(IACR ePrint 2002/016)

Content Extraction Signatures[*]

Ron Steinfeld
School of Network Computing, Monash University,
Frankston 3199 Australia
ron.steinfeld@infotech.monash.edu.au

Laurence Bull
School of Computer Science and Software Engineering, Monash University,
Caulfield East 3145 Australia
laurence.bull@infotech.monash.edu.au

Yuliang Zheng
Dept. Software and Info. Systems,
University of North Carolina at Charlotte, Charlotte, NC 28223
yzheng@uncc.edu

February 20, 2002

Abstract

Motivated by emerging needs in online interactions, we define a new type of digital signature called a 'Content Extraction Signature' (CES). A CES allows the owner, Bob, of a document signed by Alice, to produce an 'extracted signature' on selected extracted portions of the original document, which can be verified to originate from Alice by any third party Cathy, while hiding the unextracted (removed) document portions. The new signature therefore achieves verifiable content extraction with minimal multi-party interaction. We specify desirable functional and security requirements for a CES (including an efficiency requirement: a CES should be more efficient in either computation or communication than the simple multiple signature solution). We propose and analyze four CES constructions which are provably secure with respect to known cryptographic assumptions and compare their performance characteristics.

Key Words. Content-extraction, privacy, fragment-extraction, content blinding, fact verification, content verification, digital signatures, provable security.

1 Introduction

During the course of a person's lifetime one has the need to use 'formal documents' such as: Birth Certificates; Marriage Certificates; Property Deeds; and Academic Transcripts etc. Such documents are issued by some recognized and trusted authority and are thereafter used by the owner in dealings with other people or organizations, to prove the truth of certain statements (based on the trust of the verifier in the issuing authority).

In an electronic world, digital signatures, together with a Public Key Infrastructure (PKI), can be used to ensure authenticity and integrity of electronic versions of formal documents. However, as digital signatures become more widely used, situations can arise where their use significantly increases the cost of desirable document processing operations, which do *not* constitute a security breach. In other words, the standard use of digital signatures sometimes places additional

[*] This is an updated version of a paper presented at ICISC 2001, Dec. 6-7 2001, Seoul, South Korea. The full version of this paper is under preparation.

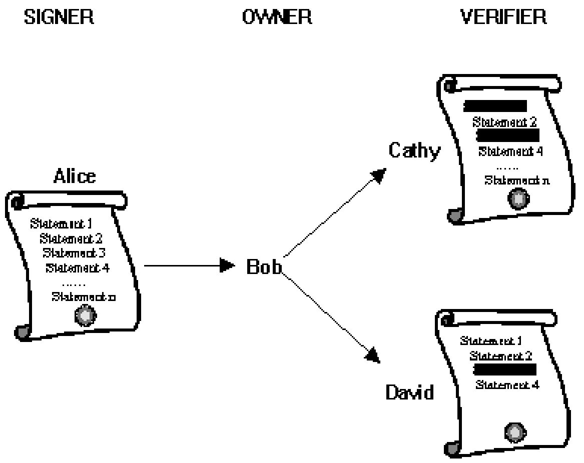

Figure 1: An example of a multiparty interaction.

constraints on *legitimate* users beyond what is really required to prevent forgeries by *illegitimate* users.

In this paper we consider one particular document processing operation which is made costly by the use of signatures, namely the *extraction* of certain selected portions of a signed document. That is, we consider the case when Bob, the owner of a document which was signed by Alice, does not wish to pass on the whole document to a third (verifying) party Cathy or David. Instead, Bob extracts a portion of the document and sends only that portion to Cathy or David as illustrated in Fig. 1. We are assuming here (and will give motivating examples later) that Alice is agreeable for Bob to perform such an extraction. Still, Cathy and David would like to verify that Alice is the originator of the document portion they received from Bob.

Ignoring for now our other requirements, we explain the difficulty with standard uses of digital signatures in this context. It is clear that if Alice simply signs the whole document using a standard digital signature, then Cathy and David will not be able to verify it unless Bob sends the whole document. The simplest solution is for Alice to divide the message into (say n) fragments and sign each fragment, giving a set of signatures to Bob, who can then forward a subset of them, corresponding to the extracted document fragments, to Cathy and David. Some problems with this simplistic approach include both high computation (n signatures) for Alice, Cathy and David as well as high communication overhead from Alice to Bob, Bob to Cathy and Bob to David (up to n times the length of one signature). **We want to do better than this simple scheme in either:**

i communication overhead savings; **OR**

ii computation savings;

as well as:

iii giving the signer ability to specify allowed extraction of document content; and

iv achieving provable security.

We call schemes that perform better at document fragment extraction than the simple scheme above: **Content Extraction Signatures** (CES).

1.1 Contents of this Paper

Detailed definitions and some proofs are omitted in this version of the paper — they will be included in the full version of this paper, under preparation. This paper is organized as follows. In Section 2 we explain the real-life background motivating the introduction of Content Extraction Signatures, and review related work. In Section 3, we describe our desirable functional and security requirements from a CES (which differ from those of a standard digital signature — there is even a new *privacy* requirement which has no counterpart in standard digital signatures), leading to a definition of a CES as a new primitive satisfying these requirements. Section 4 presents four CES schemes meeting our definition which are provably secure under well-known cryptographic assumptions. Our CES schemes do *not* employ new cryptographic techniques. However, we believe this application for these known cryptographic techniques is novel, and in the case of the RSA schemes exploits additional properties which are useful in this application (namely to reduce communication bandwidth) but are not used to advantage in most of the original uses of these techniques. In Section 5 we close with some concluding comments.

2 Background

2.1 The Emerging Need

To motivate the rest of the paper, consider the commonplace although pertinent example illustrated in Fig. 2 below. In this example, a university (A) issues a student (B) with a formal document: an Academic Transcript (Original document). The student is required to include the formal document with a job application document sent to a prospective employer (C). Note that the Academic Transcript document is likely to include the student's personal details, in particular the student's date of birth (DOB). To avoid age-based discrimination, the student B may not wish to reveal his DOB to the employer C (indeed, in some countries it is illegal for the prospective employer to seek the applicant's DOB). The university (A) understands this and is willing to allow employers to verify academic transcripts with the DOB removed (and possibly with other fields agreed to by A removed as well, but not others which A may require to be included in any extracted document). Yet if A signs the Academic Transcript using a standard digital signature the student B must send the whole document to C to allow C to verify it.

Instead, we propose in this situation the use of a *Content Extraction Signature* (CES), as described in this paper. The university (A) uses the Sign algorithm of a CES scheme to sign the original document, divided into portions (*submessages* m_0, m_1,...) and produce a content extraction signature, given to student B along with the full document. The student then extracts a *subdocument* A' of the original document consisting of a selected subset of the document submessages (e.g. not including m_1, the DOB of B, but including all other submessages). He then runs an Extract algorithm of the CES scheme to produce an *extracted signature* by the university A for the extracted subdocument A'. Student B then forwards the subdocument A' and the extracted signature for A'. The employer uses the Verify algorithm of the CES to verify the extracted signature on A'. This process can also be repeated for other prospective employers as illustrated in Fig. 2.

2.2 Virtues of Content Extraction Signatures

If one views a document as a collection of facts or statements, the use of a CES with documents enables a user to handle and process these statements in a more selective manner rather than having to use the entire document. Apart from the efficiency of only sending the pertinent information, more importantly, it enables the user, Bob, to embrace a *minimal information model* whereby all of the extra details that are not required can be omitted from the transaction. Once the document has been sent out, there are no mechanisms for expiring or recalling it. This is vitally important as the privacy and use of the information is at significantly more risk than any time in history due to

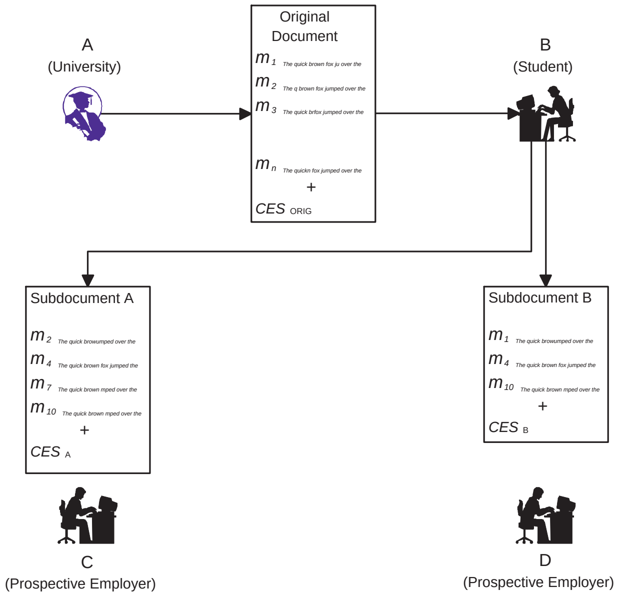
A
(University)
Original
Document
m 1 The quick brown fox ju over the
m 2 The q brown fox jumped over the
m 3 The quick brfox jumped over the
m n The quickn fox jumped over the
+
CES ORIG
B
(Student)
Subdocument A
m 2 The quick browumped over the
m 4 The quick brown fox jumped the
m 7 The quick brown mped over the
m 10 The quick brown mped over the
+
CES A
Subdocument B
m 1 The quick browumped over the
m 4 The quick brown fox jumped the
m 10 The quick brown mped over the
+
CES B
C
(Prospective Employer)
D
(Prospective Employer)

Figure 2: A real-life application for Content Extraction Signatures.

the ease with which it can be electronically relayed and stored. This is especially true with respect to the Internet.

Our approach views a document as not just a collection of facts or statements. It also recognises that the document, through its name or title, also provides a contextual framework for the collection of facts or statements. In the simplistic approach of signing each fragment so that Bob can selectively forward them to Cathy, Alice has no protection from contextual or semantic abuse due to reordering of the fragments sent to Cathy by Bob.

In order to afford the document signer some control over what sub-documents may be generated, we have included the notion of an Extraction Policy as a fundamental element of the CES. The Extraction Policy enables the signer to have some control over the use of the content even though the document has been forwarded to the user and no further contact will occur. A CES does not require any further interaction with the author (after the initial signing), meaning that the author is oblivious to all further uses of (and extraction from) the document by the user. It also permits the lifespan of the document to be independent to the lifespan of the author.

While there may be others, we currently envisage two types of situations that may create a need in practice to extract portions of a signed document before forwarding to the verifier:

i *Privacy Protection:* The full document contains some sensitive/secret portions, which the document owner may not wish to reveal to some verifiers.

ii *Bandwidth Saving:* The full document may be very long, while the verifier is only interested in a small portion of it.

2.3 Subdocument Presentation

The extraction of a subdocument along with its extracted signature (CES) raises the question: Should we enforce the requirement that positions of the blinded content be flagged with some sort of ‘blinded fragment indicator’ for the verifier? The simplistic multiple-signature approach does not enforce this requirement. In our security model for CES (see section 3), we define a ‘CES-Unforgeability’ notion which does enforce it (it preserves the total number of fragments in the original document as well as the positions of blinded/unblinded fragments). This feature might be used to simplify the signer’s Extraction Policy discussed above, by alerting the verifier to positions of deleted fragments (e.g. fragments deleted by an attacker in order to change meaning of the extracted document) without those deletions having to be explicitly ruled out by the Extraction

Policy (since explicitly ruling these deletions out can potentially make the Extraction Policy size very large in the case of large documents).

In some applications, it may be preferable to not display the positions of blinded fragments to the verifying person to achieve a better presentation, as the document is not cluttered throughout with blinded fragment indicators. This might be achieved by letting the signer add an optional statement to the Extraction Policy allowing such a display to be specified by the user who extracts a subdocument, although the blinded fragment positions are still communicated to and verified by the verification algorithm. In some other cases, the verifier, Cathy, might have some negative attitudes towards Alice if she receives a subdocument that has indicators of blinded content in it, even though it contained all of the information Cathy was seeking. So one can also require a special privacy requirement, namely that a curious Cathy cannot, by looking at the extracted subdocument and signature, tell whether she received all the submessages in the original document or not. We do not deal with this requirement in the current model.

2.4 Related Work

Bellare, Goldreich and Goldwasser[14] introduced 'incremental' signatures which allow a *signer* to efficiently update a signature after making an incremental operation on the signed document. Our 'Content Extraction Signatures' can be viewed as addressing the 'deletion' operation efficiently, but allowing this operation to be performed *without* the signer's secret key. While [14] use the standard signature model where this operation would constitute forgery, it is not necessarily a forgery in our 'Content Extraction Access Structure'-based CES-unforgeability definition (see section 3), which allows the signer to specify which deletions are to be regarded as forgeries.

The 'dual signature' defined for the Secure Electronic Transaction (SET) protocol [6], can be considered a special case of one of our schemes, although in that application it is again the signer who performs the 'extraction', and there are only two submessages (payment information and order information). It is therefore a different setting to ours.

The 'XML-Signature'[7] proposed recommendation of the WWW Consortium (W3C) defines a scheme which allows the signing of documents consisting of multiple online 'data objects', some of which may become unavailable after signing. However, the content extraction application and its requirements are not addressed in the recommendation. Indeed, the proposed recommendation cannot be used without modification to achieve our privacy requirement for content extraction signatures. Furthermore, our approach deals with self-contained documents.

Three of our CES schemes are modifications of batch signature generation and verification schemes [16] [8][11]. However, our schemes are a new application for these techniques. In particular, we use in a novel way the homomorphic property of the RSA schemes which is not used to save bandwidth in standard batch signing and verifying where the output (in the case of signing) is n signatures and the input (in the case of verifying) is n signatures. We are able to take advantage of this feature because in our setting all the n signatures are intended for the same user (from the point of view of batch signing) or are transmitted from a single user (but not the signer) from the point of view of batch verification. Fiat [11] mentions another application of the length-saving feature of his batching scheme for the purpose of saving bandwidth in distributed decryption.

3 Requirements and Definition of Content Extraction Digital Signatures

In this section we explain informally the requirements which a CES scheme should satisfy leading to an (informal) definition for a CES scheme and its desirable security notions. Refer to the full paper for more precise formulations of these definitions.

3.1 Terminology and Document Model

First we introduce some terminology. We will assume that a document is divided into a number n of smaller *submessages*, and that such documents are represented in a form which encodes an ordering of the n submessages, their number, and allows some of them to be 'blank'. We use $\mathsf{length}(\mathbf{M})$ to denote the number of submessages in $\mathbf{M}$. We use $[n]$ to denote the set of submessage indices $\{1, ..., n\}$. We use bold letters (e.g. $\mathbf{M} = (m_1, m_2, \ldots, m_n)$) to denote such a representation of documents. This representation of a document $\mathbf{M}$ allows *extraction* of a *subdocument* $\mathbf{M}'$ specified by an *extraction subset* X consisting of the indices of the submessages to be extracted and included in the subdocument. The subdocument $\mathbf{M}'$ then still has n submessages, where the submessages with indices not in X are set to 'blank', while the submessages with indices in X are equal to the corresponding submessages with the same indices in the original document $\mathbf{M}$. We denote by $\mathbf{M}[i]$ the i'th submessage in document $\mathbf{M}$. We use $\mathsf{Cl}(\mathbf{M})$ to denote the set of indices of 'clear' (i.e. non-blank) submessages in a document $\mathbf{M}$.

For example, if the original document is $\mathbf{M} = (m_1, m_2, m_3, m_4)$ and the extraction subset is $X = \{1, 3\}$, then the extracted subdocument is $\mathbf{M}' = (m_1, ?, m_3, ?)$, where ? denotes a blank submessage. Also, we have $\mathsf{Cl}(\mathbf{M}') = \{1, 3\}$. Note that by our definitions $\mathbf{M}'$ is distinct from the documents $(m_1, m_3, ?, ?)$, $(m_3, ?, m_1, ?)$, or $(m_1, ?, m_3, ?, ?)$ which are *not* subdocuments of $\mathbf{M}$.

3.2 Functional Requirements from a CES

Here we discuss the *functional* requirements from a CES, i.e. those which are concerned with *honest* users of the scheme. The basic requirement from a CES scheme can be stated as follows.

Extraction Requirement: *A Content Extraction Signature should allow anyone, given a signed document, to extract a publicly verifiable extracted signature for a specified subdocument of a signed document, without interaction with the signer of the original document.*

As explained in the introduction, the extraction requirement can be met by a simple solution of signing each submessage seperately using a standard digital signature. In order to make this problem more interesting we have asked for an efficiency requirement as follows.

Efficiency Requirement: *A Content Extraction Signature should be more efficient, either in communication overhead (length of original or extracted signatures), or in computational requirements (signing or verifying), than the simple 'multiple signature' solution.*

An additional requirement which may be useful in some applications is the following one. It allows the extraction to continue 'downstream' as the document is passed from verifier to verifier, allowing each one to continue the extraction process. It may be considered optional.

Iterative Extraction Requirement: *A Content Extraction Signature should allow the extraction of a signature for a subdocument of a source subdocument, given the source subdocument and the extracted signature for it.*

3.3 Security Requirements from a CES

Now we discuss the *security* requirements from a CES, i.e. those which prevent undesirable *dishonest* operations.

The most basic security requirement is, as for standard digital signatures, to ensure authenticity via the *unforgeability* requirement. The unforgeability requirement for a CES must differ, however, from the standard 'existential unforgeability' one, because in the latter, any proper subdocument (i.e. a subdocument which is not equal to the original document) of the signed document is considered a 'new message' and therefore the extraction (using public knowledge only) of a valid signature for it constitutes existential forgery. As we have pointed out, however, we are considering situations where it is desirable to relax this model, and allow extraction of signatures for certain subdocuments. However, it is clear that in this model the *signer must have full control to determine which subdocuments signatures can be extracted for.* The signer may want to force some portions as manadatory for inclusion in any extracted subdocument. And it may be necessary to protect against changes in meaning of extracted portions due to certain deletions.

We will address the above consideration through an Extraction Policy as an additional input to the signing algorithm of the CES scheme, called the *Content Extraction Access Structure* (CEAS for short). The signer uses this to specify which subdocuments the signer is allowing signatures to be extracted for (the CEAS is therefore an encoding of the allowed extraction subsets, using the terminology introduced above). This leads to the following unforgeability requirement for a CES, assuming the strong 'chosen message' attack, where the attacker can query a CES signing oracle on documents of the attacker's choice. Note that any document $\mathbf{D}$ queried by the attacker to the signing oracle is accompanied by a corresponding CEAS (also under the attacker's choice) specifying allowed extracted subdocuments.

CES-Unforgeability Requirement: *It is infeasible for an attacker, having access to a CES signing oracle, to produce a document/signature pair* $(\mathbf{M}, \sigma)$*, such that: (i)* σ *passes the verification test for* $\mathbf{M}$ *and (ii)* $\mathbf{M}$ *is either (A) Not a subdocument of any document queried to the CES signing oracle, or (B) Is a subdocument of a queried document* $\mathbf{D}$*, but not allowed to be extracted by the CEAS attached to the sign query* $\mathbf{D}$.

We remark that the definition of CES-unforgeability above is as strong as one may hope for in our setting, i.e. it prevents forgeries for any documents which are not subdocuments of signed documents. This implies, in particular, security against 'submessage reordering attacks' (where the submessages in the forged document are the same as those in subdocument of a signed document but in different positions or order) as well as 'mixed subdocument attacks' (where the forged document contains submessages which have all appeared in signed documents but have never appeared *together* in a signed document). Note that the simple multiple signature scheme is *not* secure against both of the latter attacks and hence does not even have the CES-unforgeability property without modification.

Unlike standard signatures, where unforgeability is the only security requirement, many applications of content extraction signatures by nature may also have a *privacy* security requirement. Indeed, as mentioned in the introduction, the reason for the user to delete some submessages in the signed document prior to handing it over to the verifier may be in order to hide the sensitive content of these submessages from the verifier. But none of the above requirements on a CES exclude the possibility that the extracted signature may in general leak some information on the content of the unextracted (deleted) submessages. Indeed, this consideration has important implications on the design and efficiency of our first two proposed CES schemes (see section 4). Hence, the privacy requirement for a CES must be defined in order to allow an assessment of whether a scheme satisfies the requirement or not. We give this definition here. We note that a similar requirement has been defined in the context of 'incremental cryptography' [15].

CES-Privacy Requirement: *It is infeasible for an attacker to distinguish between two extracted signatures* σ_0 *and* σ_1*, where* σ_0 *is extracted for a subdocument* $\mathbf{D_0}$ *of the document* $(m_0, ..., m_{i-1}, M_0, m_{i+1}, ..., m_n)$ *specified by extraction subset* X *such that* $i \notin X$ *(i.e.* $\mathbf{D_0}$ *has a blank submessage in position* i*), and* σ_1 *is extracted for subdocument* $\mathbf{D_1}$ *of the document* $(m_0, ..., m_{i-1}, M_1, m_{i+1}, ..., m_n)$ *specified by the same extraction subset* X *(i.e.* $\mathbf{D_1} = \mathbf{D_0}$*). This holds even if the attacker can choose* (M_0, M_1)*,* X*,* i*, and the remaining submessages* m_i*.*

The above definition is similar to the 'indistinguishability' based definition of semantic security for encryption schemes. It ensures the attacker cannot obtain information about submessages which have been blanked (i.e. those not extracted into the subdocument). Its precise formulation (included in the full paper) follows the standard two-stage 'find/guess' attack experiment [17, 1]. In this formulation the attacker's find algorithm chooses (M_0, M_1), X, i, and the m_i's. A uniformly random bit b is chosen and the attacker's guess algorithm is given σ_b and tries to guess b. The requirement is that it is infeasible for guess to guess b correctly with non-negligible advantage over random guessing.

3.4 Definition of a Content Extraction Signature

Consistent with the requirements discussed above, a *Content Extraction Digital Signature* (CES) scheme $\mathcal{D}$ can be defined to consist of 4 algorithms:

1 KeyGen — Takes a security parameter k and generates a secret/public key pair (SK, PK).

2 Sign — Takes a secret key SK, a document $\mathbf{M} = (m_1, m_2, ..., m_n)$, and a content extraction access structure $CEAS$, and outputs a content extraction signature σ_{Full}.

3 $\mathsf{Extract}$ — Takes a document $\mathbf{M}$, a content extraction signature σ_{Full}, an extraction subset $X \subseteq [n]$ and a public key PK, and outputs an extracted signature σ_{Ext}.

4 Verify — Takes an extracted subdocument $\mathbf{M}'$, an extracted signature σ_{Ext} and a public key PK, and outputs a verification decision $d \in \{Accept, Reject\}$.

In the above definition, the main differences from a standard digital signature are the following.

The $\mathsf{Extract}$ algorithm allows the user to extract (from a 'full' content extraction signature σ_{Full}) a signature for the subdocument consisting of the submessages whose indexes are specified by the extraction subset X. The extracted signature σ_{Ext} can then be forwarded to the verifier along with the extracted subdocument $\mathbf{M}'$.

The 'Content Extraction Access Structure' (CEAS) is an encoding of the subsets of submessage indexes in the original document which the signer can use to specify which extracted subdocuments the user is "allowed" to extract valid signatures for. Therefore the $CEAS$ is an encoding of a collections of subsets of $[n]$, where $n = \mathsf{length}(\mathbf{M})$ and $\mathbf{M}$ is the signed document. We assume these subsets are encoded as bit strings in $\{0, 1\}^n$ so that if $\mathsf{Cl}(\mathbf{M}') \in CEAS$ for some document $\mathbf{M}'$ then $\mathsf{length}(\mathbf{M}') = n = \mathsf{length}(\mathbf{M})$.

4 Proposed Content Extraction Signature Schemes

4.1 Schemes based on General Cryptographic Primitives

4.1.1 Scheme CommitVector.

We build this CES scheme out of two standard cryptographic primitives:

(i) A standard digital signature scheme $\mathcal{S}$ with signature and verification algorithms (S, V) and key generation algorithm K. We require that $\mathcal{S}$ satisfies the standard unforgeability notion, i.e. it is existentially unforgeable under adaptive chosen message attacks [19].

(ii) A message commitment scheme $\mathcal{C}$ with committing algorithm $\mathsf{Com}(.,.)$ and common parameters generation algorithm GenPar. Given a message m, $\mathsf{Com}(m, r; cp)$ denotes the commitment to message m under random value r and common parameters cp. We require that $\mathcal{C}$ satisfy the following standard properties —

1 Hiding: Given $\mathsf{Com}(m, r; cp)$, the message m is semantically secure [17]

2 Binding: It is hard to find any two distinct messages $m \neq m'$ and corresponding randomness values $r \in \mathsf{Rand}(m; cp)$ and $r' \in \mathsf{Rand}(m'; cp)$ such that $\mathsf{Com}(m, r; cp) = \mathsf{Com}(m', r'; cp)$

3 Efficient Verification: Given m and r it is easy to compute $c = \mathsf{Com}(m, r; cp)$, or output 'Reject' if $r \notin \mathsf{Rand}(m; cp)$.

In (2) and (3) above, we denote by $\mathsf{Rand}(m; cp)$ the set of valid randomness values for a committed message m.

The scheme is simple. To sign a document, one commits to the submessages and signs the concatenation of the commitments. The CEAS is appended to the signature and is also appended to each submessage before committing. The randomness values used in the commitments are also appended. Extraction of a subset of submessages requires one to recompute the commitments to the removed (unextracted) submessages and append them. The randomness values for the removed submessages are removed. The result is the extracted signature. To verify the extracted signature on an extracted subdocument, one recomputes the commitments for the extracted submessages and verifies the signature on all of the commitments.

Computation. The main advantage of this scheme over the multiple signature one is that n signatures are replaced by 1 signature (on a message of length proportional to n), but at a cost of n commitments. This can give a saving in computation if the commitments can be computed faster than signatures. In most practical cases the signature would probably be number-theory based and signature computation would involve a modular exponentiation. This rules out the use of number-theory based commitments, which would result in little or no savings. Fortunately, an efficient provably secure statistically hiding commitment scheme can be constructed from any collision-resistant hash function [20], and there exist fast candidates for such functions, such as the well known Secure Hash Algorithm (SHA) [4]. Using this commitment scheme with a number-theory based signature, one can achieve a computational saving by a factor close to n over the multiple signature scheme, for sufficiently short submessages.

Signature Length. The signature length saving of this scheme over the simple one is small or non-existent. This is because one must append the commitment randomness values for extracted submessages and the commitments for the unextracted (removed) submessages.

Security. The scheme meets our security requirements for CES (as defined in Section 3), and we can state the following results.

Theorem 1 *(1) If the standard signature scheme $\mathcal{S}$ is existentially unforgeable under chosen message attack and the commitment scheme $\mathcal{C}$ is binding, then scheme* CommitVector *has the CES-Unforgeability property. (2) If the commitment scheme $\mathcal{C}$ is hiding, then scheme* CommitVector *has the CES-Privacy property.*

Proof. *(Sketch)* (1) We use an efficient CES-Unforgeability attacker $\mathsf{A}_{\mathsf{CES}}$ for scheme CommitVector to construct two efficient attackers: $\mathsf{A}_{\mathsf{sig}}$ to break the unforgeability of scheme $\mathcal{S}$ and $\mathsf{A}_{\mathsf{com}}$ to break the binding of scheme $\mathcal{C}$. We will prove our claim by showing that one of $\mathsf{A}_{\mathsf{sig}}$ or $\mathsf{A}_{\mathsf{com}}$ succeeds in its attack with non-negligible probability. Attacker $\mathsf{A}_{\mathsf{sig}}$ works as follows. It runs $\mathsf{A}_{\mathsf{CES}}$, and answers the latter's sign queries using the signing oracle $\mathsf{S}(.)$ for scheme $\mathcal{S}$. While doing so, $\mathsf{A}_{\mathsf{sig}}$ stores in a table both $\mathbf{D}_j$, the jth document queried by $\mathsf{A}_{\mathsf{CES}}$ to Sign (for $j \in \{1, \ldots, q_S\}$, where q_S is the number of sign queries made by $\mathsf{A}_{\mathsf{CES}}$), and also $(\mathbf{c}_j[i], \mathbf{r}_j[i])$, where $\mathbf{c}_j[i]$ denotes the commitment to the ith submessage of $\mathbf{D}_j$ under randomness $\mathbf{r}_j[i]$, used by $\mathsf{A}_{\mathsf{sig}}$ to answer the jth query to Sign. We let $CEAS_j$ denote the CEAS attached to the jth Sign query of $\mathsf{A}_{\mathsf{CES}}$. Since $\mathsf{A}_{\mathsf{CES}}$ sees a perfect simulation of the real attack, it by assumption succeeds to break CES-unforgeability of CommitVector with non-negligible probability ϵ by outputting a CES forgery pair $(\mathbf{D}^*, \sigma^*_{Ext})$, where $\sigma^*_{Ext} = CEAS^* \| \sigma^* \| Conc_{i \in Cl(\mathbf{M}^*)} \mathbf{r}^*[i] \| Conc_{i \in [n^*]-Cl(\mathbf{M}^*)} \mathbf{c}^*[i]$, $n^* \stackrel{\text{def}}{=} \mathsf{length}(\mathbf{M}^*)$. For each $i \in \mathsf{Cl}(\mathbf{D}^*)$, denote by $\mathbf{c}^*[i]$ the commitment to the i'th submessage of $\mathbf{D}^*$ reconstructed from randomness value $\mathbf{r}^*[i]$ in σ^*_{Ext}. By definition of CES-unforgeability, we have that for each $j \in \{1, \ldots, q_S\}$, at least one of the following three events occurs:

(i) There exists $i \in \mathsf{Cl}(\mathbf{M}^*)$ such that $\mathbf{D}^*[i] \neq \mathbf{D}_j[i]$ and $\mathbf{c}^*[i] = \mathbf{c}_j[i]$.

(ii) There exists $i \in \mathsf{Cl}(\mathbf{M}^*)$ such that $\mathbf{D}^*[i] \neq \mathbf{D}_j[i]$ and $\mathbf{c}^*[i] \neq \mathbf{c}_j[i]$.

Scheme $\mathsf{CommitVector}$

(1) $\mathsf{KeyGen}(k)$: (1.1) Compute $(SK_S, PK_S) \leftarrow \mathsf{K}(k)$ (1.2) Compute $cp \leftarrow \mathsf{GenPar}(k)$ (1.3) Output $PK = PK_S \| cp$ and $SK = SK_S$.

(2) $\mathsf{Sign}(\mathbf{M}, CEAS, SK)$: (2.1) Parse $PK = PK_S \| cp$ (2.2) Compute commitments $\mathbf{c}[i] \leftarrow \mathsf{Com}(\mathbf{M}[i], \mathbf{r}[i]; cp)$ for $i \in [n]$. (2.3) Define $c \leftarrow Conc_{i=1}^{n} \mathbf{c}[i]$ (2.4) Compute $\sigma_c \leftarrow \mathsf{S}(CEAS \| c, SK)$. (2.5) Output $\sigma_{Full} \leftarrow CEAS \| \sigma_c \| Conc_{i=1}^{n} \mathbf{r}[i]$.

(3) $\mathsf{Extract}(\mathbf{M}, \sigma_{Full}, X, PK)$: (3.1) Parse $PK = PK_S \| cp$ (3.2) Parse $\sigma_{Full} \leftarrow CEAS \| \sigma_c \| Conc_{i=1}^{n} \mathbf{r}[i]$ (3.3) Compute missing commitments $\mathbf{c}[i] \leftarrow \mathsf{Com}(\mathbf{M}[i], \mathbf{r}[i]; cp)$ for $i \in [n] - X$ (3.4) Output $\sigma_{Ext} \leftarrow CEAS \| \sigma_c \| Conc_{i \in X} \mathbf{r}[i] \| Conc_{i \in [n]-X} \mathbf{c}[i]$.

(4) $\mathsf{Verify}(\mathbf{M}', \sigma_{Ext}, PK)$: (4.1)Parse $PK = PK_S \| cp$ (4.2) Parse $\sigma_{Ext} = CEAS \| \sigma_c \| Conc_{i \in Cl(\mathbf{M}')} \mathbf{r}[i] \| Conc_{i \in [n]-Cl(\mathbf{M}')} \mathbf{c}[i]$ (4.3) Recompute commitments $\mathbf{c}[i] \leftarrow \mathsf{Com}(\mathbf{M}'[i], \mathbf{r}[i]; cp)$ for $i \in Cl(\mathbf{M}')$ or output "Reject" if $\mathbf{r}[i] \notin \mathsf{Rand}(\mathbf{M}'[i])$ for some i. (4.4) Compute $c \leftarrow Conc_{i=1}^{n} \mathbf{c}[i]$ (4.5) Verify signature $d \leftarrow \mathsf{V}(CEAS \| c, \sigma_c, PK_S)$. (4.6) Output "Accept" iff $d =$ "*Accept*" and $\mathsf{Cl}(\mathbf{M}') \in CEAS$.

(iii) $\mathsf{Cl}(\mathbf{M}^*) \notin CEAS_j$.

Attacker $\mathsf{A_{sig}}$ outputs the forgery pair (m^*, σ^*), where $m^* \stackrel{\text{def}}{=} CEAS^* \| \mathbf{c}^*[1] \| \ldots \| \mathbf{c}^*[n^*]$. The attacker $\mathsf{A_{com}}$ works as $\mathsf{A_{sig}}$, except that it uses a self-generated signature secret key to answer Sign queries. It then searches for some $j \in \{1, \ldots, q_S\}$ and $i \in Cl(\mathbf{M}^*)$ such that event (i) holds, and if found outputs a binding collision $[(\mathbf{D}^*[i], \mathbf{r}^*[i]), (\mathbf{D}_j[i], \mathbf{r}_j[i])]$.

Now note that if event (i) occurs for some $j \in \{1, \ldots, q_S\}$ then attacker $\mathsf{A_{com}}$ succeeds in breaking the binding of commitment scheme $\mathcal{C}$. If event (i) does not occur for all $j \in \{1, \ldots, q_S\}$, then $\mathsf{A_{sig}}$ succeeds in breaking the unforgeability of signature scheme $\mathcal{S}$ because the message m^* was never queried to $\mathsf{S}(.)$ by $\mathsf{A_{sig}}$ and σ^* is a valid signature for m^*. This is because for each j, if event (ii) occurs for some i then m^* differs from the jth queried message $m_j \stackrel{\text{def}}{=} CEAS_j \| \mathbf{c}_j[1] \ldots \mathbf{c}_j[n_j]$ to $\mathsf{S}(.)$ in the ith commitment, that is $\mathbf{c}^*[i] \neq \mathbf{c}_j^*[i]$. And for each j for which event (iii) occurs, we again have $m^* \neq m_j$ because $\mathsf{Cl}(\mathbf{M}^*) \in CEAS^*$ and consequently $CEAS_j \neq CEAS^*$. So in any outcome where $\mathsf{A_{CES}}$ succeeds, one of $\mathsf{A_{com}}$ or $\mathsf{A_{sig}}$ also succeed. This proves that one of the attackers $\mathsf{A_{sig}}$ and $\mathsf{A_{com}}$ succeeds with probability at least $\epsilon/2$.

(2) We use a CES-Privacy attacker $\mathsf{A_{CES}}$ to construct an attacker $\mathsf{A_c}$ to break the hiding property of commitment scheme $\mathcal{C}$. Attacker $\mathsf{A_c}$ simply runs the find stage of $\mathsf{A_{CES}}$ to get a pair of submessages (M_0, M_1), and then completes the commitment to M_b given to it (for a random bit b unknown to $\mathsf{A_c}$) to form a CES signature with M_b unextracted. It then gives the CES signature to the guess stage of $\mathsf{A_{CES}}$ and uses its output to predict b. Since $\mathsf{A_{CES}}$ sees a perfect simulation, it follows that $\mathsf{A_c}$ succeeds in predicting b with $\mathsf{A_{CES}}$'s success probability, hence breaking the hiding property of commitment scheme $\mathcal{C}$. □

4.1.2 A Variant: Scheme HashTree.

This scheme is a modification of the previous scheme in order to improve its main shortcoming, namely its large signature length. This length consists of two components (besides the single $\mathcal{S}$ signature): the first is due to the randomness values for extracted submessages (this one is proportional to m) and the second is due to the commitments for the unextracted submessages (this one is proportional to $n - m$). The scheme $\mathsf{HashTree}$ significantly reduces the length of the second component when m is much smaller than n. It does so in a simple way: We use a collision-resistant hash function $H(.)$ to construct a hash tree by modifying the signing algorithm to regard the set of commitments as the leaves of a binary "hash tree". The hash tree is a binary tree having $n = 2^k$ leaf nodes, which are associated with the commitments $(h_1, ..., h_n)$ of the n submessages. Then, starting at the highest level of the tree below the leaves, we associate with each internal node of the tree (down to the root node) the hash value of the concatenation of its two children nodes above it. The tree has $k+1$ levels, the lowest one consisting of a single root node, with its associated hash value h_{root}. The signer signs the root hash value h_{root} with appended CEAS encoding, as in scheme CommitVector. Extraction consists of appending to the signature the hash values associated with intermediate tree nodes which are required in order to compute the root hash value from the commitments of the extracted submessages available in the subdocument. The randomness values for the extracted submessage commitments are also appended as before. Verification recomputes the root hash and verifies the signature on it.

Hash trees have been widely discussed in the literature for improving the efficiency of authentication protocols (see [18]). This scheme can be regarded a modification of the batch signature of Pavlovski and Boyd [16] except that in our application the verifier recomputes the root hash value from any number m of leaves (extracted submessage commitments) while in their application it was recomputed from only one leaf. For simplicity we have omitted some details in the above description. We refer the reader to [16] for a discussion of how to handle any number of leaves (submessages). We also note also that the privacy afforded by the use of commitments in our scheme may also be useful in the batch signatures of [16], where the signer effectively performs the extraction into subdocuments of one submessage each, and may not wish one verifier from learning anything about the content of the other submessages signed in the batch.

Computation. This scheme maintains the computational efficiency of the previous scheme, apart from some additional hashing (of total length proportional to n) to build the hash tree. As mentioned before, since fast collision-resistant hash functions are available this does not reduce computational efficiency significantly except for very large n.

Signature Length. As for the previous scheme the extracted signature still contains the m randomness value for the extracted m submessages, so this component of signature length is still proportional to m. But in this variant, the component of signature length due to unextracted submessages (which in previous scheme was proportional to the number $n - m$ of unextracted submessages) is significantly reduced for small m. For example, in the extreme case when we extract a single submessage ($m = 1$), the previous scheme required the user to send $n - 1$ commitments for the removed submessages, while in this scheme the user only needs to send the $\log_2(n)$ hash values associated with the 'sibling' nodes of the nodes on the path from the clear submessage commitment leaf node to the tree root. More generally, one can show that the worst-case overhead for a given m is approximately $m \log_2(n/m)$ sibling hash values for $m < n/2$ and $n - m$ for $m > n/2$.

Security. The security properties of the scheme can be proved as for the previous scheme. The main difference is that for this scheme, the CES-unforgeability property also relies on the collision-resistance of the hash function $H(.)$ used to build the hash tree. We have the following results, whose proof is similar to that of Theorem 1, together with the well known collision-resistance property of the Hash tree.

Theorem 2 *(1) If the standard signature scheme $\mathcal{S}$ is existentially unforgeable under chosen message attack, the commitment scheme $\mathcal{C}$ is binding, and the hash function $H(.)$ is collision-resistant then scheme* HashTree *has the CES-Unforgeability property. (2) If the commitment scheme $\mathcal{C}$ is hiding, then scheme* HashTree *has the CES-Privacy property.*

Remark: In both HashTree and CommitVector schemes, the signer sends the user all n commitment 'randomness values' r_i and user sends the verifier a subset of m of these. A well-known trick to reduce the signer-to-user communication is for the signer to generate the r_i's using a Pseudo-Random Number Generator (PRNG) from a single short seed (say r) and send just r to the user. The CES-Privacy property we defined can still be proven in this case if the PRNG is secure in the standard 'indistinguishability from random' sense. This trick does not, however, reduce the user-to-verifier communication. To also reduce the latter, one may ask if it is possible to design a special PRNG with a 'seed extraction' property, which can be informally stated as follows: Given a (short) seed r (generating Pseudorandom numbers $r_1, \ldots, r_n$), and any subset $X \subseteq [n]$, it is easy to compute a short 'extracted seed' r_X which allows one to generate the r_i for $i \in X$, but none of the other r_i's for $i \notin X$ (moreover the r_i for $i \notin X$ should be indistinguishable from random even knowing r_X). An RSA-based candidate for this type of PRNG, having constant (independent of $|X|$) length extracted seeds, is suggested in [3]. Briefly, in this PRNG, the original seed consists of the prime factors of an RSA modulus $N = pq$ and a random element $r \in \mathbb{Z}_N^*$. There are also n fixed relatively prime public exponents $e_1, \ldots, e_n$. From the seed we generate a sequence of n pseudorandom numbers $r_i = H(r^{\prod_{k \in [n]-i} e_k} \bmod N)$ for some one-way hash function $H(.)$. The extracted seed $r_X = r^{\prod_{k \in [n]-X} e_k} \bmod N$ can easily be used to generate r_i for $i \in X$. Unfortunately we cannot prove the pseudorandomness of this PRNG with respect to the RSA one-wayness assumption without additional 'non-standard' assumptions (e.g. assuming $H(.)$ to be a random oracle). This PRNG would also increase the computational requirements of the scheme. Finally, note that essentially the same construction was previously proposed [2] in the context of 'hierarchical access control'.

4.2 Schemes Based on RSA

The previous two CES schemes improve upon the computation of the simple multiple signature solution. But their signature length is linear in the number of submessages. In this section we show how to achieve the reverse tradeoff. We propose RSA-based CES schemes whose signature length is significantly shorter than that of the simple multiple signature solution. The CES signature length for these schemes is approximately equal to that of a *single* standard RSA signature, independent of n. However, in computation these schemes do not save over that required for the (batch version) of the simple multiple-signature solution.

4.2.1 Scheme RSAProd

This scheme does not reduce the length of signatures communicated between the *signer* and user. However it reduces the length of *extracted* signatures communicated to the verifier.

The scheme is a modification of an RSA batch "screening" verifier, proposed by Bellare et al. [8]. This verifier is based on the homomorphic property of RSA, i.e. $h_1^d \cdot h_2^d \bmod N = (h_1 \cdot h_2)^d \bmod N$. The verifier multiplies the RSA signatures in a batch and checks whether the result is the signature of the product of the hash values. This allows "screening" of m signatures at the cost of only one RSA verification plus $2 \cdot (m-1)$ multiplications modulo N.

The crucial observation which forms the basis of this scheme is that in the content extraction signature setting, all the signatures in the 'batch' (where a batch corresponds here to the submessages contained in the extracted subdocument and their signatures) are available to the user, who can perform the signature multiplication step above prior to forwarding to the verifier. Now all the user has to send is a single product of signatures modulo N, which is the same length as a single RSA signature, independent of the number of submessages m. The verifier then only needs to check that he received a signature for the product of the submessage hash values.

Scheme RSAProd

(1) KeyGen(k): Pick 2 random $k/2$ bit primes p and q. Compute $N = pq$. Choose a public exponent e relatively prime to $(p-1)(q-1)$. Compute $d = e^{-1} \bmod (p-1)(q-1)$. Set $PK = (N, e)$ and $SK = (N, d)$. Output (SK, PK).

(2) Sign$(\mathbf{M}, CEAS, SK)$: Compute T: In version 1 (Non-Stateful), choose T uniformly random in $\{0,1\}^{l_{tag}}$. In version 2 (Stateful), $T \leftarrow \mathit{counter}$ and $\mathit{counter} \leftarrow \mathit{counter} + 1$. For each $i \in [n]$, define $h_i \overset{\text{def}}{=} H(CEAS \| T \| i \| \mathbf{M}[i])$. Compute $\sigma_i \overset{\text{def}}{=} h_i^d \bmod N$. Output $\sigma_{Full} \overset{\text{def}}{=} (CEAS \| T \| \sigma_1 \| \ldots \| \sigma_n)$.

(3) Extract$(\mathbf{M}, \sigma_{Full}, X, PK)$: Parse $\sigma_{Full} = CEAS \| T \| \sigma_1 \| \ldots \| \sigma_n$. Compute $\sigma_f \overset{\text{def}}{=} \prod_{i \in X} \sigma_i \bmod N$. Set $\sigma_{Ext} \overset{\text{def}}{=} T \| CEAS \| \sigma_f$. Output σ_{Ext}.

(4) Verify$(\mathbf{M}', \sigma_{Ext}, PK)$: Set $X = \mathsf{CI}(\mathbf{M}')$. Parse $\sigma_{Ext} = T \| CEAS \| \sigma_f$. For each $i \in X$ compute $h_i \overset{\text{def}}{=} H(CEAS \| T \| i \| \mathbf{M}'[i])$. Test whether $\sigma_f^e = \prod_{i \in X} h_i \bmod N$. Output "Accept" iff the test is passed, **and** $X \in CEAS$.

Computation. The scheme needs the signer to produce n standard RSA signatures, so signer computation is identical to that of the trivial solution. The verifier's work is one signature verification plus $(m-1)$ multiplications modulo N. Our Verify algorithm is faster than the screener of [8], due to two reasons. First, we offload the signature multiplication step to the Extract algorithm. Second, we do not need to perform the 'pruning step' of eliminating duplicate messages, which is necessary in [8] in order to avoid the attack described in [9]. In our setting the sequence numbers appended to the submessages by the verifier guarantee distinctness and allow us to prove CES-unforgeability without pruning. This is an additional use for the sequence numbers besides the need to avoid submessage reordering attacks.

Signature Length. Although the full signature length from signer to user is still n times the length of a single RSA signature, once the user extracts a subdocument the resulting extracted signature has only the length of one RSA modulus (plus a small overhead for the CEAS encoding and CES tag). It is therefore much shorter than both the simple multiple signature solution and the previous two commitment-based schemes.

Security. We have the following security result for the scheme. This result holds in the *standard* model, i.e. without any random oracle assumption on the hash function $H(.)$. In this model the hash function $H(.)$ is chosen by the signer at key generation time and the algorithm for $H(.)$ is published with the signer's public key.

Theorem 3 *(1) If the Full-Domain-Hash RSA signature* $m \rightarrow H(m)^d \bmod N$ *(*$\mathsf{FDH} - \mathsf{RSA}$*) is existentially unforgeable under chosen message attack and CES-Tags are never reused by the CES* Sign *algorithm then scheme* RSAProd *has the CES-Unforgeability property (2) Scheme* RSAProd *has the CES-Privacy property.*

Proof. (Sketch) (1) By Lemma 1 in appendix A it suffices to prove the *'weak ordered' (WO)* CES-Unforgeability property for the simplified version of scheme RSAProd which does not append the

CES tag and CEAS to submessages before hashing (i.e. only the submessage sequence number is appended). We use a 'weak ordered' CES-Unforgeability attacker $\mathsf{A}_{\mathsf{CES}}$ for scheme $\mathsf{RSAProd}$ to construct a chosen message attacker $\mathsf{A}_{\mathsf{FDH}}$ to break the existential unforgeability of the signature scheme $\mathsf{FDH}-\mathsf{RSA}$. Attacker $\mathsf{A}_{\mathsf{FDH}}$ works as follows. It runs $\mathsf{A}_{\mathsf{CES}}$, and answers the latter's sign queries by simulating the Sign algorithm using the signing oracle $\mathsf{S}_{\mathsf{FDH}}. = \mathsf{H}(.)^d \bmod N$ for scheme $\mathsf{FDH}-\mathsf{RSA}$. While doing so, $\mathsf{A}_{\mathsf{FDH}}$ stores in a table the documents queried by $\mathsf{A}_{\mathsf{CES}}$ to Sign (we denote by $\mathbf{D}_j$ the jth queried document). Since $\mathsf{A}_{\mathsf{CES}}$ sees a perfect simulation of the real attack, it by assumption succeeds to break WO-CES-unforgeability of $\mathsf{RSAProd}$ with non-negligible probability ϵ by outputting a forgery pair $(\mathbf{D}^*, \sigma^*_{Ext})$, where $\sigma^*_{Ext} = \sigma^*$. By definition of WO-CES-unforgeability, there is a submessage index i such that $\mathbf{D}^*[i] \neq \mathbf{D}_j[i]$ for all j. Hence the message $m^* \overset{\text{def}}{=} i \| \mathbf{D}^*[i]$ has never been queried by $\mathsf{A}_{\mathsf{FDH}}$ to its signing oracle $\mathsf{S}_{\mathsf{FDH}}$. So $\mathsf{A}_{\mathsf{FDH}}$ breaks the existential unforgeability of $\mathsf{FDH}-\mathsf{RSA}$ by outputing the forgery (m^*, σ^*_{FDH}) if it can compute $\sigma^*_{FDH} \overset{\text{def}}{=} \mathsf{S}_{\mathsf{FDH}}(m^*) = H(m^*)^d \bmod N$ without querying m^* to $\mathsf{S}_{\mathsf{FDH}}$. $\mathsf{A}_{\mathsf{FDH}}$ does so as follows. Since σ^* is by assumption a valid CES signature for document $\mathbf{D}^*$, we have $\sigma^* = \prod_{k \in X} H(k \| \mathbf{D}^*[k])^d \bmod N$, that is $\sigma^* = \sigma^*_{FDH} \cdot R \bmod N$, where $R \overset{\text{def}}{=} \prod_{k \in X - \{i\}} H(k \| \mathbf{D}^*[k])^d \bmod N$ and $X \overset{\text{def}}{=} \mathsf{Cl}(\mathbf{M}^*)$. So $\mathsf{A}_{\mathsf{FDH}}$ queries the messages $k \| \mathbf{D}^*[k]$ for all $k \in X - \{i\}$ to $\mathsf{S}_{\mathsf{FDH}}$ and multiplies the answered signatures modulo N to compute R, which it then uses to compute the desired forged signature $\sigma^*_{FDH} = \sigma^*/R \bmod N$. The proof is completed by observing that the queried messages $k \| \mathbf{D}^*[k]$ for $k \neq i$ are all distinct in their sequence prefix from the forgery message m^*.

(2) The CES-Privacy is trivial since the extracted signature is independent of the unextracted submessages. □

If we use the random oracle model [13] for the hash function $H(.)$, then Theorem 3 and Lemma 2 in appendix give the following result.

Corollary 1 *If the RSA function is one-way and CES-Tags are never re-used by* Sign*, then then* $\mathsf{RSAProd}$ *has the CES-unforgeability property in the random oracle model.*

Remark 1: To achieve CES-unforgeability it is essential that CES tags are never re-used by the Sign algorithm. In the stateful version of Sign a tag length of $l_T = 80$ bit suffices for up to 2^{80} CES signatures, but for the unstateful version, one needs $l_T = 160$ bit for approx. the number of signatures to avoid birthday collisions.

Remark 2: This scheme does *not* have the optional 'Iterative Extraction' property described in section 3. Indeed, it is not hard to show that iterative extraction is as hard as inverting the RSA one-way function. However, iterative extraction would not be required in many applications. The following variant *does* allow iterative extraction.

Remark 3: It is clear from the proof of Theorem 3 that the scheme can be generalized to use any one-to-one trapdoor one-way group homomorphism in place of RSA, where the group operation and inversion are efficient.

4.2.2 A Variant: Scheme MERSAProd

The previous scheme achieves very low extracted signature length for the user to verifier communication. However the initial signature produced by the signer and given to the user is still n RSA signatures long. The following variant scheme $\mathsf{MERSAProd}$ also reduces this 'initial' signature length down to almost the length of a single RSA signature. It is a modification of a batch signature generation algorithm due to Fiat [11] and as such also gives savings in signer computation over the previous scheme. However, the scheme loses the fast verification property of the previous scheme.

Fiat [11] discovered that while standard RSA is hard to batch (see [12]), 'Multi-Exponent RSA' (MERSA) can be batched. In MERSA, the signer's public key consists of a unique RSA modulus $N = pq$ but many public exponents e_i, pairwise relatively prime and prime to $(p-1)(q-1)$ (for efficiency these n public exponents are normally chosen as the first n odd prime numbers). To each public exponent e_i there exists a corresponding secret exponent $d_i = e_i^{-1} \bmod (p-1)(q-1)$. Given a sequence $h_1, ..., h_n$ of n message hash values to be signed, Fiat's batch signing algorithm computes (more efficiently than performing n exponentiations) the n 'distinct-exponent' signatures $h_i^{d_i} \bmod N$. Fiat's algorithm is divided into 3 stages. Stage 1 outputs the product $r \overset{\text{def}}{=} \Pi_{i=1}^{n} h_i^{e/e_i} \bmod N$, where $e \overset{\text{def}}{=} \prod_{i=1}^{n} e_i$. Stage 2 signs the product and outputs $r^{1/e} = \prod_{i=1}^{n} h_i^{1/e_i} \bmod N$. Stage 3 'separates' the product into the n indvidual multi-exponent signatures $h_i^{1/e_i} \bmod N$. A crucial property of Fiat's algorithm for application in our scheme MERSAProd is the following one: *the separation algorithm (Stage 3) does not require the signer's secret key.*

Our scheme MERSAProd is a natural extension of scheme RSAProd to the MERSA case, modified to take advantage of the above useful property of Fiat's algorithm to save on signature length. It is described as follows. The key generation algorithm publishes in the public key an RSA modulus N and the first n_B primes e_i ($i = 1, ..., n$) as public exponents, keeping the corresponding d_i as secret exponents. The Sign algorithm runs only stages 1 and 2 of Fiat's algorithm to compute the product $r = \prod_{i \in [n]} h_i^{d_i} \bmod N$, where h_i denotes the hash h_i of the ith submessage (with appended CEAS, CES Tag and submessage sequence number i, as in scheme RSAProd). Only the product r (single RSA signature length) is sent to the user (together with CEAS and CES Tag). The Extract algorithm runs stage 3 of Fiat's algorithm to break up the product r into the individual signatures $h_i^{d_i}$ and then multiplies the ones corresponding to the extracted submessages in the extraction subset X as in scheme RSAProd, to compute $\sigma_f = \prod_{i \in X} h_i^{d_i} \bmod N$, to achieve the same single RSA signature length of extracted signatures. The Verify algorithm verifies σ_f on document $\mathbf{M}'$ by checking if $\sigma_f^e = \prod_{i \in X} h_i^{e/e_i} \bmod N$, where $e \overset{\text{def}}{=} \prod_{i \in X} e_i$, and $X = \mathsf{Cl}(\mathbf{M}')$. Note that this test is equivalent to testing if $\prod_{i \in X} \bar{h}_i^{e/e_i} = 1 \bmod N$ where $\bar{h}_i = h_i$ for all i except $i = L \overset{\text{def}}{=} \min_{i \in X} e_i$, for which we define $\bar{h}_L = h_L / \sigma_f^{e_L} \bmod N$. Fiat's stage 1 algorithm is again used to compute the product $\prod_{i \in X} \bar{h}_i^{e/e_i} \bmod N$ efficiently.

Computation. One can show that in using Fiat's algorithm in the above scheme, the computation involved (ignoring one-off computations which are dependant only on the public exponents) for the signer (Stages 1 and 2) is bounded in the order of $n \log_2^2(n) + \log_2(N)$ multiplications mod N. The extraction computation is in the same order, although we note that the user need only run Fiat's Stage 3 algorithm to extract and store the n signatures once, and then is able to combine any subset of m of them using only $m - 1$ modular multiplications. This might be useful if the user is supplying several distinct extracted subdocuments to multiple verifiers. Finally, the verifier performs the Stage 1 of Fiat's algorithm but using only m messages, giving a computational load in the order of $m \log_2(n) \log_2(m)$ multiplications modulo N.

Signature Length. Both initial signature length (as output by signer) and extracted signature length are only a little longer than than a single RSA signature length (due to appended CES tag and CEAS encoding).

Security. We can state the following. The proof is analogous to the proof of Theorem 3.

Theorem 4 *(1) If each of the n distinct-exponent FDH signature schemes $\mathsf{S_i}(m) \overset{\text{def}}{=} H(i \| m)^{d_i} \bmod N$ (for $i = 1, ..., n$) is existentially unforgeable even under a chosen message attack giving acccess simultaneously to all the n oracles $\mathsf{S_i}(.)$, and if CES-Tags are never reused by the CES* Sign *algorithm, then scheme* MERSAProd *has the CES-Unforgeability property (2) Scheme* MERSAProd *has the CES-Privacy property.*

Using the random oracle model for $H(.)$ we again obtain a stronger result using Lemma 3.

Corollary 2 *If the RSA function is one-way for each of the public exponents e_i (for $i \in [n]$) and CES-Tags are never reused by* Sign*, then* MERSAProd *has the CES-unforgeability property in the random oracle model.*

4.3 Performance Summary

A summary of the performance parameters of the proposed CES schemes is given in Tables 1 and 2. We use the following symbols in the tables: n (no. of submessages in original document), m (no. submessages in extracted subdocument), T_{S} (sig. gen. time), T_{V} (sig. ver. time), T_C (commitment generation time), T_{HA} (Hashing time), N (RSA modulus), T^{RSA}_{MULT} (mult. time modulo N), l_{S} (signature length), l_H (hash length), l_C (commitment length), l_r (commitment randomness length), l_T (CES tag length), l_{CEAS} (CEAS encoding length), l_m (submessage length-assumed uniform), l_{seq} (submessage sequence number length). We define $l_R \overset{\text{def}}{=} l_m + l_{CEAS} + l_T + l_{seq}$. For algorithm running time variables the value in brackets denotes the input length. We defined $f(n,m) \overset{\text{def}}{=} \min(mlog_2(n/m), n-m)$. For Extract time of scheme MERSAProd, we only count the signature multiplication step.

Scheme	Approx. Sign Time	Typ. T.S.	S-U Length
CommitVector	$T_{\mathsf{S}}(nl_C + l_{CEAS}) + nT_C(l_m)$	100	$l_{CEAS} + l_{\mathsf{S}} + nl_r$
HashTree	$T_{\mathsf{S}}(l_H + l_{CEAS}) + nT_C(l_m) + (n/2)(T_{HA}(2l_C) + T_{HA}(2l_H))$	100	$l_{CEAS} + l_{\mathsf{S}} + nl_r$
RSAProd	$n(T_{HA}(l_R) + T^{RSA}_{\mathsf{S}})$	1	$l_{CEAS} + l_T + nl^{RSA}_{\mathsf{S}}$
MERSAProd	$O(n \log_2^2 n + \log_2 N)T^{RSA}_{MULT} + nT_{HA}(l_R)$	1	$l_{CEAS} + l_T + l^{RSA}_{\mathsf{S}}$
Trivial	$nT_{\mathsf{S}}(l_R)$	1	$l_{CEAS} + l_T + nl_{\mathsf{S}}$

Table 1: Comparison of signer computation time and signer to user communication overhead. Column 'Typ. T-S' denotes typical Sign Time Saving factor. Column 'S-U Length' denotes Signer-to-User sig. length.

Scheme	Extract Time	Extract Length	Typ. Length	Approx. Verify Time
CommitVector	negligible	$l_{CEAS} + l_{\mathsf{S}} + ml_r +$ $(n-m)l_C$	$169kb$	$T_{\mathsf{V}}(nl_C + l_{CEAS}) + mT_C(l_m)$
HashTree	negligible	$l_{CEAS} + l_{\mathsf{S}} + ml_r +$ $f(n,m)l_C$	$169kb$	$T_{\mathsf{V}}(l_H + l_{CEAS}) + mT_C(l_m) +$ $m(T_{HA}(2l_C) + \log_2(n)T_{HA}(2l_H))$
RSAProd	mT_{MULT}	$l_{CEAS} + l_T + l^{RSA}_{\mathsf{S}}$	$1.28kb$	$T^{RSA}_{\mathsf{V}} + m(T^{RSA}_{MULT} + T_{HA}(l_R))$
MERSAProd	mT_{MULT}	$l_{CEAS} + l_T + l^{RSA}_{\mathsf{S}}$	$1.28kb$	$O(m \log_2 m \log_2 n)T_{MULT} + mT_{HA}(l_m)$
Trivial	negligible	$l_{CEAS} + l_T + ml_{\mathsf{S}}$	$32.3kb$	$mT_{\mathsf{V}}(l_m)$

Table 2: Comparison of user and verifier computation, user to verifier communication overhead.

The 'Typ. Time Saving' (Approximate Sign time saving ratio over simple multiple signature scheme, or over the batched one in the case of scheme MERSAProd) and 'Typ. Length' (referring to extracted signature length) figures were estimated for the following practical example. We took $n = 100$, $m = 99$ (one unextracted submessage), $l_{CEAS} = 100$ bit (a simple CEAS specifying whether a submessage is optional or mandatory for extraction), and $l_m = 64$ bit (typ. numerical fields). For the first two schemes we assumed a DSS signature [5] with $l_{\mathsf{S}} = 320$ bit, and the commitment scheme of [20], using SHA-1 [4] as the collision-resistant hash function ($k = 160$ bit), and an affine universal hash family $h_{A,b} : \{0,1\}^L \rightarrow \{0,1\}^k$ with $h(r) = Ar + b$, $A \in \{0,1\}^{L \times k}$ a Toeplitz matrix, and $b \in \{0,1\}^k$, where arithmetic is performed in the vector space $\{0,1\}^k$ over the binary field $\{0,1\}$. The commit randomness length is $l_r = 2(k+L) - 1$. To achieve a provable upper bound of $2^{-\delta}$ on distinguishing advantage of a privacy-breaking attacker, it is shown in [20] that one can take $L = 3(k+\delta) - 1$. We assumed a reasonable $\delta = 80$, giving randomness length

$l_r = 2(4k + 3\delta - 1) - 1 \approx 1.7kbit$, and commitment length $l_C = 160$ bit. For the RSA schemes, we assumed $|N| = 1024$ bit and $l_T = 160$ bit.

Notice the reversed tradeoff between the commitment-based schemes (efficient in computation but not in signature length) and the RSA schemes (efficient in signature length but not in computation), where in both cases the saving factor over the simple multiple signature scheme is in the order of n.

5 Conclusion

Motivated by emerging needs in online interactions and the need to embrace a minimal information model, we introduced a *Content Extraction Signature* (CES) as a digital signature which can be verified with knowledge of only selected extracted portions of the signed document (while hiding unextracted portions), at a communication or computation cost which is lower than the simple multiple signature solution. We defined the security and functional requirements for such signatures, and proposed four CES schemes with various performance tradeoffs. Practical application and implementation issues of our CES schemes are the subject of a forthcoming paper.

References

[1] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO '98*, LNCS 1462.

[2] G.C. Chick and S.E. Tavares. Flexible Access Control with Master Keys. In *CRYPTO '89.*

[3] R. Steinfeld. A Pseudo-Random Generator with Seed Extraction. Manuscript, Sep. 2001.

[4] NIST. Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-1. April 1995.

[5] NIST. Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186. November 1994.

[6] MasterCard and VISA. Secure Electronic Transaction (SET) Specification Books 1-3 (Version 1.0). May 31, 1997.

[7] XML Core Working Group. XML-Signature Syntax and Processing: W3C Proposed Recommendation. August 20, 2001. Available from `http://www.w3.org/TR/xmldsig-core`.

[8] M. Bellare and J.A. Garay and T. Rabin. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In *EUROCRYPT '98*, LNCS 1403. Springer-Verlag, Berlin, 1998.

[9] J.S. Coron and D. Naccache. On the Security of RSA Screening. In *PKC '99*, LNCS 1560. Springer-Verlag, Berlin, 1999.

[10] D. Naccache and D. M'Raihi and S. Vaudenay and D. Raphaeli. Can D.S.A be improved? In *EUROCRYPT '94*, LNCS 950. Springer-Verlag, Berlin, 1999.

[11] A. Fiat. Batch RSA. In *CRYPTO '89*, LNCS 435. Springer-Verlag, Berlin, 1990.

[12] H. Shacham and D. Boneh. Improving SSL Handshake Performance via Batching. In *CT-RSA 2001*, LNCS 2020. Springer-Verlag, Berlin, 2001.

[13] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *EUROCRYPT '96*, LNCS 1070. Springer-Verlag, Berlin, 1996.

[14] M. Bellare and O. Goldreich and S. Goldwasser. Incremental Cryptography: The Case of Hashing and Signing. In *CRYPTO '94*, LNCS 839, Springer-Verlag, Berlin, 1994.

[15] M. Bellare and O. Goldreich and S. Goldwasser. Incremental Cryptography and Application to Virus Protection. In *Proc. of 27th STOC* ACM, 1995.

[16] C.J. Pavlovski and C. Boyd. Efficient Batch Signature Generation Using Tree Structures. In *CrypTEC'99*. City University of Hong Kong Press, 1999.

[17] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. of Computer and System Sciences*, pages 270-299, vol. 28, no. 2, 1984.

[18] A. Menezes and P. van Oorschot and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.

[19] S. Goldwasser and S. Micali and R. Rivest. A Digital Signature Scheme Secure against Adaptively Chosen Message Attacks. *SIAM Journal on Computing*, pages 281-308, vol. 17, no. 2, 1988.

[20] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *CRYPTO '96*, LNCS 1109. Springer-Verlag, Berlin, 1996.

A Appendix

The following definition and Lemmas are required for security results stated in the paper. Refer to the full paper for proofs. The following definition is for a weaker notion of unforgeability than the full one defined in the paper.

Definition 1 *A CES scheme $\mathcal{D}$ is said to have the* weak ordered *(WO) CES-unforgeability property if the following holds: It is infeasible for an attacker, having access to a CES signing oracle, to produce a document/signature pair $(\mathbf{M}, \sigma)$, such that: (i) σ passes the verification test for $\mathbf{M}$ and (ii) $\mathbf{M}$ contains a submessage at some position i which has never appeared in position i in any document queried to the CES signing oracle.*

Lemma 1 *Any CES scheme which has the weak ordered CES-Unforgeability property can be used to construct a CES scheme which has the full CES-unforgeability property.*

The construction used in proving Lemma 1 involves choosing a unique 'CES tag' for each signed document and appending it (and a A CEAS encoding) to each submessage before applying the original CES Sign algorithm.

The following lemmas were also used in the security analysis of the RSA schemes.

Lemma 2 *(Bellare-Rogaway[13]) If $H(.)$ is a random oracle onto $\mathbb{Z}_N^*$ and the RSA function is one-way, then the FDH signature $\mathsf{S}(m) \stackrel{\mathrm{def}}{=} H(m)^d \bmod N$ is existentially unforgeable under chosen message attack.*

Lemma 3 *If $H(.)$ is a random oracle onto $\mathbb{Z}_N^*$ and the RSA function is one-way for each e_i $(i = 1, ..., n)$, then the n distinct-exponent FDH signature schemes $\mathsf{S}_\mathsf{i}(m) \stackrel{\mathrm{def}}{=} H(i\|m)^{d_i} \bmod N$ (for $i = 1, ..., n$) are existentially unforgeable even under a chosen message attack giving acccess simultaneously to all the n signing oracles $\mathsf{S}_\mathsf{i}(.)$.*