International Journal on Digital Libraries manuscript No.
(will be inserted by the editor)

A Hierarchical Extraction Policy for Content Extraction Signatures

Selectively Handling Verifiable Digital Content

Laurence Bull[1], David McG. Squire[1] and Yuliang Zheng[2]

[1] School of Computer Science and Software Engineering, Monash University, 900 Dandenong Road, Caulfield, Victoria, 3145, Australia, e-mail: l.bull@csse.monash.edu.au, e-mail: David.Squire@csse.monash.edu.au

[2] Department of Software and Information Systems, University of North Carolina at Charlotte, Charlotte, NC 28223, USA, e-mail: yzheng@uncc.edu

Submitted: 31 October 2003

Correspondence to: Laurence Bull

School of Computer Science and Software Engineering

Monash University

900 Dandenong Road

Caulfield

Victoria 3145

AUSTRALIA

Phone: +61 3 9903 1950

Fax: +61 3 9903 1077

Abstract. Content Extraction Signatures (CES), enable the selective extraction of verifiable content from signed documents. Extending this ability, we introduce a new Hierarchical Grouping Extraction Policy that is more powerful and less costly than the existing Grouping Extraction Policy, and maps naturally onto the hierarchically structured documents commonly found in Digital Libraries. We also show how to implement the new Extraction Policy using XML Signatures with a custom transform. We introduce an improved design for the XML Signature structure in order to achieve CES functionality. We then conjecture as to how to enrich Digital Library functionality through the use of Content Extraction Signatures.

Key words: Content Extraction Signatures – XML Signatures – XML Signature Custom Transforms, Selective Content Disclosure – Hierarchical Extraction Policy, Privacy-Enhancing Signatures

1 Introduction

As the Internet burgeons, the fledgling electronic society emerges, thus increasing the volume of digital information. To cope with the growing flood of data, we need new ways of handling and processing information that are not just electronic analogues of what has been done in the paper-based world.

Documents are merely containers. In the paper-based world, however, the tight binding of the medium and the message makes this distinction hard to see: we tend to think, for example, of a certificate *being* a piece of paper, rather than the facts printed on it. Traditionally, and in most computerised implementations to date, this view has been perpetuated: documents have been viewed and handled as coherent collections of semantically grouped information. Some documents, however, are merely containers of facts, such as a contract, an academic transcript, a non-fiction book, or an encyclopedia. It is with the verifiability of the facts in such documents that our focus lies.

The elegant concept of public-key cryptosystems [11], and their implementation [19], enabled a content-dependent digital signature to be created for electronic documents. Beth, Frisch and Simmons [4] suggest that this changed the primary focus of the information security field from secrecy alone to broader notions of authentication, identification and integrity verification. With the steady rollout of Public Key Infrastructure (PKI), public, corporate and governmental acceptance of, and confidence in, digital signatures has steadily grown. Blakley posits that digital signatures are quite different from their ink-based predecessors, and suggests that we should "look more closely at every way in which digital signatures differ" so that we may fully realise their worth [5]. We agree.

We are specifically interested in the technical constructs and mechanisms in a digital signature that afford the ability to selectively handle verifiable content securely and efficiently. Thus Content Extraction Signatures (CES) [20] were developed to enable the signing of content at a granularity specified by the signer, rather than following the traditional practice of unconditionally signing at the container level (i.e. the whole document).

Brands has contributed extensive work towards enhancing the privacy of document owners through the use of "Digital Credentials", along with associated protocols for their use with a certification authority. This affords the selective disclosure of data fields in the credential [6, 7]. This is in contrast with Content Extraction Signatures, which do not require a certification authority.

Micali and Rivest introduced "Transitive Signature" schemes [16]; Bellare and Niven later presented performance improvements for such schemes [2]. Transitive signatures allow a signer to sign edges and nodes of a graph such that a signature for any edge in the transitive closure of the signed graph can be generated that is indistinguishable from the signature that would have been generated had the original signer signed that edge. This scheme shares with CES the notion of enabling valid signatures to be generated for transformations of an original signed object, though in this case, the signatures are for information implicit in the original, rather than subsets of it. A general approach to homomorphic signature schemes for some binary operations has been reported by Johnson, Molnar, Song and Wagner [14].

The XML Signature (XMLsig) specification [1] is a joint proposal from the World Wide Web Consortium (W3C) [21] and the Internet Engineering Taskforce (IETF) [13]. It defines a scheme for creating digital signatures that can be applied to digital content, which may located internal to the document or externally on various sites across the web. Whilst there are some similarities, or parallels, with CES, the XMLsig does not provide for the CES security for blinded content, nor does it permit a signer to specify an extraction policy.

Polivy and Tamassia [17] present an architecture for authenticating responses to queries from untrusted mirrors of authenticated dictionaries using Web Services and XML Signatures. They also implement a custom XML Signature transform. While other work by Devanbu, Gertz, Kwong *et al.* have proposed a new approach to signing XML documents to enable certification of answers to arbitrary queries [10].

1.1 Contents of this Paper

In this paper we introduce a new Hierarchical Grouping Extraction Policy for use with Content Extraction Signatures. We demonstrate its implementation using XML Signatures, and then illustrate enriched functionality for Digital Libraries through the use of Content Extraction Signatures using the new grouping policy.

Section 2 gives the reader some background by introducing Content Extraction Signatures through a brief overview, along with a motivating example concerning the selective handling of verifiable content.

A recap of our previously introduced Extraction Policies is presented in Section 3, including a detailed revisiting of the Grouping Extraction Policy. We include an example to establish a foundation and framework for the presentation of the new Hierarchical Grouping Extraction Policy. This is followed with a comparison of the Extraction Policies to assess the new scheme.

After giving a brief overview of XML Signatures, in Section 4 we show how to implement the new Hierarchical Grouping Extraction Policy and achieve CES functionality using the open standard XML Signature to enable development of interoperable applications. We also show an improved design for the XML Signature structure that enables it to handle grouping Extraction Policies.

Having shown how to selectively handle verifiable content using CES, in Section 5 we conjecture as to how this may enrich the functionality of Digital Libraries in the emergent electronic society.

We close with some concluding remarks in Section 6.

2 Background

2.1 Content Extraction Signatures

Content Extraction Signatures (CES) [20] were originally designed for use in multiparty interactions to overcome privacy concerns by enabling the selective disclosure of verifiable document content. CES permit the owner, Bob, of a document signed by a signer, Alice, to produce an "extracted signature" for an extracted subdocument (original document less some removed, or "blinded", content), which can be verified (to originate from Alice) by any third party, Carol, without knowledge of the unextracted (blinded) document content.

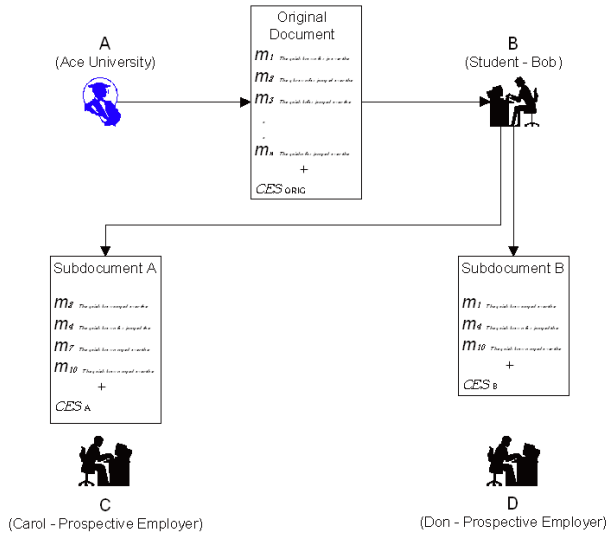

Fig. 1. A real-life scenario for selective disclosure.

To illustrate the use of CES, consider the commonplace example depicted in Figure 1, involving the document signer, Ace University, the document owner, Bob, a student, and verifiers, Carol and Don, who are potential employers. In this example, Ace University issues a

student Bob with a formal document: an Academic Transcript (original document). Bob is required to include the formal document with a job application document sent to a prospective employer Carol. Note that the Academic Transcript document is likely to include the Bob's personal details, for example his date of birth (DOB), etc. To avoid age-based discrimination, Bob might not wish to reveal his DOB to Carol (indeed, in some countries it is illegal for a prospective employer to seek the applicant's DOB). The university understands this and is willing to allow employers to verify academic transcripts with the DOB removed (and possibly with other fields agreed to by the university removed as well, but not others which the university may require to be included in any extracted document).

An essential and integral component of Content Extraction Signatures is the signer's Extraction Policy, which enables the signer to specify which fragments may be extracted, or blinded. This affords protection from semantic abuse: abuse arising from the use of the content in an out of context manner. Extraction Policy validation is a requirement for Content Extraction Signature validation.

In short, Content Extraction Signatures enable selective disclosure of verifiable content, provide security for blinded content through the use of a salt, or nonce, and enable the signer to specify the content that the document owner is allowed to extract or blind. Combined these properties give what we call *CES functionality*.

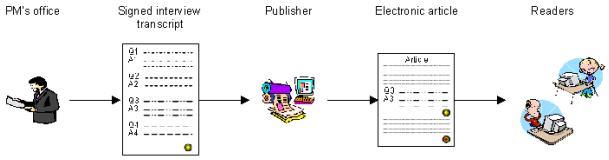

Fig. 2. Example of electronic publishing which includes verifiable content sourced from another signed document.

2.2 Bandwidth Issue

The ever "maximally" coarse granularity of signed information using the standard digital signature causes unnecessary bandwidth usage. Consider Bob, the document owner, who wants to pass on a single item of verifiable information to Carol. Instead of being able to pass this single piece of information, Bob is forced to furnish the entire document, which could be significantly greater in size than the single item, otherwise Carol will not be able to verify the signer's signature over the information.

To illustrate such a scenario, which is not a privacy issue but one of information relevance, consider an electronically published article, in which some aspect of an interview with the Prime Minister (PM) is reported. As depicted in Figure 2, the PM's office issues a transcript of the interview involving the PM, which has been signed using the standard digital signature.

The publisher would like to quote only the PM's response to a particular question as there are tight constraints on article size and it is neither appropriate, nor possible, to include the entire transcript of the interview.

It is highly desirable for the reader to be able to verify the quoted content in the article, which originates

from the signed interview transcript, as it would eliminate problems of misinterpretation and misquoting.

This example illustrates the tension that exists between verifiable content granularity and bandwidth, as illustrated in Figure 3. This tension is likely to arise in many other scenarios as the Internet burgeons. A further goal of this work is to reduce the signed content granularity and move towards reduced bandwidth.

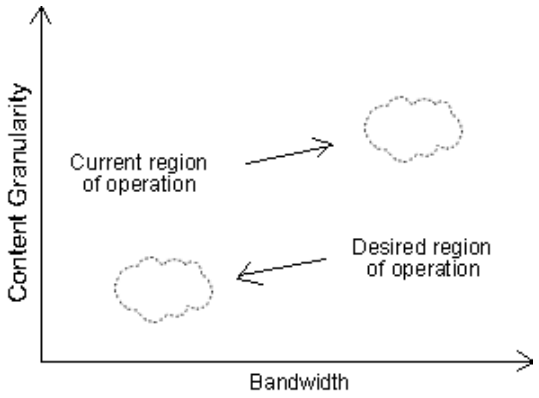

Fig. 3. Tension between verifiable content granularity and bandwidth usage.

2.3 Selective Content Disclosure Abuse

The ability to selectively disclose information contained in a document also has a potential risk, as the information accompanying a fragment in a document often provides the context. The disclosed fragment may have a different meaning when it is not accompanied with certain other information which is present in the original document.

For example, using the above scenario depicted in Figure 2, to avoid the PM's responses being quoted out of context, it is desirable that the question and the response be linked, so that the response is always preceded by the corresponding question. Hence there is a requirement that the information signer be able to exert some control over which verifiable content can be selectively disclosed by the document holder. Conceivably, the document signer would want to be able to specify which fragments can:

- be extracted in isolation,
- be extracted only when accompanied by other specified fragments,
- be extracted optionally accompanying other specified fragments, and
- never be extracted (i.e. can only ever be provided with the entire document).

The dangers involved in the selective use of information and how the meaning can be changed is illustrated with the September 2002 intelligence report from the chairman of the British Joint Intelligence Committee. The Chief of Staff to British Prime Minister, influenced the intelligence chief to omit the phrase "if he believes his regime is under threat" when discussing whether the Iraqi President was prepared to use chemical and biological weapons [18], thus changing the meaning to suggest that the weapons and their use posed an offensive threat.

It is vitally important to protect against semantic abuse when providing the ability to selective handle information. Therefore, the design of CES includes a signer-

specified Extraction Policy that enables the signer to specify precisely which content may be disclosed.

2.4 User Conceptual Models

There are notionally two modes of use: blinding fragments (extracting most of the document content and blinding some content), or extracting specific fragments (blinding most of the document and extracting only some content). Each mode reflects the perspective of the document owner and their requirements when selecting content for disclosure and represents each end of a continuum.

3 Extraction Policies

The function of the Extraction Policy is not to enforce what content is disclosed. Instead, it specifies what subdocuments are permissible and for which an *extracted CES* can be generated. The extracted CES enables the recipient of an extracted subdocument to verify the subdocument content. Thus the verification of a CES not only involves verifying the document content, it also includes checking the fragments for compliance with the Extraction Policy.

The Extraction Policy is embodied in an encoding of all the allowed fragment extraction subsets in a structure called a *Content Extraction Access Structure* (*CEAS* for short). Thus the CEAS is an integral component of Content Extraction Signatures and is included as input to the signing and verification algorithms.

3.1 Single Dimensional Policy

The single dimensional Extraction Policy and a simple structure to support it, initially proposed with CES [20], will now be recapped.

Depending on the nature of the document and the content being signed, a very simple Extraction Policy may suffice. This includes content where there are no contextual semantics and hence no need to specify fragment grouping. The fragments are simply treated individually in a binary sense as being either *mandatory* or *optional* type, where a mandatory fragment *must* be contained in the subdocument, while an optional fragment *may* be contained in the subdocument. Therefore, the Extraction Policy can be efficiently encoded using a single bit for each fragment. Thus, for a document with, say, 200 fragments, the size of the CEAS will be 200 bits.

The earlier example illustrated in Figure 1 above, involving the student forwarding a signed electronic version of his/her Academic Transcript to a prospective employer could involve a single dimensional Extraction Policy. In this example the student wants to simply blind his/her Date of Birth in the transcript.

Single dimensional Extraction Policies have very low implementation costs, but do not support fragment group-

ing, and hence, are suitable where there are no contextual semantics for the fragments.

3.2 Richer Multidimensional Policies

Now we focus on Extraction Policies that will support the ability to select and extract fragment groupings as well as the ability to specify the fragment grouping relationships as being either mandatory or optional. Thus we now have a multidimensional view of the fragment.

This presents a challenge: how do we achieve this richness and flexibility in the Extraction Policy whilst constraining the size of the CEAS, which contains the encoding of this information, and hence the size of the extraction signature? The multidimensional policies hereafter will be described with respect to the extraction conceptual model.

3.2.1 Grouping

We will now revisit in some detail the *Grouping Extraction Policy*, proposed in [8], to establish a foundation and framework for presenting a new hierarchical grouping policy along with its encoding in the CEAS.

First we will redefine our fragment types used earlier for the Single Dimensional Extraction Policy replacing the *Mandatory* and *Optional* types with *Primary* and *Secondary* targets respectively. A *primary* target fragment is allowed to be extracted in its own right from the original document to form the subdocument. Only primary targets may be directly selected, or targeted, for extraction. If a fragment is not a primary target, then it is a *secondary* target and it may only be extracted through an association with another fragment that is a primary target.

Fragment groupings are specified through the use of an association from one fragment to another fragment. A fragment may have no, or many, associations with other fragments. Each association is either *Mandatory* or *Optional* and all associations are asymmetric and transitive. Also, mandatory associations are relative to a primary target fragment and always subsume optional associations with respect to transitivity. If a fragment has a mandatory association with a primary target, it means that the associated fragment *must* accompany the primary target if it is extracted. A fragment that has an optional association with a primary target fragment *may* accompany the primary target fragment if it is extracted. Associations are mutually exclusive as a fragment cannot have both a mandatory and an optional association with another fragment.

We will now describe fragment grouping options and their use by the document owner. A fragment type and its extraction permissions can be identified as:

- a primary target with no associations—it can be extracted by itself;
- a primary target with mandatory associations—if extracted it must be accompanied by its associated mandatory fragments;

- a primary target with optional associations—if extracted it may be accompanied by its associated optional fragments;
- a primary target with mandatory associations from *all* other primary targets—a mandatory fragment which must accompany any primary fragment that is extracted;
- a secondary target with no associations—it can never be extracted;
- a secondary target with mandatory associations—can only be extracted when accompanying a primary target fragment via a mandatory association; or
- a secondary target with optional associations—it can only be extracted when accompanying a primary target fragment through an optional association.

CEAS Using Byte Lists A simple approach to storing the signer's fragment Extraction Policy is to use lists for the fragment associations. We implement for each fragment a list for either its mandatory or its optional associations.

A fragment's type is determined by whether or not its self-referent fragment number is contained in the list: primary target type if in the list, or secondary target type if not in the list.

The type of associations with the fragment numbers contained in the list are in turn determined by the fragment type: primary target lists describe mandatory associations while secondary target lists describe optional associations.

With a 32 bit fragment identifier, the size of the CEAS for a document containing 200 fragments with a fragment association density of say 20% (i.e. an average of 40 associations per fragment) and a primary target density of say 50% (i.e. 100 of all the fragments are a primary target) would be 257.92 kbits.

CEAS Using Bit Vectors Bit vectors could be used as an alternative to using lists, where for a document with n fragments, we allocate a vector of n bits for each fragment. This can be seen as a $n \times n$ bit matrix, irrespective of the number of associations. As there are n bits available per fragment, we use:

- *the self-referent bit*—to specify if the fragment is a primary target or a secondary target; and
- *the non self-referent bits (or other bits)*—to specify the mandatory or optional fragment associations, of which there are $n - 1$.

The type of association specified by the other bits depends on whether the fragment is a primary or secondary target. For primary targets the other non-self-referent bits define the mandatory associations, while for secondary targets they define the optional associations. Also, there are no optional associations between two primary fragments. This would be redundant, as you can simply extract the two primary fragments, or not, as required.

Table 1. Sample CEAS for a document with 6 fragments

Fragment no.	CEAS
1	0 0 0 0 0 0
2	0 0 0 0 0 0
3	0 0 0 0 1 0
4	0 1 0 1 0 1
5	0 0 0 0 1 1
6	0 0 0 0 0 1

A Bit Vector Example Explained A simple CEAS for a document with six fragments is illustrated in Table 1. This simple example illustrates the encoding of the various fragment types as identified above. However, it is expected that an actual Extraction Policy would likely involve a higher fragment association density. Following is an explanation of the fragment Extraction Policy for the document.

Frag1 is a secondary target and can never be extracted as no other fragments are associated with it, ie. $(CEAS_1[1] \vee \ldots \vee CEAS_n[1]) \wedge (CEAS_1[1] \vee \ldots \vee CEAS_1[n]) = F$

Frag2 is a secondary target and can only be extracted through its mandatory association with frag4. If frag4 is extracted, then frag2 must accompany it.

Frag3 is a secondary target and can only be extracted via its optional association with frag5. If frag5 is extracted, frag3 may optionally accompany it.

Frag4 is a primary target with some mandatory fragment associations that must accompany it should it be extracted. If frag4 is extracted, then frag2 and frag6 must accompany it.

Frag5 is a primary target with mandatory and optional fragment associations. Should frag5 be extracted, then frag6 must accompany it, while frag3 may optionally accompany it.

Frag6 is a primary target with no associations that must accompany it should it be extracted. Frag6 can be extracted by itself.

Frag6 is also a mandatory fragment, which must always be extracted, as every primary target has a mandatory association with it, ie.

$b_1 \wedge b_2 \wedge \ldots b_n = T$

where $b_i = \neg CEAS_i[i] \vee CEAS_i[6]$ and i indexes the fragments.

As the bit matrix hints, the CEAS is in fact a labelled directed graph, the matrix in Table 1 corresponding to the connectivity matrix. The node labels indicate fragment identity, and edges represent associations. Primary targets are represented by nodes that are connected to themselves. Nodes corresponding to primary targets have edges directed to the nodes with which they have mandatory associations. Nodes corresponding to secondary targets have edges directed to nodes with which they have optional associations.

Practical Example To illustrate a scenario where a Grouping Extraction Policy would be used, consider the electronic publishing example discussed earlier in §2.2. In

this case the Prime Minister's response to a particular question could be defined as a primary fragment with a mandatory association specified for the preceding question. If the response fragment was extracted, then the preceding question fragment must also accompany it for the extracted signature to able to be verified. Alternatively, the question fragment could be specified as a primary fragment with an optional association to the response fragment, which would be specified as a secondary fragment. In this case, the response fragment could not be directly targeted for extraction. However, it could optionally accompany the question fragment.

Lists versus Vectors List-based representations are more efficient when fragment association density (i.e. edges per node) is low, particularly for large numbers of fragments. The bit matrix will be the more efficient when the association density is high.

Recall that n was defined as the number of fragments in a document. We now define s to be the size of the fragment identifier in bits, a_d the fragment association density and p_d be the primary fragment density. The size of the list encoding in bits is

$$ns(n-1)a_d + nsp_d \tag{1}$$

while the matrix encoding is

$$n^2 \tag{2}$$

The matrix encoding will thus be the more efficient when

$$ns(n-1)a_d + nsp_d > n^2 \tag{3}$$

that is, when

$$a_d > \frac{n}{s(n-1)} - \frac{p_d}{(n-1)} \tag{4}$$

For all sufficiently large n, this reduces to

$$a_d > \frac{1}{s} \tag{5}$$

therefore, when using a fragment identifier size of 32bits, the matrix encoding will be more efficient when the fragment association density is greater than approximately 3%.

Thus, for comparison with the example in §3.2.1 above, also with 200 fragments, the matrix representation would cost 40 kbits.

3.2.2 Hierarchical Grouping

Whilst the *Grouping Extraction Policy* described in the previous section supports the grouping of fragments it does not permit the sub-grouping of fragments. Nor does it seem ideal for use with signing hierarchical documents that have hyperlinks such as web pages or more generally XML documents. We will now present and discuss a new Extraction Policy suitable for such use: a *Hierarchical Grouping Policy*.

The same basic concepts and definitions for fragment types and their associations as defined for a *Grouping Extraction Policy* are retained although we introduce a notion of locality, or scope. We will adjust the definitions for fragment type and associations, as well as introduce some restrictions for their use within a locality.

Let us consider a fragment of content, in this case comprised of three paragraphs of text. This fragment can be divided into three segments called sub-fragments, or child fragments, as illustrated in Figure 4. Extending further, each of the child fragments could in turn be divided into segments, or sub-fragments, and so forth until the desired content granularity is achieved. From the child fragment's perspective, its parent fragment is the most immediate fragment that minimally contains all of the content for that child fragment. The child fragment's content is also part of the content for all of its ancestor fragments.

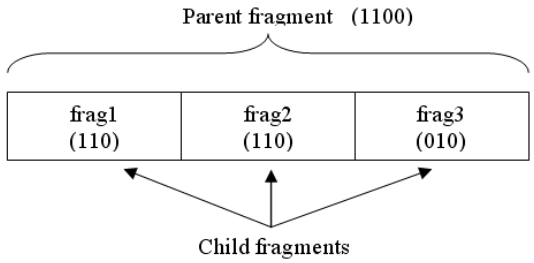

Fig. 4. Example of a parent fragment along with its Extraction Policy which has been segmented into three child fragments each with their own Extraction Policy.

The child fragment's type and associations are now handled relative to its locality and are as follows:

- a child fragment's type can be either primary or secondary target;
- a child fragment's associations are only relative to its sibling fragments;
- child fragments as a collection inherit their parent's type; and
- child fragments as a collection inherit their parent's associations.

Sub-fragments can only be associated with other fragments, which are not sibling fragments, through their parent's associations with the other fragments.

Parent fragments that are secondary targets and have no associations with other fragments, cannot have any child fragments. This is because the parent fragment can never be disclosed in a sub-document. Therefore, there is no need to define child fragments since, as a collection, they will never be able to be disclosed as they inherit the parent's type and associations.

In other words, the child fragment's type and associations are first applied with respect to all the child fragments of the parent fragment (i.e. within the scope of the parent fragment). Once this is complete, the collection of child fragments is then treated as a single item inheriting the parent fragment's type and associations. In turn where the hierarchy extends to multiple levels, the parent node is treated along with its siblings in the same manner, repeating until the root fragment is reached.

Using this scheme, a collection of child fragments can be handled selectively. Alternatively, all of the child fragments can be handled collectively, treated as a single, albeit larger, fragment if required.

CEAS Using Byte Lists The simple approach described in §3.2.1 using lists is still applicable for storing the signer's fragment Extraction Policy. However the notion

of locality, or scope, is applied so that all fragment numbering with respect to the self-referent fragment number and fragment associations is relative to the child fragments of each parent fragment. Where there are multiple levels of sub-fragments, each parent fragment is in turn treated as a child fragment of its parent and so forth until the root fragment is reached.

CEAS Using Bit Vectors We use the same scheme detailed earlier in §3.2.1 for the Grouping Extraction Policy, however, we now use it in conjunction with a notion of locality, or scope. Fragment numbering and fragment associations are treated the same as described above for byte lists.

Each fragment's vector size now changes from a fixed size of n bits for n fragments, to a varying size dependent on the number of sibling fragments it has. This means that the fragment vector size is not constant throughout the document, although it will be constant for each locality, or collection of fragment child fragments.

A Bit Vector Example Explained To illustrate the use a hierarchical grouping policy, consider a relatively simple document and its Extraction Policy encoding using bit vectors as denoted by the accompanying CEAS depicted in Figure 5. This document has four main fragments, or highest level fragments, with two of these fragments each segmented into three child fragments, or sub-fragments. Following is an interpretation of the signer's Extraction Policy for the document depicted in Figure 5.

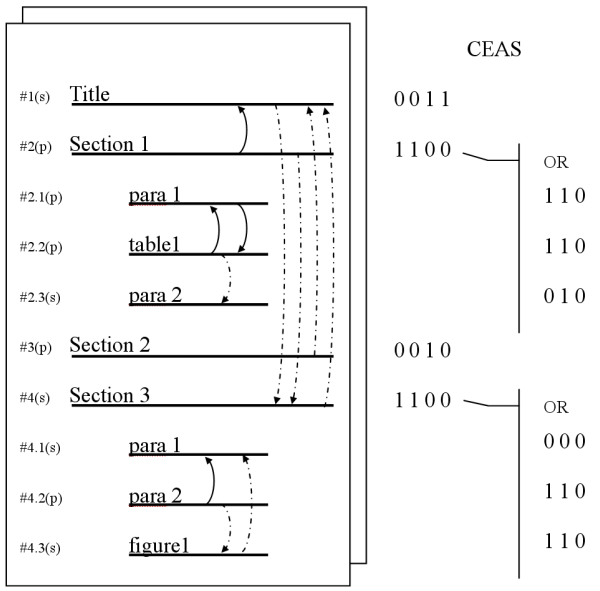

Fig. 5. Example of Hierarchical Grouping Policy encoded using bit vectors and its mapping to a structured document showing four top level fragments, two of which are parent fragments each with three child fragments.

Frag1 is a secondary target and can never be directly targeted for extraction. It can only be indirectly extracted through its mandatory association with frag2, or through its optional association with frag4.

Frag2 is a primary target that can be directly extracted. It also has a mandatory association with frag1 and an optional association with frag4. If frag2 is extracted, then it must be accompanied with frag1 and it may be accompanied with frag4.

Frag2 is also a parent fragment as it has been segmented into three child fragments: frag2.1, frag2.2 and frag2.3. Frag2 can be handled as a single fragment, which includes all of the child fragments, or as a collection of child fragments respecting the lo-

cal fragment Extraction Policy. The local policy is as follows:

Frag2.1 is a primary target with a mandatory association with frag2.2. If frag2.1 is extracted, then it must be accompanied with frag2.2.

Frag2.2 is a primary target with a mandatory association with frag2.1 and an optional association with frag2.3. If frag2.2 is extracted, it must be accompanied with frag2.1 and it may be accompanied with frag2.3.

Frag2.3 is a secondary target and can never be directly targeted for extraction. It can only be extracted through its optional association with frag2.2.

Frag3 is a primary target and if it is extracted, it may be accompanied with frag1 through its optional association.

Frag4 is a secondary target and can only accompany frag2, if it is extracted, through its optional association. Should frag4 accompany frag2, then it may also include frag1 through its optional association.

Frag4 is also a parent fragment. The local policy for handling the child fragments is as follows:

Frag4.1 is a secondary target fragment and can only be extracted by accompanying frag4.2 through its mandatory association, or it may accompany frag4.3 through its optional association.

Frag4.2 is a primary target fragment and must be accompanied with frag4.1, while it may also be accompanied with frag4.3 through its optional association.

Frag4.3 cannot be directly extracted, as it is a secondary target, although it may accompany frag4.2.

A Practical Example Considering the document depicted in Figure 5 as a journal article, a user may have a need for all of the material in Section 1 of the paper. As Section 1 is contained in frag2, the user thus extracts frag2, which also includes child fragments 2.1, 2.2 and 2.3, along with a corresponding extracted CES so that the content can be verified. However, frag2 has a mandatory association with frag1 (the title), therefore frag1 is also extracted to comply with the signer's Extraction Policy. Thereby enabling the sub-document and the extracted CES to be verified.

Another user may simply require the information contained in Table 1, which is contained in frag2.2 of the paper. Frag2.2 is a primary target and is accordingly extracted along with frag2.1 due to its mandatory association. This association may have been specified as a mandatory association due to its discussion of the contents in the table. We don't want frag2.3 so we can ignore it since it is an optional association. Once the fragment Extraction Policy for the child locality has been respected, the parent's associations and type can be applied. This means that frag1 must accompany the extracted child fragments resulting in the extraction of three fragments: frag1, frag2.1 and frag2.2.

Lists versus Vectors To compare the size of each encoding scheme we will now consider a document comprised of a shallow fragment structure similar to that depicted in Figure 5 with an increased number of fragments. The document has 150 fragments and an Extraction Policy with the following characteristics:

- 30 top level fragments;
  - $66.\dot{6}\%$ are parent fragments,
  - 50% primary target density, and
  - 20% fragment association density.
- each parent has 6 child fragments;
  - 50% primary target density, and
  - 50% fragment association density.

The size of the CEAS using the bit vector encoding scheme is as follows:

$$1.62\text{kbits} = 30^2 + 20 * 6^2 \tag{6}$$

The size of the CEAS for the list encoding scheme, allowing 32 bits for the fragment identifier, is as follows:

$$\begin{aligned} 17.568\text{kbits} = {} & 32 * (30 * .5 + 30 * (30 - 1) * .2 \\ & + 120 * .5 + 120 * (6 - 1) * .5) \end{aligned} \tag{7}$$

From this relatively straightforward example it can be seen that there is a significant difference between the costs of the two CEAS encoding schemes. This difference is apparent with the example containing just two levels of hierarchy: the difference increases as the hierarchy grows deeper.

For a more general consideration of size we define as follows:

s - size of fragment identifier

n - number of fragments for generation i

ρ - parent density for generation i, i.e. percentage of fragments for generation i that have child fragments

p_d - primary target density for generation i

a_d - fragment association density for generation i

ϕ - average number of child fragments per parent for generation i

k - total number of generations

For the Bit Vector scheme the size is as follows:

$$n_0^2 + \sum_{i=1}^{k} (n_{i-1}\rho_{i-1}\phi_i^2) \tag{8}$$

For the Byte List scheme the size is comprised of the following components:

$$\text{Sizeof Parent Primary Targets} = n_0 p_{d0} \tag{9}$$

$$\text{Sizeof Parent Frag Asns} = n_0(n_0 - 1)a_{d0} \tag{10}$$

regressively including the following generations:

$$\text{Sizeof Child Primary Targets} = (n_{i-1}\rho_{i-1}\phi_i)p_{di} \tag{11}$$

$$\text{Sizeof Child Frag Asns} = (n_{i-1}\rho_{i-1}\phi_i)(\phi_i - 1)a_{di} \tag{12}$$

for a total size of:

$$\begin{aligned} & s(n_0 p_{d0} + n_0(n_0 - 1)a_{d0} \\ + & \sum_{i=1}^{k} (n_{i-1}\rho_{i-1}\phi_i)p_{di} \\ + & (n_{i-1}\rho_{i-1}\phi_i)(\phi_i - 1)a_{di}) \end{aligned} \tag{13}$$

In summary, the new Hierarchical Grouping Extraction Policy more closely matches the commonly encountered organisation of structured documents. It enables

Table 2. Comparison of CEAS encoding scheme sizes for each of the Extraction Policies. Superscripts (n) indicate derivation using Equation n.

CEAS Encoding	Single Dim.	Grouping	Hierarchical Grouping
Byte List (kbits)	-	$145.44^{(1)}$	$17.57^{(13)}$
Bit Vector (kbits)	0.15	$22.5^{(2)}$	$1.62^{(8)}$

the grouping of fragments so that the fragments in the group can be efficiently handled either as an entire set, or as allowed subsets.

3.3 Comparison of Extraction Policies

Recall that the list-based encoding was shown, in Equation 5, to be more efficient than the bit vector approach for the *Grouping Extraction Policy* for low fragment association densities. This is also the case for the *Hierarchical Grouping Extraction Policy*, although the fragment association densities need to be much lower, particularly with documents that have many levels of hierarchy. As can be observed from Table 2 the *Hierarchical Grouping Extraction Policy* is significantly more efficient than the *Grouping Extraction Policy*.

3.4 Signing the Document

Signing the document using Content Extraction Signatures includes specifying the Extraction Policy, which involves a two step process: (i) define the fragments, and then (ii) specify the fragment associations. The process of defining a fragment includes specifying the content itself as well as whether it is a primary or secondary target. Once the fragments have all been defined, the signer specifies the mandatory and optional fragment associations for each fragment. This information is included as part of the extraction signature. On completion of signing, the document and its extraction signature (if separate to the document) are forwarded to the document user.

4 Implementation Using XML Signatures

Content Extraction Signatures enable selective disclosure of verifiable content, provide privacy for blinded content through the use of a salt, and enable the signer to specify the content the document owner is allowed to extract or blind. Combined, these properties give what we call *CES functionality*.

To enable the development of interoperable applications using Content Extraction Signatures with the new Hierarchical Grouping Policy we will now show how to implement XML Signatures to achieve *CES Functionality*. This is achieved through the use of a new enhanced custom transform and a redesigned XMLsig structure first introduced in [9].

4.1 XML Signatures in Brief

Basically, an XMLsig is comprised of four main components or elements: `<SignedInfo>`, `<SignatureValue>`, `<KeyInfo>` and `<Object>`. The `<SignedInfo>` element

includes all of the content or resources to be signed with each item having a corresponding <Reference> element, which identifies the content and a digest over it. The <Reference> elements are digested and cryptographically signed in a manner similar to signing when using a standard digital signature. The resulting signature value is stored in the <SignatureValue> element. The <KeyInfo> and <Object> elements are optional.

An XMLsig has the <Signature> element as the root element for its XML tree. It contains the four main components and has the following generic structure as defined in the specification [1]:

```
<Signature>
  <SignedInfo>
    <CanonicalizationMethod />
    <SignatureMethod />
    (<Reference>
      (<Transforms>)?
      <DigestMethod>
      <DigestValue>
    </Reference>)+
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo>)?
  (<Object>)*
</Signature>
```

where: ? denotes zero or one occurrences,

* denotes zero or more occurrences, and

+ denotes one or more occurrences.

4.1.1 The Reference Processing Model

The signed content, which may be contained in the same document as the XMLsig and/or external to the document containing the XMLsig, is referenced with a <Reference> element. The URI (Uniform Resource Identifier) [3] attribute of the <Reference> element identifies the signed item. Each <Reference> element may have zero or more transforms, which are applied to the dereferenced content prior to its being digested using the algorithm specified in the <DigestMethod> element. The resulting digest is always base64 encoded [12] and stored in the <DigestValue> element.

The <Transforms> element may contain an ordered list of transforms to be applied to the dereferenced content. Each transform is specified using a <Transform> element as follows:

```
<Transforms>
  <Transform Algorithm="t1" />
  <Transform Algorithm="t2" />
  ...
  <Transform Algorithm="tn" />
</Transforms>
```

The XMLsig's Reference Processing Model [1, §4.3.3.2] specifies that the dereferenced content is supplied to the first transform. As illustrated in Figure 6, the list of

Fig. 6. Transform chain for processing content prior to input to digest algorithm.

Adapted from [15, p.720]

transforms forms a transform chain where the output from the first transform is supplied as the input to the second transform, its output to the next, and so forth, until the last transform, the output of which is supplied to the digest algorithm. The types of transforms defined include: Canonicalization (with comments and without comments); Base64; XPath Filtering; XSLT; and Enveloped Signature transform. The XMLsig Reference Processing Model is also used for XMLsig Reference Validation [1, §3.2.1], which is a required part of XMLsig Core Validation.

4.2 XML Signature Design

As part of achieving CES functionality, compliance with the signer's Extraction Policy needs to be included into the XMLsig Core Validation [1, §3.2] processing requirements. This has been demonstrated in [9], however this was only with a simple, single-dimensional Extraction Policy. The policy checking mechanism uses the Reference Processing model and is inserted into the `<Reference>` element being processed. Using this approach has the limitation that as the transform chain is executed it proceeds within a scope that is relative (and hence limited) to the current `<Reference>` element being processed. The problem with this is that to handle fragment grouping, the *VerifyPolicy* transform needs to access other `<Reference>` element contents, which are effectively out of scope.

To solve this problem, the XMLsig needs to be restructured to enable the *VerifyPolicy* transform to access all of the fragment nodes. This can be achieved by making all of the `<Fragment>` elements children to the `<Object>` element and using a single `<Reference>` element to refer to the `<Object>` elements as follows:[1]

```
<Reference URI="#obj1" Type="...#Object">
  <Transforms>
    <Transform Algorithm="...ces#VerifyPolicy"
/>
  </Transforms>
</Reference>
```

The `<Object>` element contains a `<Fragment>` element for each item that is to be signed as follows:

```
<Object Id="obj1">
  <Fragment Id="frag1" URI="...">
    <CEAS type="LIST|VECTOR"> ... <CEAS>
    [<Salt> | <Digest>]
  </Fragment>
  <Fragment Id="frag2" URI="...">
```

[1] Prefixes such as http://pm.gov.au/transforms/ have been omitted throughout for presentation and security reasons as they are not germane to the examples.

```
    <CEAS type="LIST|VECTOR"> ... <CEAS>
    [<Salt> | <Digest>]
  </Fragment>
  ...
</Object>
```

where: | denotes an exclusive OR.

The URI attribute of the fragment references the fragment content while the CEAS element contains the encoding of the signer's Extraction Policy for that fragment. The `<Salt>` element contains a salt value used in CES to ensure privacy of blinded content [20, §3.3]. It is appended to the fragment content prior to digesting. The `<Salt>` element is always present in the original signature from the document signer.

When Bob, the document user, produces a subdocument, an extracted signature corresponding to the subdocument must be generated so that it can be validated by Carol, the subdocument recipient (or verifier), as being signed by Alice. This extracted signature has the `<Salt>` element replaced with a `<Digest>` element for the corresponding fragments which are not included (blinded) in the subdocument. The digest value is generated from the fragment content with the salt appended. Therefore, the extracted signature, which is generated for the subdocument, has a `<Salt>` or `<Digest>` element for each fragment that is present or has been blinded respectively.

4.3 Custom Extraction Policy Transform

The custom transform to verify the Extraction Policy used in [9] needs to be enhanced to handle the Hierarchical Grouping Extraction Policy. The URIs of custom transforms can be signed, as can the transform code itself, thus establishing trust. The requirement for the custom transform is to process the `<Reference>` element's dereferenced content by dereferencing the content of the `<Fragment>` elements and checking compliance with the Extraction Policy, and finally emitting a Result byte stream for input to the digest algorithm.

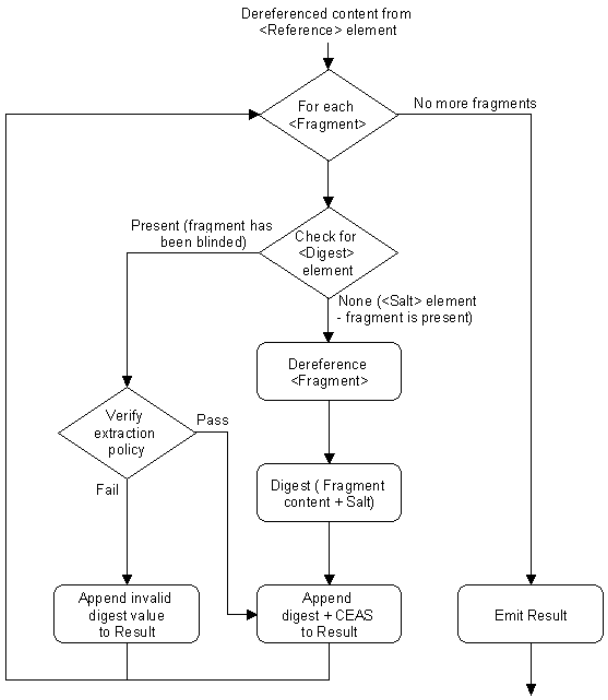

Fig. 7. *VerifyPolicy* transform algorithm for fragment grouping.

As illustrated in Figure 7, the transform processes the dereferenced content from the `<Reference>` element, which will be XML content containing at least one `<Fragment>` element. For each `<Fragment>` element

the transform first checks for the presence of a `<Digest>` element, which indicates that the fragment has been blinded. If the fragment has been blinded, then the CEAS is checked for compliance with the fragment Extraction Policy. Compliance sees the CEAS appended to the digest value, which is then appended to the Result. This Result will be emitted upon completion of processing of the last `<Fragment>` element. Should verification of the Extraction Policy fail, then two bytes of zeroes will be appended to the Result in place of the digest value. This will ultimately cause reference validation failure and in turn core validation failure as the appended bytes will not match those originally used to create the digest value stored in the reference's `<DigestValue>` element when it was signed.

On the other hand, if the fragment has not been blinded, the `<Digest>` element will not be found. Rather, a `<Salt>` element will be present. The fragment URI is dereferenced to retrieve the fragment content and the salt value from the `<Salt>` element is appended to the fragment content prior to digesting. The resulting digest has the CEAS appended to it and is then appended to the Result which will be emitted.

In addition to the explicit requirements of the transform, it also accommodates the mutation of the `<Fragment>` elements, i.e. present fragments to blinded fragments. Normally the content referenced by a `<Reference>` element is invariant and a digest over it is included in the content signed by the cryptographic signature.

5 Enriching Digital Library Functionality

Having demonstrated the technical feasibility of selectively handling verifiable information and showing how to implement the Grouping Extraction Policy for CES using the XML Signature open standard, we would now like to present some conjecture about its use with digital libraries in a future electronic society.

Digital libraries today often embrace a commercial model whereby articles, books etc. are available through various mechanisms such as subscriptions or ad hoc purchase. This information is handled at a container level where the entire container must be purchased as the user cannot simply purchase a page, or a section, from the paper. In addition, the information is not commonly signed so that the receiver can verify the contents and authenticate its source. The ability to verify and authenticate information back to its source is important these days as anybody can publish through web pages bypassing the traditional editorial/publishing process. If the user who purchases an article wants to use some of the content in a document of their own, there is little alternative to copying the content and then pasting it into the document (assuming appropriate format) as well as entering the citation information.

Ideally, the user should be able to purchase and work with just the information they require. This information should be signed so that a reader of the work can verify and authenticate the content.

In the case of a digital library, a user should be able to retrieve either the signed collection of fragments, i.e. the entire article or book, or a signed subset of fragments. If the entire collection of signed fragments is retrieved, then the user should be able to extract fragments at a latter time as required. The extracted fragments should be able to be verified, authenticated and embedded into another document. Accompanying the extracted fragments should be metadata that can be automatically used to add an entry into the bibliography if one is in use.

5.1 A Specific Example

Using a commonplace example from academia we now briefly illustrate a facet of our conjecture, ignoring any economic model that is likely to accompany an actual deployment.

Consider a researcher who is writing a paper and wishes to cite some other person's work, published and stored in a digital library, in support of some aspect of the paper. The material to be cited is contained in a published paper that has been signed by the publisher using a Content Extraction Signature. With a suitable application, the publisher makes the entire paper, or fragments thereof, available for download. In this case the researcher selects the required content and extracts it along with an extracted CES. The extraction process is inexpensive in CPU terms as it does not include any cryptographic signing. If the researcher has a local copy of the paper, the extraction is simply performed locally.

The content fragments along with the extracted CES are embedded into the researcher's paper. The in-text citation is coloured either:

- green to indicate the content, for which it anchors, has been verified,
- red to indicate the anchored content has failed verification, or
- black to indicate that verification has not yet been performed.

In addition, hovering the mouse pointer over the in-text citation displays, through a pop-up window, the content to which the citation refers, for the convenience of the reader. The embedding process also automatically inserts an entry into the list of references at the end of the document using the metadata that accompanies the embedded fragments.

The embedding of the content fragments from the referenced document into the researcher's paper makes the specific content, not the entire document, immediately available to the reader. The reader can have a high degree of confidence about the referenced material as the content is protected by a digital signature and upon verification can be certain that it has not been altered. In addition, the source is authenticated by the digital signature, thus enabling the reader to determine the veracity of the referenced content through the authority and reputation of its source.

This example represents just one possibility arising from the ability to selectively handle verifiable information in an electronic society. There may exist many other scenarios such as: web portals that aggregate information from multiple sources; or multiparty business interactions/transactions where the minimal disclosure of information to various parties is required, etc.

6 Conclusion

We have shown the tension between verifiable content granularity and bandwidth usage, along with the importance of protecting against selective disclosure abuse, or semantic abuse. Responding to these types of emerging needs, Content Extraction Signatures enable content to be signed in a finer-grained manner. We also demonstrate an extraction policy that specifies the content that can be verified when selectively disclosed.

After revisiting previous work on Extraction Policies to establish a framework upon which to build, we presented a new, richer and more efficient policy called a *Hierarchical Grouping Policy*. The new Extraction Policy is particularly suited for use with hierarchical documents such as journals, journal articles, and encyclopædias—not to mention the HTML, and increasingly XML, documents almost ubiquitous in modern electronic repositories.

We then showed how to implement CES with the new Extraction Policy using XML Signatures, along with a new custom transform and improved XML Signature structure to handle grouping Extraction Policies.

After establishing the technical feasibility of handling verifiable content in a fine-grained manner, we offered an example of its potential use to enhance the functionality of Digital Libraries in an emergent electronic society.

Acknowledgements. The authors would like to thank Mr. Gerry Butler for his valuable comments and discussion during the development of this work.

References

1. M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon. XML-signature syntax and processing. In D. Eastlake, J. Reagle, and D. Solo, editors, *W3C Recommendation*. February 12 2002. [Last accessed: September 18, 2002].
2. Mihir Bellare and Gregory Neven. Transitive signatures based on factoring and RSA. In Yuliang Zheng, editor, *Proceedings of The 8th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2002)*, volume 2501 of *Lecture Notes in Computer Science*, pages 397–414. Springer, December 2003.
3. T. Berners-Lee, R. Fielding, and L. Masinter. RFC 2396. uniform resource identifiers (URI): Generic syntax. Available online, August 1998. [Last accessed: September 25, 2002].
4. T. Beth, M. Frisch, and G.J. Simmons, editors. *Public-Key Cryptography: State of the Art and Future Directions*, volume 578 of *Lecture Notes in Computer Science*.

Springer, July 1992. E.I.S.S. Workshop Oberwolfach Final Report.

5. G.R. Blakley. Twenty years of cryptography in the open literature. In *Proceedings of 1999 IEEE Symposium on Security and Privacy*, pages 106–7. IEEE Computer Society, May 1999.

6. S.A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, MA, 2000.

7. Stefan Brands. A technical overview of digital credentials. Available online, February 20 2002. [Last accessed: February 18, 2003].

8. Laurence Bull, David McG. Squire, Jan Newmarch, and Yuliang Zheng. Grouping verifiable content for selective disclosure. In *Proceedings of The Eighth Australasian Conference on Information Security and Privacy (ACISP 2003)*, volume 2727 of *Lecture Notes in Computer Science*, pages 1–12, Wollongong, Australia, 9–11 July 2003. Springer.

9. Laurence Bull, Peter Stanski, and David McG. Squire. Content extraction signatures using XML digital signatures and custom transforms on-demand. In *Proceedings of The 12th International World Wide Web Conference (WWW2003)*, pages 170–7, Budapest, Hungary, 20–24 May 2003. ACM Press.

10. Premkumar T. Devanbu, Michael Gertz, April Kwong, Chip Martel, G. Nuckolls, and Stuart G. Stubblebine. Flexible authentication of XML documents. In *ACM Conference on Computer and Communications Security*, pages 136–45, 2001.

11. W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–54, 1976.

12. N. Freed and N. Borenstein. Multipurpose internet mail extensions (MIME) part one: Format of internet message bodies. Available online, 1996. [Last accessed: October 16, 2002].

13. IETF. The Internet Engineering Task Force. Available online. [Last accessed: October 24, 2003].

14. R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In *Proceedings of the RSA Security Conference Cryptographers Track*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–62. Springer, February 2002.

15. B. LaMacchia, S. Lange, M. Lyons, R. Martin, and K. Price. *.NET Framework Security.* Addison-Wesley, Boston, MA, 2002.

16. Silvio Micali and Ronald L. Rivest. Transitive signature schemes. In B. Preneel, editor, *Proceedings of The Cryptographer's Track at the RSA Conference (CT-RSA 2002)*, volume 2271 of *Lecture Notes in Computer Science*, pages 236–243. Springer, December 2002.

17. Daniel J. Polivy and Roberto Tamassia. Authenticating distributed data using web services and XML signatures. In *Proceedings of the 2002 ACM workshop on XML security (XMLSEC-02)*, pages 80–89, New York, November 22 2002. ACM Press.

18. Reuters. Dossier phrase dropped after blair aide email. Available online, September 2003. [Last accessed: September 24, 2003].

19. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosys-

tems. *Communications of the ACM*, 21(2):120–8, 1978.

20. R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. In *Proceedings of The 4th International Conference on Information Security and Cryptology (ICISC 2001)*, volume 2288 of *Lecture Notes in Computer Science*, pages 285–304. Springer, December 2001.
21. W3C. The World Wide Web Consortium. Available online. [Last accessed: October 24, 2003].