contracts.
4) A *archival signature* is used for long-term archiving, e.g., to guarantee the integrity of electronic land records. It contains the signer's certificate chain and revocation statuses, and one or more archival time-stamps. An archival time-stamp is applied on the whole content of a signature, including previous archival time-stamps.

Signatures must explicitly identify the fulfilling policy as a signed attributed. Moreover, signatures are valid for up to six years, which is the maximum validity of certificates in the Brazilian PKI. By contrast, this does not apply to archival signatures because their validities can be extended indefinitely by using further time-stamps. Table I summarizes the four presented policies.

The Brazilian PKI distributes these policies as follows. First, the policies are described in the form of machine-readable files using (i.e., using the ASN.1[14] and XML languages) following the signature policy standard [6][7]. Next, the hash of each of these files is included together with the corresponding policy identifier in a list. This list is also in the ASN.1 and XML forms and is signed by an authority. Finally, the policies and the signed list are published in a public repository. Signature softwares can verify the authenticity of the list and use signature policies with minimal user interaction.

We now analyze the limitations and implications of the presented approach. The limitations are two fold. First, the policies themselves prevent the signature verification from using any data that is not defined in the policies. For example, the basic signature policy allows using no time-stamps. Second, the policy used to create a signature cannot be later changed. This is because the digital signature fixes the identifier of the used signature policy. Changing the signature policy identifier corrupts the digital signature. It is worth to mention that these limitations apply not only to the Brazilian signature policies but also to any implementation using signature policies.

 ISBN: 978-1-61208-493-0

TABLE II. THE SIZES OF BASIC, DATED, COMPLETE, AND ARCHIVAL SIGNATURES IN KILOBYTES.

	Signature type			
	Basic	Dated	Complete	Archival
Size (kB)	2.96	5.76	18.40	18.40

The implication of using the presented approach is that the validity of signatures other than archival signatures cannot be extended. To illustrate this in practice, assume the following scenario. A seller used the *dated signature policy* to sign a receipt saying that he or she has sold some goods to a buyer. However, the goods have not been delivered and the buyer wants to open a dispute at a court. If the dated signature is about to become invalid because, for example, the time-stamp issuer's certificate will expire soon, then the buyer may have no time to submit the signature as valid evidence to the court. In this case, adding further time-stamps to extend the signature validity does not help the buyer, since the used signature policy establishes that only a single time-stamp is verified. Moreover, re-signing the receipt is of no use because the seller may not want to re-sign it or may have even passed away.

Therefore, the parties using signatures to establish commitments should know in advance the required validity for signatures or select the signature policy with the longest validity time, namely the archival signature policy. Both approaches have shortcomings. Knowing the required validity time for signatures before creating them may not be always possible. Selecting the archival signature policy to guarantee the longest signature validity is inefficient. More precisely, an archival signature must contain the signer's chain, the corresponding revocation statuses, and an archival time-stamp even if they are not necessary.

To illustrate this overhead, Table II compares the sizes of basic, dated, complete, and archival signatures soon after they are created. In the case of the archival signature, it contains only one archival time-stamp. To generate the signatures we used real certificates and time-stamps from the Brazilian PKI.

As it can be seen from the table, the overhead of using an archival signature policy to ensure the longest possible signature validity time is significant. More precisely, the archival signature is approximately 6 and 3 times as big as the basic signature and the dated signatures, respectively. Compared to the complete signature, there is no overhead.

Therefore, it would be desirable that parties first create small signatures with simple policies, e.g., the basic or dated signatures, and then add time-stamps or further data (e.g, certificate chains and revocation statuses) by using more evolved policies only when necessary. The next section shows how this can be done.

IV. EXTENSIBLE SIGNATURE POLICIES

As we have seen, signature policies have shortcomings. For instance, if a policy allows applying no time-stamps to a signature, then the validity of this signature cannot be extended beyond the lifetime of the signer's certificate. To address these shortcomings, we propose the *extended signature policies*. We first introduce our approach and then detail how parties can use it.

An extended signature policy is a policy that not only defines how a signature is created and verified, but also explicitly specifies which additional policies can be applied to this signature. These policies can be applied at any moment during the signature validity. More precisely, the additional policies allow including further information as unsigned attributes in the original signature when necessary. In this way, time-stamps in the form of unsigned attributes can be applied only when the signer's certificate is about to become invalid. Note that, when parties select an extensible policy, they agree not only on the selected policy, but also on the additional policies that the selected policy identifies.

We now detail how such policies are used. Assume that two parties participating in a transaction want to create a digital signature to establish one party's commitment. To do that, they select an extensible signature policy. Then, one party performs the signature, e.g., by signing a digital document. The signing process is the same as when using the state-of-the-art signature policies. The signer selects an extensible policy and creates a signature following the rules of the selected policy. He or she signs the document together with the identifier of the selected policy and provides this identifier as a signed attribute within the signature. Note that the only difference now is that the signer uses an extensible instead of a state-of-the-art signature policy.

When an additional operation needs to be performed over an existing signature that supports extensible policies, e.g., extending the validity of the signature, the following steps are required:

1) Select an additional policy to be applied to the signature. For example, a policy that provides certificate chains, revocation statuses, and time-stamps.
2) Check that the initial policy allows the selected policy to be applied.
3) Apply the selected policy to the signature by including the necessary information, e.g., time-stamps, as unsigned attributes in the signature.
4) Add the identifier of the selected policy as an unsigned attribute to the signature.
5) Optionally, apply a time-stamp on the signature to demonstrate when the additional policy was applied. Add this time-stamp also as an unsigned attribute to the signature.

The signature verification is also different in our approach. Now it is necessary to check not only whether the signature is valid, but also whether the additional policies applied are allowed. Assume that a signature was created using the policies $p_1, p_2, \ldots, p_k$, where p_1 is the initial policy and $p_2, \ldots, p_k$ are the additional policies. Additionally, assume that an authentic copy of these policies is provided. Then, the following steps should be executed:

1) For $j = 2, \ldots, k$, check that policy p_{j-1} allows policy p_j to be used. This is done by verifying that p_{j-1} explicitly specifies the identifier of p_j.
2) For $j = 1, \ldots, k$ verify the signature using the verification rules specified by policy p_j. For example, check that the time-stamp sequence is valid. For performance reasons, we suggest verifying the digital signature with the signature verification algorithm

 ISBN: 978-1-61208-493-0

(see Section II) only at the first iteration. The reason is that this algorithm is usually time-consuming.

V. IMPLEMENTATION

In this section, we demonstrate our solution by implementing the extensible signature policies and a prototype needed to create and verify signatures.

A. Extensible Signature Policies

We defined three new signature policies, namely, the *basic extensible signature policy*, *dated extensible signature policy*, and *complete extensible signature policy*. They are similar to the existing *basic*, *dated*, and *complete* signature policies found in the Brazilian PKI. The difference is that the new policies specify which additional policies can be applied to a signature. More specifically, the *basic extensible signature policy* allows to add a time-stamp to the signature by applying the *dated extensible signature policy* or the *archival signature policy*. The *dated extensible signature policy* permits to add the signer's certification path, revocation statuses, and a time-stamp to a signature by applying the *complete extensible* or *archival signature policies*. The *complete extensible signature policy* allows to add an archival time-stamp to the signature by using the *archival signature policy*. An oriented graph is provided in Figure 2 to illustrate the signature policies found in the Brazilian Public Key Infrastructure and our extensible signature policies. An edge from policies p to p' indicates that p' can be applied to signatures generated under p. Note that we have not exhausted the possible relations between policies in the graph.

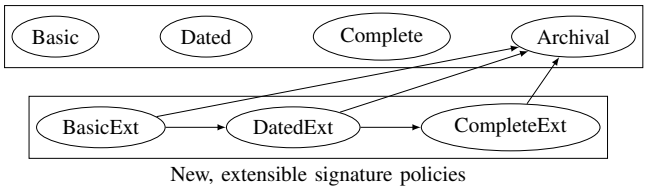

Figure 2. Existing and new, extensible signature policies.

Now we explain how to implement the extensible policies using the signature policy standard. More precisely, we describe how to specify which additional policies an extensible policy allows parties to use. Since specifying this information is a feature that was not planned in signature policies standard, we use *policy extensions* as follows. We create an extension containing two parts: i) the first part is the identifier of our extension; ii) the second is a list comprising the identifiers of the additional policies. In this list, we also include a hash computed from each additional policies in the ASN.1 or XML format. For example, in the case of the signature policy *basic extensible* (see *BasicExt* in Figure 2), this list includes the *archival signature policy*. Since this extension needs to be checked during the signature verification, we place it in the section *Verification Rules* of the signature policy standard. We present our extension in ASN.1 and XML in Figures 3 and 4, respectively, since the standard uses ASN.1 and XML.

```
AddSignPolicies ::= SEQUENCE OF
 AddSignPolicy
AddSignPolicy ::= SEQUENCE {
signPolicyIdentifier
         OBJECT IDENTIFIER,
signPolicyHash   SignPolicyHash }
```

Figure 3. The new policy extension written in ASN.1.

```
<xsd:complexType
         name="AddSignPoliciesType">
 <xsd:sequence maxOccurs="unbounded">
  <xsd:element name="AddSignPolicy"
   type="AddSignPolicyType"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AddSignPolicyType">
 <xsd:sequence>
  <xsd:element name="SignPolicyIdentifier"
   type="XAdES:ObjectIdentifierType"/>
  <xsd:element name="SignPolicyDigest"
   type="ds:DigestValueType"/>
 </xsd:sequence>
</xsd:complexType>
```

Figure 4. The new policy extension described in XML.

B. A Prototype for Creating and Verifying Signatures

We developed a prototype by extending the software provided by the Brazilian PKI for creating and verifying signatures. The original software is open-source and can be requested to the National Institute of Information Technology, the agency running the Brazilian PKI.

Our modifications to the original software were minimal. The signing process remains unchanged. The only difference now is that users can also select the new policies we introduced in Section V-A. For example, in Figure 5, we use our prototype to select the file *test_file.txt* to be signed and the XML file *basic_policy_v2.2_ext.xml* providing the *basic extensible policy*. From the XML file, the prototype extracts and shows the description of the selected policy. This description is presented in Portuguese and shows the scenarios where the signer can make use of this policy. When we click on the button *Next*, the prototype asks for the private key and signs the selected file *test_file.txt*. This is not illustrated due to space restrictions.

Most of the changes on the verification process remain hidden from the users perspective in our prototype. As in the original software, they select the signature they want to verify, and the software automatically checks the signature and provides the verification status. For example, in Figure 6, we use our prototype to select the signature found in the file *basic_signature_ext.xml*. The prototype verifies and shows that the selected signature is valid. Moreover, if additional policies had been used, then the prototype would also check whether their use is allowed. If the signature is valid and the used policies are allowed, then the user can apply an additional policy to this signature. To this end, our prototype first reads the most recent policy applied to the signature, obtains the list of allowed policies (see Section V-A), and shows this list to the user. Next, the user selects an additional policy

 ISBN: 978-1-61208-493-0

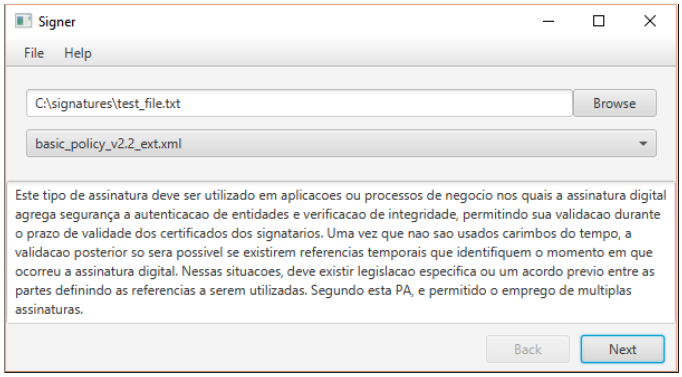

Figure 5. The prototype being used to sign a file.

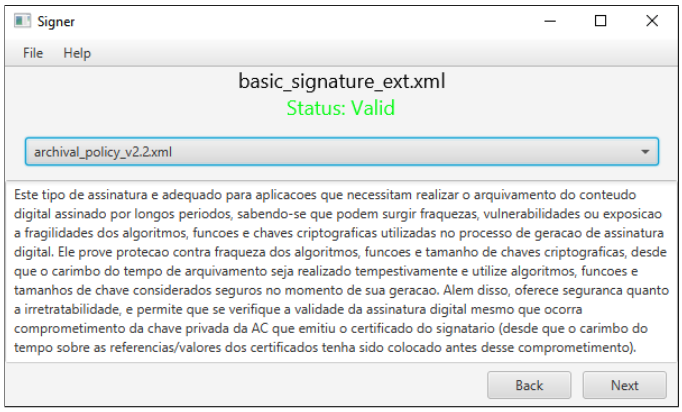

Figure 6. The prototype being used to verify and extend the validity of a signature.

from the list and clicks on the button *Next*. For example, in Figure 6 we apply the allowed policy *archival signature policy* found in the file *archival_policy_v2.2_ext.xml* to the *basic extensible signature* found in *basic_signature_ext.xml*. Finally, the prototype proceeds according to the selected policy.

VI. Evaluation

We demonstrated that the new, extensible policies are practical because the overall effort to replace the state-of-the-art policies by extensible policies is not significant for the involved parties. Our implementation used the existing infrastructure in Brazil as a study case to validate our proposal's applicability and compatibility with existing methods.

For the parties using signatures to perform commitments, deciding which signature policy to select becomes more complex in our proposal. More precisely, they should consider not only how signatures should be created and verified, but also whether they want to allow the validity of these signatures to be extended. Note that making our approach simpler by ruling out all policies that cannot be extended is not desirable. This is because in some scenarios participants may indeed want to select policies that prevent extending the validity of signatures. For example, electronic voting schemes might require that signatures used for short-term authentication expire in order to protect voters' privacy.

For a signing party, the signature process is the same when using extensible or state-of-the-art policies. Verifying parties should notice no differences between extensible and state-of-the-art policies when checking signatures with time-stamps. This is because extensible policies require more verifications than state-of-the-art policies, but these verifications consume negligible running time. More precisely, the additional verifications are checking whether an additional policy applied to the signature is allowed by the previously used policy, e.g., the initial policy employed to generate the signature. This operation consists of checking whether the identifier of the additional policy is specified by the previously used policy (see Section V-A). This simple operation should be negligible if compared to the signature verification algorithm, which consumes significant running time to verify the digital signatures on documents or time-stamps [11].

The impact on the existing infrastructure is not significant. We created the new policies by using the policy extensions feature provided by the signature policy standard (Section IV). The impact of this is that only the verification procedure of signature softwares needs to be adapted to support extensible policies. This adaption is not significant, as we showed by using our prototype in Section V. The signing procedure of these softwares remains compatible with extensible policies. Likewise, the software that policy issuers use to issue policies needs to be adapted to use our policy extension. By contrast, the procedures for signing and publishing the policies remain unchanged.

VII. Conclusion and Future Work

Digital signatures are useful to prevent frauds and create commitments between parties involved in a transaction. Since signatures can be used in several ways, e.g., by employing distinct signature algorithms, the parties involved need to agree beforehand how their signatures should be created and verified. To address this need, signature policies have been proposed and standardized. Those policies are a set of rules to create and verify signatures. To date, Brazil has adopted signature policies to regulate legally binding signatures.

Our contribution includes the analysis and improvement of signature policies in Brazil. Nevertheless, this work is not restricted to the Brazilian signature policies, but it is applicable to any scenario using the signature policies standard. Our analysis shows that signature policies can limit the practical use of signatures. The reason is that, if a signature policy allows for no time-stamps, i.e., signed evidence showing when some data existed, then users cannot extend the validity of a signature beyond the expiration or revocation of the certificates needed to verify the signature. Note that this is necessary in several scenarios, e.g., when a party wants to extend the validity of a signature in order to provide it as valid evidence to a court. Conversely, if a signature policy requires time-stamps, their use can be unnecessarily inefficient. More precisely, the signature can only be validated accordingly if it contains a time-stamp, even though this time-stamp is not needed yet because the signature will expire in the far future.

Our solution is the *extensible signature policies*. These policies allow additional policies to be applied to a signature. Therefore, parties can select a policy that fulfills their current requirements. If the requirements change in the future, e.g., time-stamps become necessary, parties can apply an additional policy that addresses the new requirements without invalidat-

 ISBN: 978-1-61208-493-0

ing the previously used policy. Verifiers check the signature according to the initial and additional policies.

Finally, we demonstrate the applicability of our proposal by implementing the new policies on the Brazilian PKI and adapting an existing signature software. More precisely, we have shown that the changes required to put our solution into practice do not cause a great impact on signature users and existing infrastructure.

Future work. We plan to allow signers to select not only an initial signature policy that can be extended, but also the rules that any additional policy should contain. For example, a signer may want that any additional signature policy applied to his or her signature prevents verifiers from using Certificate Revocation Lists when verifying his or her signature.

Moreover, further work needs to be done to allow for an archival signature policy that can be extended. This is necessary in particular situations, for example, when signatures must remain valid for several decades. Our approach cannot be used in such cases because, when creating an archival signature policy, the additional policies that can be applied to this archival signature policy may not yet exist. To address this issue, we plan to use the so-called *chameleon hash functions* [15]. These function can be used to precompute references to additional signature policies that will be created and include these references in the archival signature policy.

VIII. ACKNOWLEDGMENTS

This work has been funded by CAPES, Brazil.

REFERENCES

[1] R. van Welie, R. Willemsen, J. Abraham, and B. Nagelvoort, "Global B2C E-commerce Report 2015," Ecommerce Foundation, Tech. Rep., 2014.

[2] N. Bose and M. Potter, "Fraud rates on online transactions seen up during holidays: study," 2015, http://www.reuters.com/article/us-retail-fraud-idUSKCN0T611T20151117 [retrieved: June, 2016].

[3] W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol. 22, no. 6, 1976, pp. 644–654.

[4] Presidency of the Republic, "Interim Measure n. 2.200-2, August 24, 2001," Official Diary of the Union, aug 2001, p. 65.

[5] J. A. Buchmann, Introduction to Cryptography. Springer, 2002.

[6] ETSI, "Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies," Tech. Rep. ETSI TR 102 272, 2003.

[7] ——, "TC Security - Electronic Signatures and Infrastructures (ESI); XML format for signature policies," Tech. Rep. ETSI TR 102 038, 2002.

[8] R. S. Martini, "Signature Policies Requirements for the Brazilian PKI," Information Technology Institute, Tech. Rep. 7, 2015.

[9] C. Kaufman, R. Perlman, and M. Speciner, Network Security: Private Communication in a Public World (2nd Edition). Prentice Hall, 2002.

[10] D. Bayer, S. Haber, and W. S. Stornetta, Improving the Efficiency and Reliability of Digital Time-Stamping. New York, NY: Springer New York, 1993, pp. 329–334.

[11] M. Vigil, J. A. Buchmann, D. Cabarcas, C. Weinert, and A. Wiesmaier, "Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: A survey," Computers & Security, vol. 50, 2015, pp. 16–32.

[12] ETSI, "Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)," Tech. Rep. TS 101 733, 2012.

[13] ——, "XML Advanced Electronic Signatures (XAdES)," Tech. Rep. ETSI TS 101 903, 2009.

[14] O. Dubuisson, ASN. 1: communication between heterogeneous systems. Morgan Kaufmann, 2001.

[15] H. Krawczyk and T. Rabin, "Chameleon signatures," in Proceedings of the Network and Distributed System Security Symposium. San Diego, CA: The Internet Society, 2000.

 ISBN: 978-1-61208-493-0