Content Extraction Signatures*

Ron Steinfeld†
Department of Computing, Macquarie University,
North Ryde 2109 Australia
rons@ics.mq.edu.au

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

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

April 28, 2003

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

*This is a full and extended version of an earlier paper presented at ICISC 2001, Dec. 6-7 2001, Seoul, South Korea.

†This work was done while the author was at the School of Network Computing, Monash University, Frankston, Australia.

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

Consider the retrieval and exchange of individual pieces of information contained within a given document, rather than the handling of the whole document. Underlying this activity is the notion that the document itself is not the most important thing to the holder in such cases. The value, rather, lies with the document content, i.e. its constituent pieces of information. Furthermore, this concept is not constrained to small containers of information such as letters, contracts, bank statements or receipts. It is also scalable to very large collections such as books or encyclopedias.

There are everyday situations where, for reasons of relevance or privacy, one wants to disclose only certain parts of a document. Under the current digital signature regime, however, the entire document is signed, thereby forcing the document holder to disclose all of its contents to a third party for the signature to be verifiable.

Stewart Baker, chief counsel for America's National Security Agency (NSA), has warned of the threats to our privacy in an electronic society [2]:

> The biggest threats to our privacy in a digital world come not from what we keep secret but from what we reveal willingly. We lose privacy in a digital world because it becomes cheap and easy to collate and transmit data... Restricting these invasions of privacy is a challenge, but it isn't a job for encryption. Encryption can't protect you from the misuse of data you surrendered willingly.

Blakley claims 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 [11].

We agree with these views and with the work reported in this paper, propose a means of reducing the erosion of a document holder's privacy in real-world electronic interactions.

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

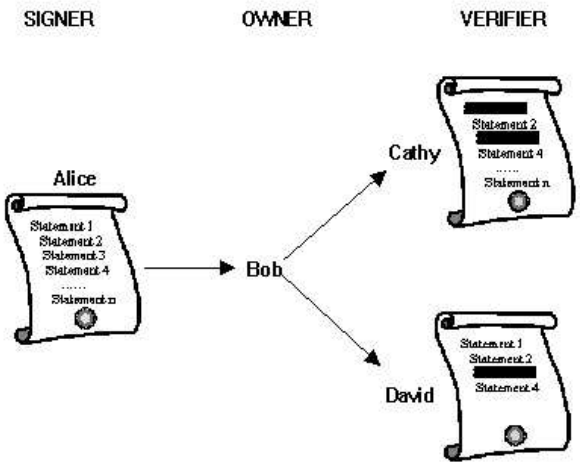

Figure 1: An example of a multiparty interaction.

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

This paper is organized as follows. In Section 3, we give preliminary definitions and results which we need in this paper. In Section 2 we explain the real-life background motivating the introduction of Content Extraction Signatures, and review related work. In Section 4, 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 5 presents four CES schemes meeting our definition and for which we provide rigorous proofs of security under well-defined and known cryptographic assumptions. Our security proofs are *concrete*, meaning they provide quantitative upper bounds on the insecurity of our schemes in terms of the

insecurity of the underlying cryptographic primitives. 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 7 we close with some concluding comments.

2 Background

2.1 The Emerging Need

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.

We give practical examples of each of these situations.

2.1.1 Privacy Example

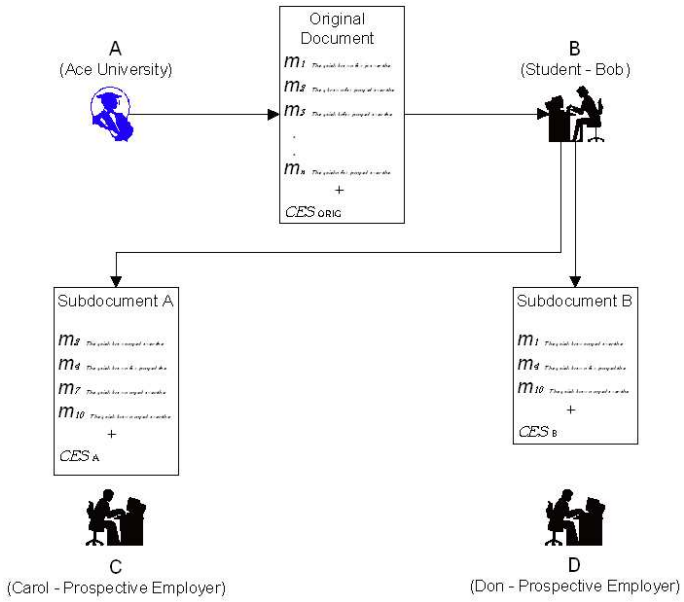

Figure 2: A real-life application for Content Extraction Signatures.

Consider the commonplace although pertinent example illustrated in Fig. 2. 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. This example demonstrates the tension between verifiable information granularity and the information holder's privacy, as illustrated in Fig. 3.

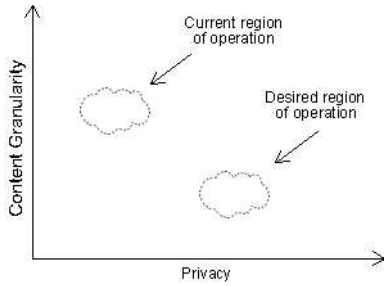

Figure 3: Verifiable content granularity versus information holder's privacy.

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.

2.1.2 Bandwidth Example

The granularity of signed information in multiparty interactions does not only affect privacy; it also causes unnecessary bandwidth usage. Consider Bob, a document holder, who wants to pass a single item of verifiable information to Carol. Instead of being able to pass this single item of information, Bob is forced to furnish the entire document, which could be significantly greater in size than the single item. If he does not, Carol will not be able to verify the signature.

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 Fig. 4, the PM's office issues a transcript in two languages such as French and English (as in Canada), of an interview involving the PM, which has been signed using the standard digital signature.

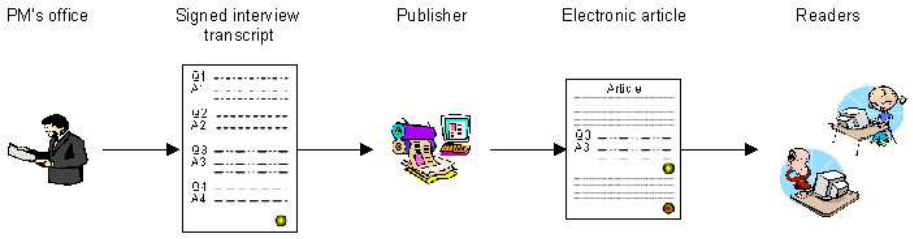

Figure 4: Example of electronic publishing which includes verifiable content.

Since the publisher has tight constraints on article size and is focussing on a particular language and topic, it is neither appropriate, nor possible, to include the entire transcript of the interview along with all language translations. Therefore, the publisher would like to quote only the PM's response to a particular question in the required translation. It is desirable for the reader to be able to verify the quoted content in the article from the signed transcript. This would avoid problems of misinterpretation and misquoting.

If the interview transcript is signed using the standard digital signature, this scenario is not possible as the standard digital signature requires all of the signed information to be present, otherwise it will fail verification by the reader.

This example illustrates the tension that exists between verifiable content granularity and bandwidth, as illustrated in Fig. 5. 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 thereby move along the tension curve towards reduced bandwidth.

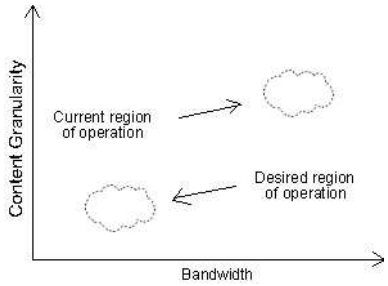

Figure 5: Verifiable content granularity versus bandwidth usage.

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

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.

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 §4), 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 [7] 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 [7] 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 §4), 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 [28], 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' [3] proposed recommendation of the WWW Consortium (W3C) defines a scheme which is most similar to our scheme CommitVector described in § 5.1.1, that 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.

The substantial work of Brands [13, 14] on *digital credentials* also aims to enhance the privacy of signed document owners, and provides protocols for verifiable selective disclosure of signed credentials. However, unlike our RSA-based schemes, the communication efficiency requirement is not addressed by Brands' schemes, and indeed they involve communication overhead which is, in the worst case, linear in the number of signed credentials (corresponding to submessages in our model), as for the multiple signature scheme. Furthermore, unlike our commitment-based schemes which are built from general cryptographic primitives, Brand's schemes are based on specific number-theoretic assumptions.

Three of our CES schemes are modifications of batch signature generation and verification schemes [32], [6], [21]. 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 [21] mentions another application of the length-saving feature of his batching scheme for the purpose of saving bandwidth in distributed decryption.

Shortly after the publication of the preliminary version of the results in this paper [37], Johnson, Molnar, Song and Wagner have independently proposed in [27] the notion of 'redactable signature' schemes which correspond quite closely to our definition of Content Extraction Signatures, although the privacy security notion is not formalized in [27]. They also proposed two schemes to implement their definition. The first scheme in [27] resembles our 'Hash Tree' scheme (HT), and improves the communication cost of the commitment randomness strings using what we call a 'PseudoRandom Generator with Seed Extraction' (see the remark in § 5.1.2). This PRNG is derived from the tree-based construction of a pseudorandom function from any pseudorandom generator due to Goldreich, Goldwasser and Micali [23]. Although this construction reduces the communication overhead in some cases it does not, however, improve the worst-case overhead over that of our scheme HT. The second scheme in [27] resembles our scheme MERP in costs and in the use of Multi-Exponent RSA. However, whereas our scheme MERP is derived from the $\mathsf{FDH}-\mathsf{RSA}$ standard signature scheme, the scheme in [27] is derived from the standard RSA signature of Gennaro, Halevi and Rabin [22].

Finally, we remark that the *aggregate signature* scheme proposed recently by Boneh Gentry, Lynn and Shacham [12] can be readily converted into a communication-saving CES scheme based on the

Computational Co-Diffie-Hellman (co-CDH) assumption in Gap Diffie-Hellman (GDH) Groups. It has a similar 'product' structure to our RSA-based CES schemes.

3 Preliminaries

3.1 General Notations

We use $[n]$ to denote the set of integers $\{1, 2, \ldots, n\}$. We use $\exp : \mathbb{R} \rightarrow \mathbb{R}$ to denote the natural exponentiation function. We use the notation $\mathsf{A}(.,.)$ to denote an algorithm, with input arguments separated by commas (our underlying computational model is a probabilistic Turing Machine). If the algorithm A is randomized we write the random input of A separated by the symbol ';' from the rest of the inputs to A. If algorithm A makes calls to oracles, we list the oracles separated from the algorithm inputs by the symbol '|'. Given a set S we denote by $s \xleftarrow{R} S$ the assignment of a uniformly and independently distributed random element from the set S to the variable s. We use the notation $\Pr[\mathsf{Event}]_{exp}$ to denote the probability of event Event in experiment *exp*.

Following the practice of modern cryptology, we will give a *concrete* security analysis of the cryptographic schemes proposed in this paper (see [4] for an introduction to this approach). Namely, For each cryptographic scheme or primitive (say scm) discussed in the paper, we will define a *security notion* (say sec), which specifies an attack model on the scheme and what the attacker needs to achieve for a 'successful' attack in the sense of the security notion. The scheme will have an integer security parameter k which allows to control the security achieved by the scheme (e.g. by controlling the length of secret keys). We define the *insecurity function* $\mathbf{InSec}^{\mathrm{sec}}_{\mathsf{scm}}(r_1(k), \ldots, r_n(k))$ to measure how insecure the scheme scm is in the sense of the security notion sec, as a function of the attacker's *resource parmeters* $(r_1(k), \ldots, r_n(k))$, which are functions of k (these resource parameters include quantities such as the run-time+program size of the attacker, number of oracle queries of attacker, length of messages sent by attacker). We call an attacker *efficient* if each of its resource parameters $r_1(k), \ldots, r_n(k)$ is upper bounded by a *polynomial* function of k (i.e. there exists k_0 and c such that $r_i(k) < k^c$ for all $k > k_0$). We say that scheme scm achieves the security notion sec if $\mathbf{InSec}^{\mathrm{sec}}_{\mathsf{scm}}(r_1(k), \ldots, r_n(k))$ is a negligible function of k for all efficient attackers (a function $f(k)$ is called *negligible* if for each $c > 0$, there exists k_0 such that $f(k) < 1/k^c$ for all $k > k_0$).

3.2 Standard Cryptographic Primitives

In this section we give several definitions of standard cryptographic primitives which are used as building blocks to construct CES schemes in §5.

3.2.1 Digital Signature Schemes

A standard digital signature scheme $\mathsf{DS} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ consists of three algorithms. On input a security parameter k, the key-generation algorithm GK outputs a secret/public key-pair (sk, pk). On input a secret key sk and a message m, the signing algorithm S outputs a signature σ. On input a public key pk, a message m and a signature σ, the verifying algorithm V outputs a signature validity decision $d \in \{Acc, Rej\}$. The standard unforgeability notion for digital signature schemes is 'existential unforgeability under adaptive chosen message attack' [25], called CMA, prevents the forging of a

signature for any 'new' message which was not signed by the signing oracle available to the attacker. The precise definition follows.

Definition 3.1 (CMA**).** *Let* $\mathsf{DS} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *be a digital signature scheme. Let* A *be an attack algorithm. Define the experiment*

Experiment **CMAExp**$(\mathsf{DS}, \mathsf{A}, k)$
  $(sk, pk) \leftarrow \mathsf{K}(k)$
  $(m^*, \sigma^*) \leftarrow \mathsf{A}(k, pk | \mathsf{S}(sk, .))$
  $d \leftarrow \mathsf{V}(pk, m^*, \sigma^*)$
  `If` $d = Acc$ and for all j,
  $m^* \neq m_j$
    `Return` 1
  `Else`
    `Return` 0

Here, we denote by m_j *the* j*th message queried by* A *to its* S *oracle. We quantify* A*'s success in breaking the* CMA *security notion of scheme* DS *by the probability* $\mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A},\mathsf{DS}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathbf{CMAExp}(\mathsf{DS}, \mathsf{A}, k) = 1]$. *We define* A*'s resource parameters as* $RP = (t, q_s, \ell)$ *if* A *has running time/program size at most* t*, and makes at most* q_s *message queries to the* S *oracle, each of length at most* ℓ *bits. We quantify the insecurity of scheme* DS *in the sense of* CMA *against arbitrary attackers with resource parameters* $RP = (t, q_s, \ell)$ *by the probability*

$$\mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{DS}}(t, q_s, \ell) \stackrel{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A},\mathsf{DS}}.$$

The attacker set AS_{RP} *contains all attackers with resource parameters* $RP = (t, q_s, \ell)$.

3.2.2 Message Commitment Schemes

A message commitment scheme $\mathsf{CM} = (\mathsf{I}, \mathsf{C})$ consists of two algorithms. On input a security parameter k, the initialization algorithm I outputs a scheme parameter string sp. On input a scheme parameter string sp, a message string m, and a random string r, the commitment algorithm C outputs a commitment string c for the message m (if r is too short, C terminates with an error). The commitment algorithm is also used to *verify* that a given string c is a valid commitment of a message m under random string (randomness) input r by recomputing $c' = \mathsf{C}(sp, m; r)$ and verifying that $c' = c$. In the following, where only one commitment scheme is used we will use the simpler function notation $C(m; r)$ to mean $\mathsf{C}(sp, m; r)$, where sp is understood by context.

A commitment scheme satisfies two security notions: The *hiding* notion and the *binding* notion.

The *hiding* notion ensures that the commitment c to a message m under random string r does not leak any information on the message if r is not given, i.e. the message m is 'semantically secure' [24]. The precise indistinguishability-based definition follows.

Definition 3.2 (Hiding). *Let* $\mathsf{CM} = (\mathsf{I}, \mathsf{C})$ *denote a message commitment scheme. Define the experiment*

Experiment **HideExp**$(\mathsf{CM}, \mathsf{A} = (\mathsf{A}_{\mathsf{find}}, \mathsf{A}_{\mathsf{guess}}), k)$

$sp \leftarrow \mathsf{I}(k)$
$(s, m_0, m_1) \leftarrow \mathsf{A}_{\mathsf{find}}(k, sp)$
$b \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}$
$c^* \leftarrow C(m_b; r^*)$
$b' \leftarrow \mathsf{A}_{\mathsf{guess}}(s, c^*)$
If $b' = b$ `Return` 1
Else `Return` 0

We quantify A*'s success in breaking the* Hiding *security notion of scheme* CM *by the advantage* $\mathbf{Succ}^{\mathrm{hide}}_{\mathsf{A},\mathsf{CM}}(k) \stackrel{\mathrm{def}}{=} 2(\Pr[\mathbf{HideExp}(\mathsf{CM}, \mathsf{A}, k) = 1] - \frac{1}{2})$*. We define* A*'s resource parameters as* $RP = (t, \ell)$ *if* A *has running time/program size at most* t*, and messages* (m_0, m_1) *have length at most* ℓ *bits. We quantify the insecurity of scheme* CM *in the sense of* Hiding *against arbitrary attackers with resource parameters* $RP = (t, \ell)$ *by the probability*

$$\mathbf{InSec}^{\mathrm{hide}}_{\mathsf{CM}}(t, \ell) \stackrel{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{Hid}}_{\mathsf{A},\mathsf{CM}}.$$

The attacker set AS_{RP} *contains all attackers with resource parameters* $RP = (t, \ell)$*.*

The relaxed-binding (*r-binding*) notion ensures that after a commitment c to an attacker-chosen message m_1 under a (truly) *random* string r_1 has been produced, it is infeasible for the attacker to 'open' the commitment with a new message $m_2 \neq m_1$, i.e. find a matching random string input r_2 such that c is also the commitment to m_2 under random string r_2 (otherwise the attacker can fool someone who verifies that $c = \mathsf{C}(sp, m_2; r_2)$ that c was a commitment for m_2).

The *r-binding* notion (using the same terminology) has been also defined in [1]. Note that *r-binding* is a weaker requirement from the commitment algorithm than full *binding* (collision-resistance), because in *r-binding* r_1 is randomly chosen, and not under the control of the attacker. We do not give the details of the definition of the full *binding* notion since they are straightforward (and similar to collision-resistance of hash functions defined below)— the only difference from *r-binding* is that r_1 is chosen by the attacker. The corresponding insecurity function is denoted $\mathbf{InSec}^{\mathrm{bind}}_{\mathsf{CM}}(t, \ell)$.

Definition 3.3 (Relaxed Binding: $\mathrm{r-binding}$**).** *Let* $\mathsf{CM} = (\mathsf{I}, \mathsf{C})$ *be a message commitment scheme. Let* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *be an attack algorithm. Define the experiment*

Experiment $\mathbf{RBindExp}(\mathsf{CM}, \mathsf{A}, k)$
$sp \leftarrow \mathsf{I}(k)$
$(s, m_1) \leftarrow \mathsf{A}_1(k, sp)$
$C^* \leftarrow C(m_1; r_1)$
$(m_2, r_2) \leftarrow \mathsf{A}_2(s, C^*, r_1)$
If $C(m_1; r_1) = C(m_2; r_2)$,
  `Return` 1
`Else`
  `Return` 0

We quantify A*'s success in breaking the* $\mathrm{r-binding}$ *security notion of scheme* CM *by the probability* $\mathbf{Succ}^{\mathrm{r-bind}}_{\mathsf{A},\mathsf{CM}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathbf{BindExp}(\mathsf{CM}, \mathsf{A}, k) = 1]$*. We define* A*'s resource parameters as* $RP = (t, \ell)$ *if*

A *has running time/program size at most* t*, and messages* m_1 *and* m_2 *are each of length at most* ℓ *bits. We quantify the insecurity of scheme* CM *in the sense of* r − bind *against arbitrary attackers with resource parameters* $RP = (t, \ell)$ *by the probability*

$$\mathbf{InSec}^{\mathrm{r-bind}}_{\mathsf{CM}}(t, \ell) \stackrel{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{r-bind}}_{\mathsf{A},\mathsf{CM}}.$$

The attacker set AS_{RP} *contains all attackers with resource parameters* $RP = (t, \ell)$*.*

3.2.3 Collision-Resistant Hash Functions

A Collision Resistant Hash Function Family (CRHF) $\mathsf{CR} = (\mathsf{KH}, \mathsf{H}())$ consists of two algorithms. On input a security parameter k, the key generation algorithm KH outputs a hash function key α(specifying a unique function from the family). On input a key α and string m, the hashing algorithm H outputs a hash string $h \in \{0,1\}^{l_H}$ for some output bit length $l_H(k)$. In the following, where only one hash function is used we will use the simpler function notation $H(m)$ to mean $\mathsf{H}(\alpha, m)$, where the key α is understood by context.

The *collision-resistance* security notion for a CRHF is that it is infeasible, given the random function key α to find any two distinct messages $m_1 \neq m_2$ with the same hash value: $H(m_1) = H(m_2)$. Due to the similarity with the binding notion of commitment schemes above we do not write the details of the full definition here. The corresponding insecurity function is denoted $\mathbf{InSec}^{\mathrm{cr}}_{\mathsf{CR}}(t, \ell)$.

3.3 RSA-Based Signatures

In the security analysis of our RSA-based CES schemes, we make use of the following definitions and results.

3.3.1 The RSA Trapdoor One-Way Function

The well-known RSA trapdoor one-way function $\mathsf{RSA} = (\mathsf{GK}, \mathsf{f}, \mathsf{f}^{-1})$, introduced in [33], is defined as follows. On input a security parameter k, the key generation algorithm GK outputs a secret/public key pair (sk, pk). The public key $pk = (N, e)$ consisting of an RSA modulus $N = pq$, where p and q are random $k/2$-bit primes, and e is an integer satisfying $\gcd(e, \phi) = 1$, where $\phi = (p-1)(q-1)$. The secret trapdoor key $sk = d$ is the integer $d = e^{-1} \bmod \phi$. Given the public key (N, e) and input $x \in \mathbb{Z}^*_N$, the RSA function evaluation algorithm f outputs $y = \mathsf{f}((N, e), x) = x^e \bmod N$. Given the secret trapdoor d, and an input $y \in \mathbb{Z}^*_N$ the RSA function inversion algorithm f^{-1} outputs $x = \mathsf{f}^{-1}((N, e, d), y) = y^d \bmod N$.

There are many variants of the RSA function depending on the exact key generation algorithm GK. In this paper, we will denote by RSA the RSA function whose key generation algorithm is that of CES scheme RSAP (see § 5.2.1). We will use $\mathsf{RSA_i}$ to denote the RSA function variant whose key generation is that of CES scheme MERP (see § 5.2.2), except that whereas scheme MERP has a public exponent vector $\mathbf{e}$ consisting of the first n_B odd primes, the function $\mathsf{RSA_i}$ uses just the ith odd prime in this vector $\mathbf{e}[i]$ as its (single) public exponent. Of course, the corresponding secret exponent for $\mathsf{RSA_i}$ is $\mathbf{d}[i] = \mathbf{e}[i]^{-1} \bmod \phi$.

The *One-Wayness* security notion (OW) for the RSA function is that it is infeasible for an attacker A, given a random public key (N, e) output by GK (with corresponding secret key (N, e, d)) and

a uniformly random $y \in \mathbb{Z}_N^*$, to invert the RSA function and output $x = y^d \bmod N$. A precise definition follows.

Definition 3.4 (RSA One-Wayness: OW). *Let* $\mathsf{RSA} = (\mathsf{GK}, \mathsf{f}, \mathsf{f}^{-1})$ *be an RSA function. Let* A *be an attack algorithm. Define the experiment*

Experiment $\mathbf{OWExp}(\mathsf{RSA}, \mathsf{A}, k)$
$((N, e), d) \leftarrow \mathsf{GK}(k)$
$y \xleftarrow{\mathrm{R}} \mathbb{Z}_N^*$
$x' \leftarrow \mathsf{A}(k, (N, e))$
if $x' = y^d \bmod N$ `Return` 1
else `Return` 0

We quantify A*'s success in breaking the One-Wayness (*OW *notion) of* RSA *by the probability* $\mathbf{Succ}^{\mathrm{OW}}_{\mathsf{A},\mathsf{RSA}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathbf{OWExp}(\mathsf{RSA}, \mathsf{A}, k) = 1]$. *We define* A*'s resource parameters(RPs) as* $RP = (t)$ *if* A *has running time/program size at most* t. *We quantify the insecurity of* RSA *in the sense of* OW *against arbitrary attackers with RPs* $RP = (t)$ *by the probability* $\mathbf{InSec}^{\mathrm{OW}}_{\mathsf{RSA}}(t) \stackrel{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{OW}}_{\mathsf{A},\mathsf{RSA}}$ *The set* AS_{RP} *contains all attackers RPs* $RP = (t)$.

3.3.2 The FDH-RSA Signature Scheme

The $\mathsf{FDH}-\mathsf{RSA}$ signature, formally introduced in [10], follows the classical 'hash-then-invert' paradigm for building a signature from a trapdoor one-way function (in this case RSA). It uses a hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$. In the 'standard' security model which corresponds most closely to practice, $H(.)$ is built from a collision-resistant hash function like SHA-1 [31], and the algorithm for $H(.)$ is published in the public key. In the random-oracle model [9], we model $H(.)$ as a random function chosen uniformly from the set of all functions from $\{0,1\}^*$ to $\mathbb{Z}_N^*$, and allow the attacker only black-box oracle access to $H(.)$.

Definition 3.5 (Sig. Scheme $\mathsf{FDH}-\mathsf{RSA}$). *Let* $\mathsf{RSA} = (\mathsf{GK}, \mathsf{f}, \mathsf{f}^{-1})$ *be an RSA function. The scheme* $\mathsf{FDH}-\mathsf{RSA} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *is defined as follows.*

Algorithm $\mathsf{K}(k)$
$(d, (N, e)) \leftarrow \mathsf{GK}(k)$
`Return` $((N, e, d), (N, e))$

Algorithm $\mathsf{S}((N, e, d), m)$
$\sigma \leftarrow H(m)^d \bmod N$
`Return` σ

Algorithm $\mathsf{V}((N, e), m, \sigma)$
if $\sigma^e \equiv H(m) \bmod N$
`Return` Acc
else `Return` Rej.

Note that if RSA_i is used (see § 3.3.1) as the underlying RSA function we call the signature scheme $\mathsf{FDH}-\mathsf{RSA}_i$.

In the random-oracle model, the CMA unforgeability of $\mathsf{FDH}-\mathsf{RSA}$ can be proven assuming the One-Wayness of the RSA function. We will use the following efficient concrete result of this type, due to Coron [18] (a less efficient reduction was given by Bellare and Rogaway [10]).

Lemma 3.1 (Coron). *If the* RSA *function is One-Way (*OW *notion) then the signature scheme* $\mathsf{FDH}-\mathsf{RSA}$ *is existentially unforgeable under chosen message attack (*CMA *notion) in the random-oracle model. Concretely:*

$$\mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{FDH}-\mathsf{RSA}}(t, q_s, \ell, q_H) \leq \exp(1) \cdot (q_s + 1) \cdot \mathbf{InSec}^{\mathrm{OW}}_{\mathsf{RSA}}(t') \tag{1}$$

where $t' = t + (q_H + q_s + 1) \cdot O(k^3)$.

3.3.3 The FDH-MER Signature Scheme

The $\mathsf{FDH-MER}$ signature, is a 'Multi-Exponent RSA' variant of $\mathsf{FDH-RSA}$. It is not a well known signature but is formally introduced as part of the security analysis of one of our CES schemes MERP in § 5.2.2. It uses the same key generation GK as MERP, having public key $(N, \mathbf{e})$ and secret key $(N, \mathbf{e}, \mathbf{d})$, with $\mathbf{e}[i]$ denoting the ith odd prime and $\mathbf{d}[i] = \mathbf{e}[i]^{-1} \bmod \phi$. In this scheme the message space is $[n_B] \times \{0,1\}^*$ (i.e. messages are (integer,string) pairs). The integer prefix of a message is used to select the secret exponent (from the vector $\mathbf{d}$) used to RSA-invert the hash of the message during signing, as follows.

Definition 3.6 (Sig. Scheme $\mathsf{FDH-MER}$). *Let* GK *denote the key generation algorithm for scheme* MERP *(see § 5.2.2). The scheme* $\mathsf{FDH-MER} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *is defined as follows.*

Algorithm $\mathsf{K}(k)$	Algorithm $\mathsf{S}((N, \mathbf{e}, \mathbf{d}), (k, m))$	Algorithm $\mathsf{V}((N, \mathbf{e}), (k, m), \sigma)$
$((N, \mathbf{e}), (N, \mathbf{e}), \mathbf{d}) \leftarrow \mathsf{GK}(k)$	$\sigma \leftarrow H(k, m)^{\mathbf{d}[k]} \bmod N$	if $\sigma^{\mathbf{e}[k]} \equiv H(k, m) \bmod N$
`Return` $((N, \mathbf{e}, \mathbf{d}), (N, \mathbf{e}))$	`Return` σ	`Return` Acc
		else `Return` Rej.

When $H(.)$ is a cryptographic hash function, it seems likely that $\mathsf{FDH-MER}$ is CMA-secure as long as $\mathsf{FDH-RSA}_i$ is CMA-secure for all $i \in [n_B]$. However we cannot prove this statement in the standard model for $H(.)$ because we cannot use a signing oracle for exponent i ($\mathsf{FDH-RSA}_i$) to compute $H(k, m)^{1/e_k}$ for $k \neq i$ (in order to answer signature queries of an attacker on $\mathsf{FDH-MER}$). However, we can do so in the random-oracle model for $H(.)$ by setting $H(k, m)$ to random elements whose inverse relative to e_k is known. By composing with Coron's reduction for $\mathsf{FDH-RSA}$ (see above), we get the following result.

Lemma 3.2 (Security of $\mathsf{FDH-MER}$). *If the* RSA *function is One-Way for all* n_B *first odd primes as public exponents (i.e.* RSA_i *satisfies* OW *notion for all* $i \in [n_B]$*) then the signature scheme* $\mathsf{FDH-MER}$ *is existentially unforgeable under chosen message attack (*CMA *notion) in the random-oracle model. Concretely:*

$$\mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{FDH-MER}}(t, q_s, \ell, q_H) \leq \exp(1) \cdot (q_s + 1) \cdot \left(\sum_{i \in [n_B]} \mathbf{InSec}^{\mathrm{OW}}_{\mathsf{RSA}_i}(t') \right) \tag{2}$$

where $t' = t + O((q_H + q_s)^2 \cdot (\ell + \log n_B) + [1 + q_H(q_H + q_s)]k^3)$.

Proof. We show how to use an efficient attacker A breaking scheme $\mathsf{FDH-MER}$ in the sense of CMA (in the ROM) to construct n_B efficient attackers A_r (for $r \in [n_B]$), where attacker A_r breaks scheme $\mathsf{FDH-RSA}_r$ in the sense of CMA (in the ROM), such that the sum of success probabilities of the A_r's is lower bounded as

$$\sum_{r \in [n_B]} \mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A}_r, \mathsf{FDH-RSA}_r}(k) \quad \geq \quad \mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A}, \mathsf{FDH-MER}}(k), \tag{3}$$

where A_r has RPs $(t[r], q_s[r], \ell[r], q_H[r])$ with $t[r] = t + O((q_H + q_s)^2(\ell + \log n_B) + q_H(q_H + q_s)k^3)$; $q_s[r] = q_s$; $\ell[r] = \ell$; $q_H[r] = q_H$. Applying Coron's Lemma 3.1 for each $r \in [n_B]$ we have:

$$\mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A}_r,\mathsf{FDH}-\mathsf{RSA}_r}(k) \leq \exp(1) \cdot (q_s + 1) \cdot \mathbf{InSec}^{\mathrm{OW}}_{\mathsf{RSA}_r}(t_R[r]), \tag{4}$$

where $t_R[r] = t[r] + O(k^3(q_H + q_s + 1)) = t + O((q_H + q_s)^2 \cdot (\ell + \log n_B) + [1 + q_H(q_H + q_s)]k^3)$. Substituting (4) in (3) gives the desired bound (2). So it remains to describe A_r and show it satisfies (3).

Attacker A_r works as follows. A_r is invoked by the CMA experiment for $\mathsf{FDH}-\mathsf{RSA}_r$ on input $(k, (N, \mathbf{e}[r]))$. A_r is able to query messages to its random oracle $H_r(.)$, and to its signing oracle S_r for scheme $\mathsf{FDH}-\mathsf{RSA}_r$. Note that as A_r runs it maintains in memory a list HList (initially empty) with entries of the form $((\bar{k}_j, \bar{m}_j), \bar{h}_j, \bar{\sigma}_j)$. We assume (without loss of generality) that A always queries a message to $H(.)$ before it queries it to S.

Setup. For $i \in [n_B]$, A_r sets $\mathbf{e}[i]$ to equal the ith odd prime. It then runs A on input $(k, (N, \mathbf{e}))$.

$H(.)$ **Queries.** When A makes the jth query $(\bar{k}_j, \bar{m}_j) \in \mathbb{Z} \times \{0,1\}^*$ to its random-oracle $H(.)$, A_r simulates A's random function $H(.)$ such that it coincides with A_r's random function $H_r(.)$ on inputs with integer prefix r, i.e. $H(r, m) = H_r(r, m)$ for any $m \in \{0,1\}^*$. The details follow. There are four cases. If $\bar{k}_j = r$, A_r just queries $(\bar{k}_j, \bar{m}_j)$ to its $H_r(.)$ oracle to get response $\bar{h}_j$ and forwards $\bar{h}_j$ to A. Else, if $(\bar{k}_j, \bar{m}_j)$ already appears in the HList (i.e. is equal to $(\bar{k}_i, \bar{m}_i)$ for some $i \in [q_H + q_S]$) then A_r returns $\bar{h}_i$ from the HList entry for $(\bar{k}_i, \bar{m}_i)$ to A. Else, if $\bar{k}_j \in [n_B]$, A_r prepares a simulated signature $\bar{\sigma}_j$ (in case it will be needed in future) by picking $\bar{\sigma}_j$ uniformly at random from $\mathbb{Z}_N^*$, computes $\bar{h}_j = \bar{\sigma}_j^{\mathbf{e}[\bar{k}_j]} \bmod N$, adds entry $((\bar{k}_j, \bar{m}_j), \bar{h}_j, \bar{\sigma}_j)$ to HList, and returns $\bar{h}_j$ to A. Else $(k_j \notin [n_B])$, A_r just picks a random $\bar{h}_j$, adds entry $((\bar{k}_j, \bar{m}_j), \bar{h}_j)$ to HList, and returns $\bar{h}_j$ to A.

S **Queries.** When A makes the jth query (k_j, m_j) to its signing oracle S, A_r answers by simulating S as follows. There are two cases. If $k_j = r$, A_r just queries its sig. oracle S_r on input (k_j, m_j) and forwards the answered sig. $\sigma_j = H(r, m_j)^{1/\mathbf{e}[r]} \bmod N$ to A. Else, if $k_j \in [n_B] - \{r\}$, A_r searches the HList for the (unique) entry $((\bar{k}_i, \bar{m}_i), \bar{h}_i, \bar{\sigma}_i)$ such that $(\bar{k}_i, \bar{m}_i) = (k_j, m_j)$, and returns to A the simulated signature $\sigma_j = \bar{\sigma}_i = H(\bar{k}_i, \bar{m}_i)^{1/\mathbf{e}[\bar{k}_i]}$. Note that in either case, $\sigma_j = H(k_j, m_j)^{1/\mathbf{e}[k_j]}$, as required.

Output. Eventually A terminates and outputs a forgery pair $((k^*, m^*), \sigma^*)$ for $\mathsf{FDH}-\mathsf{MER}$. Then A_r just forwards $((k^*, m^*), \sigma^*)$ as its forgery pair for $\mathsf{FDH}-\mathsf{RSA}_r$.

This completes the description of A_r. The claimed RPs $q_s[r]$, $\ell[r]$ and $q_H[r]$ are immediately clear from this description. The running time bound $t[r]$ is obtained as follows. For each answered $H(.)$ or S query, A_r searches the HList, having at most $(q_H + q_s)$ entries each of bit-length at most $\ell + \log n_B$. This takes time $O((q_H + q_s)(\ell + \log n_B))$ per query. Also, for each H query A_r performs an exponentiation mod N with exponent of bit length $O(\log \mathbf{e}[n_B])$. It is known from a concrete version of the Prime Number Theorem [34] that $\log \mathbf{e}[n_B] = O(\log n_B)$, which means that the exponentiation takes time $O(k^2 \log n_B) = O(k^3)$ per H query, since $\log n_B = O(k)$. This gives overall running time bound for A_r of $t[r] = t + O((q_H + q_s)^2(\ell + \log n_B) + q_H(q_H + q_s)k^3)$, as claimed.

To complete the proof, it remains to evaluate the success (in the CMA sense) probability of A_r. The CMA attack experiments for schemes $\mathsf{FDH}-\mathsf{MER}$ and $\mathsf{FDH}-\mathsf{RSA}_r$ are denoted by subscripts c and r, respectively.

We define the following events. Event SuccA means that A succeeds against $\mathsf{FDH-MER}$ in the CMA sense. Recall that SuccA means $(\sigma^*)^{\mathbf{e}[k^*]} \equiv H(k^*, m^*) \bmod N$ (call this event ValidA), and also that (k^*, m^*) was not queried to S (call this event NewMA). We let event SuccA_r mean that A_r succeeds against $\mathsf{FDH-RSA}_r$ in the CMA sense. Recall that SuccA_r means $(\sigma^*)^{\mathbf{e}[r]} \equiv H_r(k^*, m^*) \bmod N$ (call this event ValidA_r), and also that (k^*, m^*) was not queried to S_r (call this event NewMA_r).

Recalling that the simulated $H(.)$ coincides with $H_r(.)$ on inputs with prefix r, we now immediately see that if the event $\mathsf{SuccA} \wedge (k^* = r)$ occurs then event SuccA_r occurs. Consequently, since A_r perfectly simulates the view of A in exp. r exactly as in the real attack exp. c, it follows that the event $\mathsf{SuccA} \wedge (k^* = r)$ has the same probability in both experiments c and r for all $r \in [n_B]$. Therefore:

$$\mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A}_r,\mathsf{FDH-RSA}_r}(k) \quad \geq \quad \Pr[\mathsf{SuccA} \wedge (k^* = r)]_c \text{ for all } r \in [n_B]. \tag{5}$$

Summing (5) over all $r \in [n_B]$, we finally get the desired bound (3) using that the sum of $\Pr[\mathsf{SuccA} \wedge (k^* = r)]_c$ over $r \in [n_B]$ is equal to $\mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A},\mathsf{FDH-MER}}(\mathrm{k})$. This completes the proof. □

4 Requirements and Definition of Content Extraction Signatures

In this section we present a definition of a CES and then explain informally the requirements which a CES scheme should satisfy leading to precise definitions of desirable security notions for a CES.

4.1 Document Model and Related Terminology

We first introduce our document model and related terminology.

We will assume that a document is divided into a number n of smaller *submessages*, and that a document is represented in a form which encodes an ordering of the n submessages, their number, and allows some of them to be 'blank' (we use the special symbol ? to denote a blank submessage). We use $\mathbf{M}$ to denote such a representation of a document and $\mathsf{len}(\mathbf{M})$ to denote the number of submessages in the 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 $\mathbf{M}$.

The above 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 the 'blank' symbol ?, while the submessages with indices in X are set equal to the corresponding submessages with the same indices in the original document $\mathbf{M}$.

Definition 4.1 (Subdocument Extraction Xtr). *For any document* $\mathbf{M}$*, the* subdocument of $\mathbf{M}$ specified by extraction subset X*, denoted* $\mathsf{Xtr}(\mathbf{M}, X)$*, is the unique document* $\mathbf{M}'$ *satisfying the following properties:*

(1) $\mathsf{len}(\mathbf{M}') = \mathsf{len}(\mathbf{M}) = n$*, and*

(2) $\mathbf{M}'[i] = \mathbf{M}[i]$ *for all* $i \in X$ *and* $\mathbf{M}'[i] =?$ *for all* $i \in [n] - X$*.*

We denote the relation '$\mathbf{M}'$ is a subdocument of $\mathbf{M}$' between a pair of documents $\mathbf{M}'$ and $\mathbf{M}$ as $\mathbf{M}' \leq \mathbf{M}$.

Definition 4.2 (Subdocument Relation $\leq$). *For any pair of documents* $\mathbf{M}'$ *and* $\mathbf{M}$*, we say that* $\mathbf{M}'$ *is a subdocument of* $\mathbf{M}$*, denoted* $\mathbf{M}' \leq \mathbf{M}$ *if* $\mathbf{M}' = \mathsf{Xtr}(\mathbf{M}, X)$ *for some* X*, or equivalently, the following holds:*

(1) $\mathsf{len}(\mathbf{M}') = \mathsf{len}(\mathbf{M})$*, and*

(2) $\mathsf{Cl}(\mathbf{M}') \subseteq \mathsf{Cl}(\mathbf{M})$ *and*

(3) $\mathbf{M}'[i] = \mathbf{M}[i]$ *for all* $i \in \mathsf{Cl}(\mathbf{M}')$*.*

We write $\mathbf{M}' \not\leq \mathbf{M}$ if $\mathbf{M}'$ is not a subdocument of $\mathbf{M}$.

For example, given a document $\mathbf{M} = (m_1, m_2, m_3, m_4)$ and an extraction subset $X = \{1, 3\}$, the extracted subdocument of $\mathbf{M}$ specified by X is $\mathbf{M}' \stackrel{\text{def}}{=} \mathsf{Xtr}(\mathbf{M}, X) = (m_1, ?, m_3, ?)$. 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 $\mathbf{M}$. Also we have $\mathbf{M}[2] = m_2$, whereas $\mathbf{M}'[2] =?$ because the second submessage in $\mathbf{M}'$ is 'blank'.

4.2 Definition of a Content Extraction Signature

Using the above terminology, a *Content Extraction Digital Signature* (CES) scheme is defined as follows.

Definition 4.3 (CES). *A Content Extraction Signature scheme* $\mathsf{CES} = (\mathsf{GK}, \mathsf{Sig}, \mathsf{Ext}, \mathsf{Ver})$ *consists of 4 algorithms:*

1. GK — *Key Generation algorithm. Takes a security parameter* k *and generates a secret/public key pair* (sk, pk)*.*
2. Sig — *Signature Generation Algorithm. Takes a secret key* sk*, a document* $\mathbf{M}$*, and a content extraction access structure* $CEAS$*, and outputs a content extraction signature* σ_F*.*
3. Ext — *Signature Extraction Algorithm. Takes a public key* pk*, a document* $\mathbf{M}$*, a content extraction signature* σ_F*, an extraction subset* X*, and outputs an extracted signature* σ_E*.*
4. Ver — *Signature Verification Algorithm. Takes a public key* pk*, an extracted subdocument* $\mathbf{M}'$*, and an extracted signature* σ_E*, and outputs a verification decision* $d \in \{Acc, Rej\}$*.*

In the above definition, the main differences from a standard digital signature are the following. These differences are further discussed in the requirements sections below.

The Ext algorithm allows the user to extract (from a 'full' content extraction signature σ_F) a signature for the subdocument consisting of the submessages whose indexes are specified by the extraction subset X. The extracted signature σ_E 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 $n = \mathsf{len}(\mathbf{M})$ is the total number of submessages in the original signed document $\mathbf{M}$. We do not specify a specific encoding scheme for $CEAS$, leaving this to be specified by the application. Note that there are 2^n subsets of $[n]$ and hence 2^{2^n} collections of subsets

of $[n]$ (distinct CEASs). Hence the encoding length of 2^n bits for an arbitrary CEAS is exponential in n. But in practice we envisage a 'variable length' encoding scheme which allow 'common' CEASs to be compactly encoded. The definition of 'common' is application dependant and hence a suitable encoding scheme is also application dependant.

4.3 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. We do not fix a theoretical definition of the efficiency requirement since in practice it depends very much on implementation issues. However, for specific schemes and typical implementations estimated comparisons of efficiency will be made later on.

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.*

4.4 Security Requirements from a CES

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

4.4.1 Unforgeability

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 $\mathbf{M}$ 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* $\mathbf{M}$*, but not allowed to be extracted by the CEAS attached to the sign query* $\mathbf{M}$*.*

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.

Following the practice of proof-oriented modern cryptography we now give a quantitative definition of the CES-Unforgeability security notion, which we denote by $\mathrm{CES}-\mathrm{UF}$.

Definition 4.4 (Unforgeability Notion: $\mathrm{CES}-\mathrm{UF}$**).** *Let* $\mathsf{CES} = (\mathsf{GK}, \mathsf{Sig}, \mathsf{Ext}, \mathsf{Ver})$ *be a CES scheme. Let* A *be an attack algorithm. Define the experiment*

Experiment $\mathbf{CESUFExp}(\mathsf{CES}, \mathsf{A}, k)$
  $(sk, pk) \leftarrow \mathsf{GK}(k)$
  $(\mathbf{M}^*, \sigma_E^*) \leftarrow \mathsf{A}(k, pk | \mathsf{Sig}(sk, ., .))$
  $d \leftarrow \mathsf{Ver}(pk, \mathbf{M}^*, \sigma_E^*)$
  `If` $d = Acc$ and for all j,
  either $\mathbf{M}^* \not\leq \mathbf{M}_j$ or $\mathsf{CI}(\mathbf{M}^*) \notin CEAS_j$
    `Return` 1
  `Else`
    `Return` 0

Here, we denote by $(\mathbf{M}_j, CEAS_j)$ *the* j*th query of* A *to its* Sig *oracle. We quantify* A*'s success in breaking the* $\mathrm{CES}-\mathrm{UF}$ *security notion of scheme* CES *by the probability* $\mathbf{Succ}^{\mathrm{CES}-\mathrm{UF}}_{\mathsf{A},\mathsf{CES}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathbf{CESUFExp}(\mathsf{CES}, \mathsf{A}, k) = 1]$*. We define* A*'s resource parameters as* $RP = (t, q_s, n, \ell)$ *if* A *has*

running time/program size at most t, and makes at most q_s queries to the Sig *oracle, where each queried document consists of at most n submessages each of length at most ℓ bits. We quantify the insecurity of scheme* CES *in the sense of* $\mathrm{CES}-\mathrm{UF}$ *against arbitrary attackers with resource parameters $RP = (t, q_s, n, \ell)$ by the probability*

$$\mathbf{InSec}^{\mathrm{CES}-\mathrm{UF}}_{\mathsf{CES}}(t, q_s, n, \ell) \stackrel{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{CES}-\mathrm{UF}}_{\mathsf{A},\mathsf{CES}}.$$

The attacker set AS_{RP} contains all attackers with resource parameters $RP = (t, q_s, n, \ell)$.

4.4.2 Privacy

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. This consideration has important implications on the design and efficiency of our first two proposed CES schemes (see §5). 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' [8]. The requirement can be informally stated as follows.

CES-Privacy Requirement: *It is infeasible for an attacker to obtain, given an extracted signature σ for a subdocument $\mathbf{M}'$ of some document $\mathbf{M}$, any information on any submessage in the original document $\mathbf{M}$ which has not been extracted into the subdocument $\mathbf{M}'$* (i.e. has been 'blanked').

Our precise definition of the corresponding privacy notion $\mathrm{CES}-\mathrm{PR}$, included below, is an indistinguishability-based one, similar to the definition of semantic security for encryption schemes and follows the standard two-stage 'find/guess' attack experiment [24, 5]. In this formulation the attacker runs in two stages. In the find stage, the attacker's find algorithm $\mathsf{A}_{\mathsf{find}}$ chooses a pair of documents $\mathbf{M_0}$ and $\mathbf{M_1}$ which are identical except that they differ in the submessage in some position i^*, i.e. $\mathbf{M_0}[i] = \mathbf{M_1}[i]$ for $i \neq i^*$ but $\mathbf{M_0}[i^*] = M_0$ and $\mathbf{M_1}[i^*] = M_1$ for some pair of submessages M_0 and M_1 (note that below $\mathsf{A}_{\mathsf{find}}$ specifies $\mathbf{M_0}$ and $\mathbf{M_1}$ by outputting $(\mathbf{M}, M_0, M_1)$, where $\mathbf{M}[i] = \mathbf{M_0}[i] = \mathbf{M_1}[i]$ for all $i \neq i^*$). The experiment then chooses one of the subdocuments $\mathbf{M_0}$ and $\mathbf{M_1}$ at random by choosing a random bit $b \in \{0, 1\}$ and setting $\mathbf{M}^* = \mathbf{M}_b$. In the guess stage, the attacker's guess algorithm $\mathsf{A}_{\mathsf{guess}}$ is given the extracted signature σ^*_E for a subdocument $\overline{\mathbf{M}}^*$ of $\mathbf{M}^*$ into which the submessage M_b of $\mathbf{M}^*$ at position i^* has *not* been extracted, i.e. $\overline{\mathbf{M}}^*[i^*] = ?$. The goal of $\mathsf{A}_{\mathsf{guess}}$ is then to use any information leaked in σ^*_E to guess the bit of information b on the i^*th submessage. The requirement is that it is infeasible for guess to guess b correctly with non-negligible advantage over random guessing. We note that we also allow $\mathsf{A}_{\mathsf{find}}$ to choose i^* as well as the CEAS and extraction subset X^* (satisfying $i^* \notin X^*$) used to compute σ^*_E and even allow $\mathsf{A}_{\mathsf{find}}$ and $\mathsf{A}_{\mathsf{guess}}$ knowledge of the signer's secret key. The output s of $\mathsf{A}_{\mathsf{find}}$ denotes state information that is passed over as input to $\mathsf{A}_{\mathsf{guess}}$ (such as all the inputs of $\mathsf{A}_{\mathsf{find}}$).

Definition 4.5 (Privacy Notion: $\mathrm{CES}-\mathrm{PR}$**).** *Let* $\mathsf{CES} = (\mathsf{GK}, \mathsf{Sig}, \mathsf{Ext}, \mathsf{Ver})$ *denote a CES scheme. Define the experiment*

Experiment $\mathbf{CESPREXP}(\mathsf{CES}, \mathsf{A} = (\mathsf{A}_{\mathsf{find}}, \mathsf{A}_{\mathsf{guess}}), k)$

$(sk, pk) \leftarrow \mathsf{GK}(k)$
$(s, \mathbf{M}, M_0, M_1, i^*, CEAS^*, X^*) \leftarrow \mathsf{A}_{\mathsf{find}}(k, sk)$
$b \xleftarrow{\mathrm{R}} \{0,1\}$; $\mathbf{M}^* \leftarrow \mathbf{M}$; $\mathbf{M}^*[i^*] \leftarrow M_b$
$\sigma_F^* \leftarrow \mathsf{Sig}(sk, \mathbf{M}^*, CEAS^*)$
$\sigma_E^* \leftarrow \mathsf{Ext}(pk, \mathbf{M}^*, \sigma_F^*, X^*)$
$b' \leftarrow \mathsf{A}_{\mathsf{guess}}(s, \sigma_E^*)$
If $b' = b$ and $i^* \notin X^*$ then
  `Return` 1
Else
  `Return` 0

We quantify A*'s success in breaking the* CES − PR *security notion of scheme* CES *by the advantage* $\mathbf{Succ}^{\mathrm{CES-PR}}_{\mathsf{A},\mathsf{CES}}(k) \stackrel{\mathrm{def}}{=} 2(\Pr[\mathbf{CESPREXP}(\mathsf{CES}, \mathsf{A}, k) = 1] - \frac{1}{2})$*. We define* A*'s resource parameters as* $RP = (t, n, \ell)$ *if* A *has running time/program size at most* t*,* $\mathbf{M}$ *has at most* n *submessages each of length at most* ℓ *bits. We quantify the insecurity of scheme* CES *in the sense of* CES − PR *against arbitrary attackers with resource parameters* $RP = (t, n, \ell)$ *by the probability*

$$\mathbf{InSec}^{\mathrm{CES-PR}}_{\mathsf{CES}}(t, n, \ell) \stackrel{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{CES-PR}}_{\mathsf{A},\mathsf{CES}}.$$

The attacker set AS_{RP} *contains all attackers with resource parameters* $RP = (t, n, \ell)$*.*

We remark that (as is preferable) the CES − PR notion may hold even against a computationally *unbounded* attacker (as for example, in the simple multiple signature scheme). Since the time resource for such attackers are unbounded, we write $\mathbf{InSec}^{\mathrm{CES-PR}}_{\mathsf{CES}}(\infty)$ to denote the maximal advantage of such attackers.

5 Proposed Content Extraction Signature Schemes

5.1 Schemes based on General Cryptographic Primitives

5.1.1 Scheme CommitVector (CV)

We build this CES scheme out of two standard cryptographic primitives (refer to §3 for precise definitions of these primitives and their security notions):

(i) A standard digital signature scheme DS with signature and verification algorithms (S, V) and key generation algorithm K. We require that S satisfies the standard unforgeability notion, i.e. it is existentially unforgeable under adaptive chosen message attacks [25].

(ii) A message commitment scheme CM with committment algorithm $\mathsf{C}(.,.;.)$ and common parameters generation algorithm I. Given a message m, $\mathsf{C}(sp, m; r)$ denotes the commitment to message m under random string r and scheme parameters sp (we use the simpler notation $C(m; r)$ to denote $\mathsf{C}(sp, m; r)$ where sp is understood without ambiguity since only one commitment scheme is used). We require CM satisfy the standard *hiding* and *binding* security notions (we actually only need a *relaxed binding* requirement — see §3 for details).

Scheme CommitVector (CV)

Algorithm $\mathsf{GK}(k)$
$(sk_s, pk_s) \leftarrow \mathsf{K}(k)$
$sp \leftarrow \mathsf{I}(k)$
$pk \leftarrow (pk_s, sp)$; $sk \leftarrow sk_s$
Return (sk, pk)

Algorithm $\mathsf{Ext}(pk, \mathbf{M}, \sigma_F, X)$
Parse $pk = (pk_s, cp)$
Parse $\sigma_F \leftarrow (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in [n]})$
for $i \in [n] - X$, $\mathbf{c}[i] \leftarrow C(\mathbf{M}[i]; \mathbf{r}[i])$
$\sigma_E \leftarrow (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in X}, \langle \mathbf{c}[i] \rangle_{i \in [n]-X})$
Return σ_E

Algorithm $\mathsf{Sig}(sk, \mathbf{M}, CEAS)$
Parse $pk = (pk_s, cp)$; $sk = sk_s$
for $i \in [n]$, $\mathbf{c}[i] \leftarrow C(\mathbf{M}[i]; \mathbf{r}[i])$
$\sigma_c \leftarrow \mathsf{S}(sk_s, (CEAS, \langle \mathbf{c}[i] \rangle_{i \in [n]}))$
$\sigma_F \leftarrow (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in [n]})$
Return σ_F

Algorithm $\mathsf{Ver}(pk, \mathbf{M}', \sigma_E)$
Parse $pk = (pk_s, cp)$; Let $X' = \mathsf{CI}(\mathbf{M}')$
Parse $\sigma_E = (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in X'}, \langle \mathbf{c}[i] \rangle_{i \in [n]-X'})$
for $i \in \mathsf{CI}(\mathbf{M}')$, $\mathbf{c}[i] \leftarrow C(\mathbf{M}'[i]; \mathbf{r}[i])$
$d_s \leftarrow \mathsf{V}(pk_s, (CEAS, \langle \mathbf{c}[i] \rangle_{i \in [n]}), \sigma_c)$
if $d_s = Acc$ and $\mathsf{CI}(\mathbf{M}') \in CEAS$
$d \leftarrow Acc$
else $d \leftarrow Rej$
Return d

Figure 6: Definition of CES Scheme CV.

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 strings 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 strings 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.

The precise specification of the scheme CV is given in Fig. 6.

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 [26], and there exist fast candidates for such functions, such as the well known Secure Hash Algorithm (SHA) [31]. Using this commitment scheme with a number-theory based signature, one can approach in practice a computational saving by a factor close to n over the multiple signature scheme.

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 idea for the security of the scheme is that, on the one hand, signing the submessage commitments means that (due to the hiding property) no information is leaked on the unextracted submessages by the extracted signature because the corresponding randomness strings are removed. On the other hand, the binding property of the commitment ensures that submessages in signed documents cannot be modified without affecting the commitment, hence implying unforgeability, assuming the unforgeability of the underlying signature scheme. Moreover, because the forger cannot control the randomness strings used in signing, only a relaxed-binding for the commitment is sufficient. This intuition is formally proved and made precise below.

Theorem 5.1. *(1) If the standard signature scheme* DS *is existentially unforgeable under chosen message attack (*CMA *notion) and the commitment scheme* CM *has the relaxed-binding property (*$\mathrm{r-bind}$ *notion), then scheme* $\mathsf{CommitVector}$ *has the CES-Unforgeability property (*$\mathrm{CES-UF}$ *notion). Concretely, the following bound holds:*

$$\begin{aligned}\mathbf{InSec}^{\mathrm{CES-UF}}_{\mathsf{CES}}(t,q_s,n,\ell) \leq \quad & \mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{DS}}(t[DS],q_s[DS],\ell[DS]) \\ & +q_s\cdot n\cdot \mathbf{InSec}^{\mathrm{r-bind}}_{\mathsf{CM}}(t[CM],\ell[CM]),\end{aligned} \tag{6}$$

where $t[DS]=t$*;* $q_s[DS]=q_s$*;* $\ell[DS]=\ell$*, and* $t[CM]=t$*;* $\ell[CM]=\ell$*.*

(2) If the commitment scheme CM *is hiding (*hide *notion), then scheme* $\mathsf{CommitVector}$ *has the CES-Privacy property. Concretely, the following bound holds:*

$$\mathbf{InSec}^{\mathrm{CES-PR}}_{\mathsf{CES}}(t,n,\ell) \leq \quad \mathbf{InSec}^{\mathrm{hide}}_{\mathsf{CM}}(t[CM],\ell[CM]) \tag{7}$$

where $t[CM]=t$*;* $\ell[CM]=\ell$*.*

Proof. Proof of (1). We show how to use any efficient CES-Unforgeability attacker $\mathsf{A}_{\mathsf{CES}}$ for breaking scheme $\mathsf{CommitVector}$ in the sense of $\mathrm{CES-UF}$ to construct two efficient attackers:

(1) Attacker $\mathsf{A}_{\mathsf{sig}}$ breaks the unforgeability of signature scheme DS in the sense of CMA; and

(2) Attacker $\mathsf{A}_{\mathsf{com}}$ breaks the relaxed-binding of commitment scheme CM in the sense of $\mathrm{r-bind}$.

We will prove our claim by showing that if $\mathsf{A}_{\mathsf{CES}}$ succeeds with with non-negligible probability, then at least one of $\mathsf{A}_{\mathsf{sig}}$ or $\mathsf{A}_{\mathsf{com}}$ succeeds with non-negligible probability, thus contradicting either the assumed binding of CM or the assumed unforgeability of DS. More precisely, we show that:

$$\mathbf{Succ}^{\mathrm{CES-UF}}_{\mathsf{A}_{\mathsf{CES}},\mathsf{CV}}(k) \leq \mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A}_{\mathsf{sig}},\mathsf{DS}}(k) + q_s\cdot n\cdot \mathbf{Succ}^{\mathrm{r-bind}}_{\mathsf{A}_{\mathsf{com}},\mathsf{CM}}(k), \tag{8}$$

where $\mathsf{A}_{\mathsf{CES}}$ has Resource Parameters (RPs) (t,q_s,n,ℓ), $\mathsf{A}_{\mathsf{sig}}$ has RPs $(t[DS],q_s[DS],\ell[DS])$ with

$$t[DS]=t; q_s[DS]=q_s; \ell[DS]=\ell. \tag{9}$$

and $\mathsf{A}_{\mathsf{com}}$ has RPs $(t[CM],\ell[CM])$ with

$$t[CM]=t; \ell[CM]=\ell. \tag{10}$$

The theorem then follows immediately from (8), (9) and (10) by taking maximums over all attackers $\mathsf{A_{sig}}$ and $\mathsf{A_{com}}$ with the given RPs. It remains to describe the attackers $\mathsf{A_{sig}}$ and $\mathsf{A_{com}}$ and show that they satisfy (8), (9) and (10).

Attacker $\mathsf{A_{sig}}$ is invoked by the CMA experiment **CMAExp** on input (k, pk_s), where k is the security parameter and pk_s is a public key for DS output by the key generation algorithm $\mathsf{K}(k)$, with sk_s denoting the corresponding secret key. $\mathsf{A_{sig}}$ is able to query messages to be signed by the signing oracle $\mathsf{S}(sk_s, .)$ for scheme DS. $\mathsf{A_{sig}}$ then runs as follows:

Setup. $\mathsf{A_{sig}}$ runs the init. algorithm I for scheme CM to get commitment parameters $sp = \mathsf{I}(k)$. As done by the CES CV scheme key gen. algorithm GK, it defines the a key pair (sk, pk) for scheme CV by setting $pk = (pk_s, sp)$ and $sk = sk_s$. $\mathsf{A_{sig}}$ then runs the CES $-$ UF attacker $\mathsf{A_{CES}}$ for scheme CV on input (k, pk).

Sig **Queries.** When $\mathsf{A_{CES}}$ makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its Sig oracle, $\mathsf{A_{sig}}$ answers it by simulating the action of the Sig algorithm. Since $\mathsf{A_{sig}}$ does not know the signing key sk_s, it makes use of its $\mathsf{S}(sk_s, .)$ signing oracle, which suffices to simulate Sig. The details follow. Let $n_j = \mathsf{len}(\mathbf{M}_j)$ denote the number of submessages in $\mathbf{M}_j$. First, $\mathsf{A_{sig}}$ computes the commitment vector $\mathbf{c}_j$, where $\mathbf{c}_j[i] = C(\mathbf{M}_j[i]; \mathbf{r}_j[i])$ using the n_j randomness strings $\mathbf{r}_j[i]$ for $i \in [n_j]$. Then $\mathsf{A_{sig}}$ queries $(CEAS_j, \mathbf{c}_j)$ to its sign oracle $\mathsf{S}(sk_s, .)$ to get the DS signature $\bar{\sigma}_j$. Finally, it sets the simulated CES signature to $\sigma_j = (CEAS_j, \bar{\sigma}_j, \mathbf{r}_j)$.

Output. Eventually $\mathsf{A_{CES}}$ terminates and outputs a CES document/extracted-signature forgery pair $(\mathbf{M}^*, \sigma^*)$. Let $n^* = \mathsf{len}(\mathbf{M}^*)$ and $X^* = \mathsf{CI}(\mathbf{M}^*)$. $\mathsf{A_{sig}}$ parses the extracted signature $\sigma^* = (CEAS^*, \bar{\sigma}^*, \langle \mathbf{r}^*[i] \rangle_{i \in X^*}, \langle \mathbf{c}^*[i] \rangle_{i \in [n^*] - X^*})$. It computes the commitments for the clear submesssages using $\mathbf{c}^*[i] = C(\mathbf{M}^*[i]; \mathbf{r}^*[i])$ for $i \in X^*$ to obtain (together with $\langle \mathbf{c}^*[i] \rangle_{i \in [n^*] - X^*}$ from σ^*) the full forgery commitment vector $\mathbf{c}^* = \langle \mathbf{c}^*[i] \rangle_{i \in [n^*]}$. Note that if the CES forgery $(\mathbf{M}^*, \sigma^*)$ is valid (i.e. $\mathsf{Ver}(pk, \mathbf{M}^*, \sigma^*) = Acc$) then $(M^*, \bar{\sigma}^*)$, with $M^* = (CEAS^*, \mathbf{c}^*)$, is a valid message/signature pair for DS (i.e. $\mathsf{V}(pk_s, M^*, \bar{\sigma}^*) = Acc$). So $\mathsf{A_{sig}}$ terminates and outputs the DS forgery $(M^*, \bar{\sigma}^*)$.

This completes the description of $\mathsf{A_{sig}}$. To establish (8), we evaluate the success probability of $\mathsf{A_{sig}}$.

We use the notation $\Pr[.]_{exp}$ to denote probability in experiment exp. The attack experiments for CES scheme CV(**CESUFExp**), sig. scheme DS(**CMAExp**), and commit. scheme CM (**RBindExp**) are denoted using the shortened notations c, d and cm, respectively.

We define the following events. Event BrkDS means that $\mathsf{A_{sig}}$ succeeds in the CMA sense. Event BrkCES means that $\mathsf{A_{CES}}$ succeeds in the CES $-$ UF sense. Event NewM means that $M^* = (CEAS^*, \mathbf{c}^*) \neq (CEAS_j, \mathbf{c}_j)$ for all $j \in [q_s]$.

We know as observed above that if BrkCES occurs so $(\mathbf{M}^*, \sigma^*)$ is a valid forgery for CV, then the output forgery (M^*, σ^*) of $\mathsf{A_{sig}}$ is valid for DS. But for $\mathsf{A_{sig}}$ to succeed in CMA sense, it is also necessary (and sufficient) that M^* has not been queried to $\mathsf{S}(sk_s, .)$ by $\mathsf{A_{sig}}$, meaning that event NewM occurs. So $\mathsf{A_{sig}}$'s success probability can be lower bounded as follows:

$$\begin{aligned}
\mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A_{sig}},\mathsf{DS}}(k) &\geq \Pr[\mathsf{BrkCES} \wedge \mathsf{NewM}]_d && (11)\\
&= \Pr[\mathsf{BrkCES}]_d - \Pr[\mathsf{BrkCES} \wedge \neg\mathsf{NewM}]_d && (12)\\
&= \mathbf{Succ}^{\mathrm{CES-UF}}_{\mathsf{A_{CES}},\mathsf{CV}}(k) - \Pr[\mathsf{BrkCES} \wedge \neg\mathsf{NewM}]_c, && (13)
\end{aligned}$$

where to get (13) we used the fact that in experiment d, $\mathsf{A_{sig}}$ simulates the view of $\mathsf{A_{CES}}$ exactly as in the original attack experiment c, and hence the events BrkCES and $\mathsf{BrkCES} \wedge \neg\mathsf{NewM}$ have the same probabilities in experiments c and d.

To complete the proof of (8), we show below how to construct the attacker $\mathsf{A_{com}}$ such that

$$\Pr[\mathsf{BrkCES} \wedge \neg\mathsf{NewM}]_c \leq q_s \cdot n \cdot \mathbf{Succ}^{\mathrm{snotr-bind}}_{A_{com},\mathsf{CM}}(k). \tag{14}$$

Substituting (14) in (13) gives (8).

Let Coll denote the event $\mathsf{BrkCES} \wedge \neg\mathsf{NewM}$. Before we construct $\mathsf{A_{com}}$, we need to establish the following claim: If Coll occurs then there exists a Sig query index $j^* \in [q_s]$ and a submessage index i^* such that $(\mathbf{M}_{j^*}[i^*]; \mathbf{r}_{j^*}[i^*])$ and $(\mathbf{M}^*[i^*]; \mathbf{r}^*[i^*])$ form a collision pair for the commit. algorithm $C(.;.)$, meaning:

$$\mathbf{M}_{j^*}[i^*] \neq \mathbf{M}^*[i^*] \text{ but } \mathbf{c}_{j^*}[i^*] = C(\mathbf{M}_{j^*}[i^*]; \mathbf{r}_{j^*}[i^*]) = C(\mathbf{M}^*[i^*]; \mathbf{r}^*[i^*]) = \mathbf{c}^*[i^*]. \tag{15}$$

We now show that Coll implies (15). First, since Coll implies $\neg\mathsf{NewM}$, there exists j^* such that $(CEAS^*, \mathbf{c}^*) = (CEAS_{j^*}, \mathbf{c}_{j^*})$. Also, since Coll implies BrkCES, we know that $(\mathbf{M}^*, \sigma^*)$ is a valid CES signature, meaning $\mathsf{CI}(\mathbf{M}^*) \in CEAS^* = CEAS_{j^*}$. So by def. of BrkCES we must have $\mathbf{M}^* \not\leq \mathbf{M}_{j^*}$. Now $n^* \neq n_{j^*}$ cannot occur since it implies $\mathbf{c}^*$ and $\mathbf{c}_{j^*}$ have different lengths, contradicting earlier result that these vectors are equal. Also, $\mathsf{CI}(\mathbf{M}^*) \not\subseteq \mathsf{CI}(\mathbf{M}_{j^*})$ cannot occur because we may assume that $\mathsf{CI}(\mathbf{M}_{j^*}) = [n_{j^*}]$. Consequently $\mathbf{M}^* \not\leq \mathbf{M}_{j^*}$ implies that there exists $i^* \in \mathsf{CI}(\mathbf{M}^*)$ satisfying (15), as required.

We now describe attacker $\mathsf{A_{com}} = (\mathsf{A_1}, \mathsf{A_2})$. $\mathsf{A_1}$ is invoked by the r − bind experiment cm on input (k, sp), where k is the security parameter and sp is the commitment scheme parameters from $\mathsf{I}(k)$. The idea is that $\mathsf{A_{com}}$ makes a guess (β, α) for (j^*, i^*) defined above and chooses the corresponding message $\mathbf{M}_\beta[\alpha]$ as one of its binding collision messages, using $\mathbf{M}^*[\alpha]$ as the other, so that if the guess was right and Coll occurs then $\mathsf{A_{com}}$ succeeds to find a commitment collision. The details follow.

Setup. $\mathsf{A_1}$ runs $\mathsf{K}(k)$ to get a DS key pair (sk_s, pk_s). It defines (pk, sk) for CV as in experiment c above. It also chooses uniformly at random a pair of indices $(\beta, \alpha) \in [q_s] \times [n]$ as a guess for the query/submessage index pair (j^*, i^*) corresponding to one of the commitment collision pairs, as defined in (15) above. $\mathsf{A_1}$ then runs the CES − UF attacker $\mathsf{A_{CES}}$ for scheme CV on input (k, pk).

Sig **Queries.** When $\mathsf{A_{CES}}$ makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its Sig oracle, $\mathsf{A_1}$ answers it by simulating Sig as done by $\mathsf{A_{sig}}$. The only differences are as follows. Firstly, $\mathsf{A_1}$ knows sk_s so uses it directly. Secondly, when $j = \beta$, $\mathsf{A_1}$ saves its state in s, and outputs $(s, \mathbf{M}_\beta[\alpha])$. The experiment chooses random $\bar{r}$, computes $\bar{c} = C(\mathbf{M}_\beta[\alpha]; \bar{r})$, and runs $\mathsf{A_2}$ on input $(s, \bar{c}, \bar{r})$. $\mathsf{A_2}$ sets $\mathbf{r}_\beta[\alpha] = \bar{r}$, $\mathbf{c}_\beta[\alpha] = \bar{c}$, and completes the answer to $\mathsf{A_{CES}}$'s βth Sig query and all subsequent queries as done by $\mathsf{A_{sig}}$.

Output. Eventually $\mathsf{A_{CES}}$ terminates and outputs a CES document/extracted-signature forgery pair $(\mathbf{M}^*, \sigma^*)$. $\mathsf{A_2}$ then outputs $(\mathbf{M}^*[\alpha], \mathbf{r}^*[\alpha])$ as the collision pair corresponding to $(\mathbf{M}_\beta[\alpha], \mathbf{r}_\beta[\alpha])$.

Note that $\mathsf{A_{com}}$ perfectly simulates the view of $\mathsf{A_{CES}}$ as in Experiments d and c, regardless of the outcome of (β, α). Hence, for all (j, i),

$$\Pr[\mathsf{Coll} \wedge (j^*, i^*) = (j, i) | (\beta, \alpha) = (j, i)]_{cm} = \Pr[\mathsf{Coll} \wedge (j^*, i^*) = (j, i)]_c.$$

But if Coll occurs so (15) holds and $\mathsf{A_{com}}$'s guess (β, α) for (j^*, i^*) was correct then $\mathsf{A_{com}}$ succeeds because its output is a commitment collision: $\mathbf{M}^*[\alpha] \neq \mathbf{M}_\beta$ but $C(\mathbf{M}^*[\alpha]; \mathbf{r}^*[\alpha]) = C(\mathbf{M}_\beta[\alpha], \mathbf{r}_\beta[\alpha])$.

Hence using that (β, α) is uniformly chosen in $[q_s] \times [n]$ we get

$$\begin{aligned}
\mathbf{Succ}^{\mathrm{r-bind}}_{A_{com},CM}(k) &\geq \Pr[\mathsf{Coll} \wedge (j^*, i^*) = (\beta, \alpha)]_{cm} \\
&= \sum_{(j,i)\in[q_s]\times[n]} \Pr[\mathsf{Coll} \wedge (j^*, i^*) = (j, i) | (\beta, \alpha) = (j, i)]_{cm} \\
&\qquad \cdot \Pr[(\beta, \alpha) = (j, i)]_{cm} \\
&= (\frac{1}{q_s \cdot n}) \cdot \sum_{(j,i)\in[q_s]\times[n]} \Pr[\mathsf{Coll} \wedge (j^*, i^*) = (j, i)]_c \\
&= (\frac{1}{q_s \cdot n}) \Pr[\mathsf{Coll}]_c. \qquad (16)
\end{aligned}$$

The inequality (16) immediately implies the desired result (14). The time and query bounds (9) and (10) for $\mathsf{A_{sig}}$ and $\mathsf{A_{com}}$ are clear from the specification of $\mathsf{A_{sig}}$ and $\mathsf{A_{com}}$, respectively. This completes the proof of part (1) of the Theorem.

Proof of (2). We show how to use any efficient CES-Privacy attacker $\mathsf{A_{CES}} = (\mathsf{A_1}, \mathsf{A_2})$ to construct an efficient attacker $\bar{\mathsf{A}}_{\mathsf{com}} = (\bar{\mathsf{A}}_1, \bar{\mathsf{A}}_2)$ to break the hiding property of commitment scheme CM, thus contradicting the assumed hiding property of the latter. More precisely, we show that:

$$\mathbf{Succ}^{\mathrm{CES-PR}}_{\mathsf{A_{CES}},\mathsf{CV}}(k) \leq \mathbf{Succ}^{\mathrm{hide}}_{\mathsf{A_{com}},\mathsf{CM}}(k), \qquad (17)$$

where $\mathsf{A_{CES}}$ has Resource Parameters (RPs) (t, n, ℓ), and $\mathsf{A_{com}}$ has RPs $(t[CM], \ell[CM])$ with

$$t[CM] = t; \ell[CM] = \ell. \qquad (18)$$

The theorem then follows immediately from (17), and (18) by taking maximums over all attackers $\mathsf{A_{com}}$ with the given RPs. It remains to describe the attacker $\mathsf{A_{com}}$ and show that it satisfies (17) and (10).

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 C.

We will denote the CES − PR notion experiment **CESPRExp** for CV by c and the hide notion exp. **HideExp** for CM by cm.

We now describe attacker $\bar{\mathsf{A}}_{\mathsf{com}} = (\mathsf{A_1}, \mathsf{A_2})$. It is invoked by exp. cm on input (k, sp), where k is the security parameter and sp is the commitment scheme parameters from $\mathsf{I}(k)$. The idea is straightforward: $\mathsf{A_{com}}$ will run $\mathsf{A_{CES}}$, incorporate its challenge commitment into the challenge extracted signature given to $\mathsf{A_{CES}}$, so that $\mathsf{A_{com}}$ can just use $\mathsf{A_{CES}}$'s prediction bit to solve its challenge. The details follow.

Setup. $\bar{\mathsf{A}}_1$ runs $\mathsf{K}(k)$ to get a DS key pair (sk_s, pk_s). It defines (pk, sk) for CV as in the GK algorithm for CV. It then runs $\mathsf{A_1}$ on input (k, sk).

End of $\bar{A}$'s Find Stage. Eventually $\mathsf{A_1}$ terminates and outputs $s' = (s, \mathbf{M}^*, M_0, M_1, i^*, CEAS^*, X^*)$. $\bar{\mathsf{A}}_1$ then saves its state into $\bar{s} = (k, sp, (sk, pk), s')$, and uses (M_0, M_1) as the message pair to be challenged on. It terminates and outputs $(\bar{s}, M_0, M_1)$.

$\bar{\mathsf{A}}$**'s Challenge.** The *cm* experiment chooses a random $b \in \{0,1\}$, computes challenge commitment $c^* = C(M_b; r^*)$ for a random string r^*, and runs $\bar{\mathsf{A}}_2$ on input $(\bar{s}, c^*)$.

$\bar{\mathsf{A}}$**'s Guess Stage.** $\bar{\mathsf{A}}_2$ retrieves $(k, sp, (sk, pk), s')$ from $\bar{s}$. It then simulates an extracted signature σ_E^* for input document $\mathbf{M}^*$ and extraction subset X^* (using c^* for the i^*th commitment). The details follow. Let $n^* = \mathsf{len}(\mathbf{M}^*)$. For $i \in [n^*] - \{i^*\}$, $\bar{\mathsf{A}}_2$ computes $\mathbf{c}^*[i] = C(\mathbf{M}^*[i]; \mathbf{r}^*[i])$ using random strings $\mathbf{r}^*[i]$. For $i = i^*$, it sets $\mathbf{c}^*[i^*] = c^*$. Then it produces $\bar{\sigma^*} = \mathsf{S}(sk_s, (CEAS^*, \mathbf{c}^*))$. Finally it sets the simulated extracted CES signature to $\sigma_E^* = (CEAS^*, \bar{\sigma^*}, \langle \mathbf{r}^*[i] \rangle_{i \in X^*}, \langle \mathbf{c}^*[i] \rangle_{i \in [n^*] - X^*}$ (note that $i^* \notin X^*$), and runs A_2 on input (s, σ_E^*).

Output. Eventually A_2 terminates and outputs a guess bit b' for b. Then $\bar{\mathsf{A}}_2$ just output b' as its guess for b.

This completes the description of $\mathsf{A}_{\mathsf{com}}$. To establish (17), we just observe that since $\mathsf{A}_{\mathsf{com}}$ in experiment *cm* perfectly simulates (when $i^* \notin X^*$) the view of $\mathsf{A}_{\mathsf{CES}}$ as in experiment *c*, where the bit b which $\mathsf{A}_{\mathsf{com}}$ needs to guess is equal to the bit which $\mathsf{A}_{\mathsf{CES}}$ needs to guess. Therefore $\Pr[b' = b]_{cm} \geq \Pr[b' = b \wedge (i^* \notin X^*)]_{cm} = \Pr[b' = b \wedge (i^* \notin X^*)]_c$, which immediately implies the desired result (17). The stated RPs (18) follow from the specification of $\mathsf{A}_{\mathsf{com}}$. This completes the proof of part (2). □

5.1.2 A Variant: Scheme HashTree (HT)

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 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 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 family CR (an instance function from the family is denoted $H(.)$) 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 constructed by associating 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 signer signs the root hash value with an appended CEAS (and length) encoding, as in scheme CV. 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 [29]). This scheme can be regarded as a generalization and improvement (due to our additional privacy property) of the batch signature of Pavlovski and Boyd [32] 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. We remark that the privacy afforded by the use of commitments in our scheme may also be useful in the context of batch signatures as discussed in [32], 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.

For the precise specification of scheme HT we need the following definitions.

To an input document $\mathbf{M}$ to Sig containing $n = \mathsf{len}(M)$ submessages, we associate a directed rooted complete and balanced binary tree $T_n = (V_n, E_n)$ having n leaf vertices. Let $h_n = \lfloor \log_2(n) \rfloor$ and $m_n = n - 2^h \in [0, 2^h - 1]$. The tree T_n has $h + 1$ levels. There are exactly $2m$ leaves at level h, which are precisely the children of the m leftmost (internal) vertices at level $h - 1$. The remaining $2^h - m$ rightmost vertices at level $h - 1$ are the remaining leaves of T_n. To each vertex of T_n we associate an *address* (label) $a \in \{0,1\}^*$, and a *value* $v_a \in \{0,1\}^*$.

The vertex addresses are assigned using the 'universal address system'. In this system the root has address 0 (so the root value is denoted v_0), and the address of all other vertices is obtained recursively by the following rule: the left and right children of a node of address a (value v_a) are assigned addresses $(a, 0)$ and $(a, 1)$, respectively (values $v_{a,0}$ and $v_{a,1}$). Consequently, each vertex at level i has an address of the form $(0, a_i, \ldots, a_0)$ for $a_k \in \{0,1\}$.

The vertex values are assigned by first initialising the values of all the leaf vertices v_a using the commitment vector $\mathbf{c}$. Then each internal vertex value is determined recursively by the following rule: the value v_a of an internal vertex of address a is assigned the hash of the concatanation of the values of the left and right children of the vertex, i.e $v_a = H((v_{a,0}, v_{a,1}))$. If $a = (0, a_m, \ldots, a_i)$ is an address of any node except the root, we therefore have that $v_{0,a_m,\ldots,a_{i+1}} = H(Y_{a_i}(v_{0,a_m,\ldots,a_i}, v_{0,a_m,\ldots,\bar{a}_i}))$, where, given a pair of strings (s_1, s_2), the function $Y_1(.)$ swaps the strings to return $Y_1(s_1, s_2) = (s_2, s_1)$, whereas $Y_0(.)$ denotes the identity function.

We define the following functions $A_n(.)$, $P_n(.)$, $S_n(.)$ and and a set I_n associated with the tree T_n.

The function $A_n(.)$ associates to each submessage index $i \in [n]$ a unique leaf vertex address $A_n(i)$, so that the value $v_{A_n(i)}$ of the vertex at that address is set to the commitment $\mathbf{c}[i]$ for the ith submessage $\mathbf{M}[i]$. The first $2m$ indices map to the leftmost $2m$ leaves at level h and the other indices map to the rightmost $2^h - m$ leaves at level $h - 1$. Hence $A_n(.)$ is defined as follows. For $i \in [2m]$, we let $A_n(i) = (0, i_h, \ldots, i_0)$, where $(i_h, \ldots, i_0)$ denotes the $(h+1)$-bit binary representation of $i - 1$. For $i \in [2m + 1, 2^h + m]$, we let $A_n(i) = (0, \bar{i}_{h-1}, \ldots, \bar{i}_0)$, where $(\bar{i}_{h-1}, \ldots, \bar{i}_0)$ denotes the h-bit binary rep. of $i - m - 1$.

The function $P_n(.)$ associates to each leaf vertex address the list of addresses of the vertices on the unique path from that leaf to the root (starting at the root). That is, for each leaf address $a = (0, a_m, \ldots, a_0) \in \{0,1\}^{m+1}$ (for $m \in \{h, h+1\}$), we have the leaf-to-root address path $P_n(a) = (0, (0, a_m), (0, a_m, a_{m-1}), \ldots, (0, a_m, \ldots, a_0))$. Note that, except for the root v_0, each vertex $v_{0,a_m,\ldots,a_i}$ in the path $P_n(a)$ has a sibling vertex $v_{0,a_m,\ldots,\bar{a}_i}$. The value of this vertex is also needed to compute the value of the parent vertex.

The function $S_n(.)$ associates to each leaf vertex address a the list of addresses of the siblings of the vertices in $P_n(a)$ (except the root). That is, for $a = (0, a_m, \ldots, a_0)$, we have $S_n(a) = ((0, \bar{a}_m), (0, a_m, \bar{a}_{m-1}), \ldots, (0, a_m, \ldots, \bar{a}_0))$.

The set I_n consists of the addresses of all internal vertices of the tree T_n (i.e. only the leaves are not included).

The precise definition of the scheme HT is given in Fig. 7.

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.

Scheme HashTree (HT)

Algorithm $\mathsf{GK}(k)$
$(sk_s, pk_s) \leftarrow \mathsf{K}(k)$
$sp \leftarrow \mathsf{I}(k)$; $\alpha \leftarrow \mathsf{KH}(k)$
$pk \leftarrow (pk_s, sp, \alpha)$; $sk \leftarrow sk_s$
Return (sk, pk)

Algorithm $\mathsf{Sig}(sk, \mathbf{M}, CEAS)$
$n \leftarrow \mathsf{len}(\mathbf{M})$
Parse $pk = (pk_s, cp, \alpha)$; $sk = sk_s$
for $i \in [n]$, (Leaf Vert.)
$\mathbf{c}[i] \leftarrow C(\mathbf{M}[i]; \mathbf{r}[i])$
$\mathbf{a}[i] \leftarrow A_n(i)$; $v_{\mathbf{a}[i]} \leftarrow \mathbf{c}[i]$
for each $a \in I_n$ (Int. Vert.)
$v_a \leftarrow H((v_{a,0}, v_{a,1}))$
$\sigma_c \leftarrow \mathsf{S}(sk_s, (CEAS, n, v_0))$
$\sigma_F \leftarrow (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in [n]})$
Return σ_F

Algorithm $\mathsf{Ext}(pk, \mathbf{M}, \sigma_F, X)$
$n \leftarrow \mathsf{len}(\mathbf{M})$
Parse $pk = (pk_s, cp, \alpha)$; $sk = sk_s$
Parse $\sigma_F = (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in [n]})$
for $i \in [n]$, (Leaf Vert.)
$\mathbf{c}[i] \leftarrow C(\mathbf{M}[i]; \mathbf{r}[i])$
$\mathbf{a}[i] \leftarrow A_n(i)$; $v_{\mathbf{a}[i]} \leftarrow \mathbf{c}[i]$
for each $a \in I_n$ (Int. Vert.)
$v_a \leftarrow H((v_{a,0}, v_{a,1}))$
$P_X \leftarrow \langle P_n(\mathbf{a}[i]) \rangle_{i \in X}$ (Path Vert.)
$S_X \leftarrow \langle S_n(\mathbf{a}[i]) \rangle_{i \in X}$ (Sib. Vert.)
$\sigma_E \leftarrow (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in [X]}, \langle v_a \rangle_{a \in S_X - P_X})$
Return σ_E

Algorithm $\mathsf{Ver}(pk, \mathbf{M}', \sigma_E)$
$n \leftarrow \mathsf{len}(\mathbf{M}')$; $X' \leftarrow \mathsf{Cl}(\mathbf{M}')$
$P'_X \leftarrow \langle P_n(\mathbf{a}[i]) \rangle_{i \in X'}$ (Path Vert.)
$S'_X \leftarrow \langle S_n(\mathbf{a}[i]) \rangle_{i \in X'}$ (Sib. Vert.)
Parse $pk = (pk_s, cp, \alpha)$; $sk = sk_s$
Parse $\sigma_E = (CEAS, \sigma_c, \langle \mathbf{r}[i] \rangle_{i \in [X]}, \langle v_a \rangle_{a \in S'_X - P'_X})$
for $i \in X'$, (Leaf Vert.)
$\mathbf{c}[i] \leftarrow C(\mathbf{M}'[i]; \mathbf{r}[i])$
$\mathbf{a}[i] \leftarrow A_n(i)$; $v_{\mathbf{a}[i]} \leftarrow \mathbf{c}[i]$
(Compute root v_0:)
for each $a \in P_{X'}$ (Int. Vert.)
$v_a \leftarrow H((v_{a,0}, v_{a,1}))$
$d_s \leftarrow \mathsf{V}(pk_s, (CEAS, n, v_0), \sigma_c)$
if $d_s = Acc$ and $\mathsf{Cl}(\mathbf{M}') \in CEAS$
$d \leftarrow Acc$
else $d \leftarrow Rej$
Return d

Figure 7: Definition of CES scheme HT.

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, 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 5.1, together with the collision-resistance property of the Hash tree.

Theorem 5.2. *(1) If the standard signature scheme* DS *is existentially unforgeable under chosen message attack (*CMA *notion), the commitment scheme* CM *has the relaxed-binding property (*$\mathrm{r-bind}$ *notion), and the Hash function family* CR *is collision-resistant (*cr *notion) then scheme* HT *has the CES-Unforgeability property (*$\mathrm{CES-UF}$ *notion). Concretely, the following bound holds:*

$$\begin{aligned}\mathbf{InSec}^{\mathrm{CES-UF}}_{\mathsf{HT}}(t, q_s, n, \ell) \leq \quad & \mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{DS}}(t[DS], q_s[DS], \ell[DS]) && (19)\\ & +q_s \cdot n \cdot \mathbf{InSec}^{\mathrm{r-bind}}_{\mathsf{CM}}(t[CM], \ell[CM]), \\ & +\mathbf{InSec}^{\mathrm{cr}}_{\mathsf{CR}}(t[CR]) && (20)\end{aligned}$$

where $t[DS] = t$; $q_s[DS] = q_s$; $\ell[DS] = \ell$, $t[CM] = t$; $\ell[CM] = \ell$ *and* $t[CR] = t$.

(2) If the commitment scheme CM *is hiding (*hide *notion), then scheme* HT *has the CES-Privacy property (*$\mathrm{CES-PR}$ *notion). Concretely, the following bound holds:*

$$\mathbf{InSec}^{\mathrm{CES-PR}}_{\mathsf{HT}}(t, n, \ell) \leq \quad \mathbf{InSec}^{\mathrm{hide}}_{\mathsf{CM}}(t[CM], \ell[CM]) \qquad (21)$$

where $t[CM] = t$; $\ell[CM] = \ell$.

Proof. Proof of (1). We show how to use any efficient CES-Unforgeability attacker $\mathsf{A}_{\mathsf{CES}}$ for breaking scheme HT in the sense of $\mathrm{CES-UF}$ to construct three efficient attackers:

(1) Attacker $\mathsf{A}_{\mathsf{sig}}$ breaks the unforgeability of signature scheme DS in the sense of CMA;

(2) Attacker $\mathsf{A}_{\mathsf{com}}$ breaks the relaxed-binding of commitment scheme CM in the sense of $\mathrm{r-bind}$;

(3) Attacker A_{cr} breaks the collision-resistance of the hash-function family CR in the sense of cr.

We will prove our claim by showing that if $\mathsf{A}_{\mathsf{CES}}$ succeeds with with non-negligible probability, then at least one of $\mathsf{A}_{\mathsf{sig}}$, $\mathsf{A}_{\mathsf{com}}$ or A_{cr} succeeds with non-negligible probability, thus contradicting either the assumed binding of CM or the assumed unforgeability of DS, or the assumed collision-resistance of CR. More precisely, we show that:

$$\mathbf{Succ}^{\mathrm{CES-UF}}_{\mathsf{A}_{\mathsf{CES}},\mathsf{HT}}(k) \leq \mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A}_{\mathsf{sig}},\mathsf{DS}}(k) + q_s \cdot n \cdot \mathbf{Succ}^{\mathrm{r-bind}}_{\mathsf{A}_{\mathsf{com}},\mathsf{CM}}(k) + \mathbf{Succ}^{\mathrm{cr}}_{\mathsf{A}_{\mathsf{cr}},\mathsf{CR}}(k), \qquad (22)$$

where $\mathsf{A_{CES}}$ has Resource Parameters (RPs) (t, q_s, n, ℓ), $\mathsf{A_{sig}}$ has RPs $(t[DS], q_s[DS], \ell[DS])$ with $t[DS] = t$; $q_s[DS] = q_s$; $\ell[DS] = \ell$, $\mathsf{A_{com}}$ has RPs $(t[CM], \ell[CM])$ with $t[CM] = t$; $\ell[CM] = \ell$, and $\mathsf{A_{cr}}$ has RPs $(t[CR])$ with $t[CR] = t$. The theorem follows immediately from (22) by taking maximums over all attackers $\mathsf{A_{sig}}$, $\mathsf{A_{com}}$ and $\mathsf{A_{cr}}$ with the given RPs. It remains to describe the attackers $\mathsf{A_{sig}}$, $\mathsf{A_{com}}$ and $\mathsf{A_{cr}}$ and show that they satisfy (22).

Attacker $\mathsf{A_{sig}}$ is invoked by the CMA experiment **CMAExp** on input (k, pk_s), where k is the security parameter and pk_s is a public key for DS, with sk_s denoting the corresponding secret key. $\mathsf{A_{sig}}$ is able to query messages to be signed by the signing oracle $\mathsf{S}(sk_s, .)$ for scheme DS. $\mathsf{A_{sig}}$ then runs as follows:

Setup. As done by GK of HT, $\mathsf{A_{sig}}$ runs $\mathsf{I}(k)$ for scheme CM and $\mathsf{KH}(k)$ for scheme CR to get scheme parameters sp and α, respectively. $\mathsf{A_{sig}}$ then defines a key pair (sk, pk) for scheme HT by $pk = (pk_s, sp, \alpha)$ and $sk = sk_s$. $\mathsf{A_{sig}}$ then runs the CES − UF attacker $\mathsf{A_{CES}}$ for scheme HT on input (k, pk).

Sig **Queries.** When $\mathsf{A_{CES}}$ makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its Sig oracle, $\mathsf{A_{sig}}$ answers it by simulating the action of the Sig algorithm using its $\mathsf{S}(sk_s, .)$ signing oracle. The details follow. Let $n_j = \mathsf{len}(\mathbf{M}_j)$. First, $\mathsf{A_{sig}}$ computes the commitment vector $\mathbf{c}_j$, where $\mathbf{c}_j[i] = C(\mathbf{M}_j[i]; \mathbf{r}_j[i])$. Then $\mathsf{A_{sig}}$ initializes the n_j leaf values v_a^j of the tree T_j corresponding to this query by setting $v^j_{\mathbf{a}_j[i]} = \mathbf{c}_j[i]$, with $\mathbf{a}_j[i] = A_{n_j}(i)$ for each $i \in [n_j]$. Then $\mathsf{A_{sig}}$ recursively computes all internal vertex values of T_j by the hash rule $v_a^j = H((v^j_{a,0}, v^j_{a,1}))$ for each $a \in I_{n_j}$, finally reaching the root value v_0^j. $\mathsf{A_{sig}}$ then queries the message $(CEAS_j, n_j, v_0^j)$ to be signed by its S oracle and receives the signature σ_c^j. Finally, $\mathsf{A_{sig}}$ returns the simulated HT signature $\sigma_j = (CEAS, \sigma_c^j, \langle \mathbf{r}[i] \rangle_{i \in [n]})$ to $\mathsf{A_{CES}}$.

Output. Eventually $\mathsf{A_{CES}}$ terminates and outputs a HT CES forgery pair $(\mathbf{M}^*, \sigma^*)$. Let $n^* = \mathsf{len}(\mathbf{M}^*)$, $X^* = \mathsf{CI}(\mathbf{M}^*)$, $P_X^* = \langle P_n(\mathbf{a}[i]) \rangle_{i \in X^*}$ (list of path vertices addresses) and $S_X^* = \langle S_n(\mathbf{a}[i]) \rangle_{i \in X^*}$ (list of path vertices siblings' addresses). $\mathsf{A_{sig}}$ parses the forgery signature as $\sigma^* = (CEAS^*, \sigma_c^*, \langle \mathbf{r}^*[i] \rangle_{i \in [X^*]}, \langle v_a^* \rangle_{a \in S_X^* - P_X^*})$. Then $\mathsf{A_{sig}}$ computes the $|X^*|$ commitments corresponding to the clear submessages in the forgery document $\mathbf{M}^*$ as $\mathbf{c}^*[i] = C(\mathbf{M}^*[i]; \mathbf{r}^*[i])$ for each $i \in X^*$ and initializes the corresponding leaf values v_a^* of the forgery tree T^* as $v^*_{\mathbf{a}^*[i]} = \mathbf{c}^*[i]$, with $\mathbf{a}^*[i] = A_{n^*}(i)$ for each $i \in X^*$. Then $\mathsf{A_{sig}}$ recursively computes the values of the internal vertices of T^* on the paths in P_{X^*} from the clear submessage leaves to the root using the siblings from σ^* and the hash rule $v_a^* = H(v^*_{a,0}, v^*_{a,1})$ for each $a \in I_{n^*}$, finally reaching the root value v_0^*. Then $\mathsf{A_{sig}}$ sets $M^* = (CEAS^*, n^*, v_0^*)$ and outputs the DS forgery pair (M^*, σ_c^*).

This completes the description of $\mathsf{A_{sig}}$. To establish (22), we evaluate the success probability of $\mathsf{A_{sig}}$.

The attack experiments for CES scheme HT, sig. scheme DS, commit. scheme CM and hash scheme CR are denoted using the shortened notations c, d, cm and cr, respectively.

We define the following events. Event BrkDS means that $\mathsf{A_{sig}}$ succeeds in the CMA sense. Event BrkCES means that $\mathsf{A_{CES}}$ succeeds in the CES − UF sense. Event NewM means that the DS forgery message $M^* = (CEAS^*, n^*, v_0^*)$ is not equal to any of the messages $(CEAS_j, n_j, v_0^j)$ for $j \in [q_s]$ queried by $\mathsf{A_{sig}}$ to S when simulating Sig.

First, if BrkCES occurs then $(\mathbf{M}^*, \sigma^*)$ is a valid forgery for HT, and hence by definition of Ver, we have that $(M^*, \sigma_c^*) = ((CEAS^*, n^*, v_0^*), \sigma_c^*)$ is a valid pair for DS. But for $\mathsf{A_{sig}}$ to succeed in CMA sense, it is also necessary (and sufficient) that M^* has not been queried to $\mathsf{S}(sk_s, .)$ by $\mathsf{A_{sig}}$, meaning

that event NewM occurs. So $\mathsf{A_{sig}}$'s success probability can be lower bounded as follows:

$$\begin{aligned}
\mathbf{Succ}^{\mathrm{CMA}}_{\mathsf{A_{sig}},\mathsf{DS}}(k) &\geq \Pr[\mathsf{BrkCES} \wedge \mathsf{NewM}]_d && (23)\\
&= \Pr[\mathsf{BrkCES}]_d - \Pr[\mathsf{BrkCES} \wedge \neg\mathsf{NewM}]_d && (24)\\
&= \mathbf{Succ}^{\mathrm{CES-UF}}_{\mathsf{A_{CES}},\mathsf{CV}}(k) - \Pr[\mathsf{BrkCES} \wedge \neg\mathsf{NewM}]_c, && (25)
\end{aligned}$$

where to get (25) we used the fact that in experiment d, $\mathsf{A_{sig}}$ simulates the view of $\mathsf{A_{CES}}$ exactly as in the original attack experiment c, and hence the events BrkCES and $\mathsf{BrkCES} \wedge \neg\mathsf{NewM}$ have the same probabilities in experiments c and d.

Let Coll denote the event $\mathsf{BrkCES} \wedge \neg\mathsf{NewM}$. Below we will establish the following claim: If Coll occurs then there exists a Sig query index $j^* \in [q_s]$ and a submessage index $i^* \in \mathsf{CI}(\mathbf{M}^*)$ such that

$$(CEAS^*, n^*, v_0^*) = (CEAS_{j^*}, n_{j^*}, v_0^{j^*}) \text{ and } \mathbf{M}^*[i^*] \neq \mathbf{M}_{j^*}[i^*]. \quad (26)$$

We split the event Coll into two subevents CollC and CollH. Event CollC means that Coll occurs so there exist j^* and i^* satisfying (26) and also that $\mathbf{c}^*[i^*] = \mathbf{c}_{j^*}[i^*]$ so that $(\mathbf{M}_{j^*}[i^*]; \mathbf{r}_{j^*}[i^*])$ and $(\mathbf{M}^*[i^*]; \mathbf{r}^*[i^*])$ form a collision pair for the commit. algorithm $C(.;.)$. Event CollH means that Coll occurs but CollC does not, so there exist j^* and i^* satisfying (26) but also $\mathbf{c}^*[i^*] \neq \mathbf{c}_{j^*}[i^*]$. We will construct attackers $\mathsf{A_{com}}$ and $\mathsf{A_{cr}}$ such that their success probabilities are lower bounded respectively as:

$$\mathbf{Succ}^{\mathrm{r-bind}}_{\mathsf{A_{com}},\mathsf{CM}}(k) \geq (\frac{1}{q_s \cdot n_B}) \cdot \Pr[\mathsf{CollC}]_c, \quad (27)$$

and

$$\mathbf{Succ}^{\mathrm{cr}}_{\mathsf{A_{cr}},\mathsf{CR}}(k) \geq \Pr[\mathsf{CollH}]_c. \quad (28)$$

Since $\mathsf{Coll} = \mathsf{CollC} \vee \mathsf{CollH}$ we have $\Pr[\mathsf{Coll}]_c \leq \Pr[\mathsf{CollC}]_c + \Pr[\mathsf{CollH}]_c$, and using (27) and (28) we get

$$\Pr[\mathsf{Coll}]_c \leq q_s \cdot n_B \cdot \mathbf{Succ}^{\mathrm{r-bind}}_{\mathsf{A_{com}},\mathsf{CM}}(k) + \mathbf{Succ}^{\mathrm{cr}}_{\mathsf{A_{cr}},\mathsf{CR}}(k). \quad (29)$$

Substituting (29) in (25) gives the desired bound (22).

We now show that Coll implies the existence of $j^* \in [q_s]$ and $i^* \in \mathsf{CI}(\mathbf{M}^*)$ satisfying (26). First, since Coll implies $\neg\mathsf{NewM}$, there exists j^* such that $(CEAS^*, n^*, v_0^*) = (CEAS_{j^*}, n_{j^*}, v_0^{j^*})$. Also, since Coll implies BrkCES, we know that $(\mathbf{M}^*, \sigma^*)$ is a valid CES signature, meaning $\mathsf{CI}(\mathbf{M}^*) \in CEAS^* = CEAS_{j^*}$. So by def. of BrkCES we must have $\mathbf{M}^* \not\leq \mathbf{M}_{j^*}$. This implies either $n^* \neq n_{j^*}$ (which does not occur since we have already shown above that $n^* = n_{j^*}$), or $\mathsf{CI}(\mathbf{M}^*) \not\subseteq \mathsf{CI}(\mathbf{M}_{j^*})$ (which does not occur because $\mathsf{CI}(\mathbf{M}_{j^*}) = [n_{j^*}]$), or that exists $i^* \in \mathsf{CI}(\mathbf{M}^*)$ satisfying $\mathbf{M}^*[i^*] \neq \mathbf{M}_{j^*}[i^*]$, which is the second part of (26), as required.

It remains to describe the attackers $\mathsf{A_{com}}$ and $\mathsf{A_{cr}}$ and show that they satisfy (27) and (28), respectively.

We now describe attacker $\mathsf{A_{com}} = (\mathsf{A_1}, \mathsf{A_2})$. $\mathsf{A_1}$ is invoked by the r − bind experiment cm on input (k, sp), where k is the security parameter and sp is the commitment scheme parameters from $\mathsf{I}(k)$. The idea is that $\mathsf{A_{com}}$ makes a guess (β, α) for (j^*, i^*) defined above and chooses the corresponding message $\mathbf{M}_\beta[\alpha]$ as one of its binding collision messages, using $\mathbf{M}^*[\alpha]$ as the other, so that if the guess was right and CollC occurs then $\mathsf{A_{com}}$ succeeds to find a commitment collision. The details follow.

Setup. A_1 runs $\mathsf{K}(k)$ to get a DS key pair (sk_s, pk_s). It defines (pk, sk) for HT as in experiment c above. It also chooses uniformly at random a pair of indices $(\beta, \alpha) \in [q_s] \times [n]$ as a guess for the query/submessage index pair (j^*, i^*) corresponding to one of the commitment collision pairs, as defined in (26) above. A_1 then runs the CES − UF attacker $\mathsf{A}_{\mathsf{CES}}$ for scheme HT on input (k, pk).

Sig **Queries.** When $\mathsf{A}_{\mathsf{CES}}$ makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its Sig oracle, A_1 answers it by simulating Sig as done by $\mathsf{A}_{\mathsf{sig}}$. The only differences are as follows. Firstly, A_1 knows sk_s so uses it directly. Secondly, when $j = \beta$, A_1 saves its state in s, and outputs $(s, \mathbf{M}_\beta[\alpha])$. The experiment chooses random $\bar{r}$, computes $\bar{c} = C(\mathbf{M}_\beta[\alpha]; \bar{r})$, and runs A_2 on input $(s, \bar{c}, \bar{r})$. A_2 sets $\mathbf{r}_\beta[\alpha] = \bar{r}$, $\mathbf{c}_\beta[\alpha] = \bar{c}$, and completes the answer to $\mathsf{A}_{\mathsf{CES}}$'s βth Sig query and all subsequent queries as done by $\mathsf{A}_{\mathsf{sig}}$.

Output. Eventually $\mathsf{A}_{\mathsf{CES}}$ terminates and outputs a CES document/extracted-signature forgery pair $(\mathbf{M}^*, \sigma^*)$. A_2 then outputs $(\mathbf{M}^*[\alpha], \mathbf{r}^*[\alpha])$ as the collision pair corresponding to $(\mathbf{M}_\beta[\alpha], \mathbf{r}_\beta[\alpha])$.

This completes the description of $\mathsf{A}_{\mathsf{com}}$. Note that $\mathsf{A}_{\mathsf{com}}$ perfectly simulates the view of $\mathsf{A}_{\mathsf{CES}}$ as in Experiments d and c, regardless of the outcome of (β, α). Hence, for all (j, i), we have

$$\Pr[\mathsf{CollC} \wedge (j^*, i^*) = (j, i) | (\beta, \alpha) = (j, i)]_{cm} = \Pr[\mathsf{CollC} \wedge (j^*, i^*) = (j, i)]_c$$

But if CollC occurs so (26) holds, $\mathbf{c}^*[i^*] = \mathbf{c}_{j^*}[i^*]$ and $\mathsf{A}_{\mathsf{com}}$'s guess (β, α) for (j^*, i^*) was correct then $\mathsf{A}_{\mathsf{com}}$ succeeds because its output is a commitment collision: $\mathbf{M}^*[\alpha] \neq \mathbf{M}_\beta$ but $C(\mathbf{M}^*[\alpha]; \mathbf{r}^*[\alpha]) = C(\mathbf{M}_\beta[\alpha], \mathbf{r}_\beta[\alpha])$. Hence using that (β, α) is uniformly chosen in $[q_s] \times [n]$ we get

$$\begin{aligned}
\mathbf{Succ}^{\mathrm{r-bind}}_{A_{com}, CM}(k) &\geq \Pr[\mathsf{Coll} \wedge (j^*, i^*) = (\beta, \alpha)]_{cm} \\
&= \sum_{(j,i) \in [q_s] \times [n]} \Pr[\mathsf{Coll} \wedge (j^*, i^*) = (j, i) | (\beta, \alpha) = (j, i)]_{cm} \\
&\qquad \cdot \Pr[(\beta, \alpha) = (j, i)]_{cm} \\
&= (\frac{1}{q_s \cdot n}) \cdot \sum_{(j,i) \in [q_s] \times [n]} \Pr[\mathsf{Coll} \wedge (j^*, i^*) = (j, i)]_c \\
&= (\frac{1}{q_s \cdot n}) \Pr[\mathsf{Coll}]_c. \qquad (30)
\end{aligned}$$

The inequality (30) immediately implies (27). The claimed time and query bounds for $\mathsf{A}_{\mathsf{com}}$ are clear from its description.

Finally, we describe attacker A_{cr}. A_{cr} is invoked by the cr experiment cr on input (k, α).

Setup. A_{cr} runs $\mathsf{K}(k)$ to get a DS key pair (sk_s, pk_s) and runs $\mathsf{I}(k)$ to get CM parameters sp. It defines (pk, sk) for HT as in experiment c above. A_{cr} then runs the CES − UF attacker $\mathsf{A}_{\mathsf{CES}}$ for scheme HT on input (k, pk).

Sig **Queries.** When $\mathsf{A}_{\mathsf{CES}}$ makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its Sig oracle, A_{cr} answers it by simulating Sig as done by $\mathsf{A}_{\mathsf{sig}}$. The only difference is that A_{cr} knows sk_s so uses it directly.

Output. Eventually $\mathsf{A}_{\mathsf{CES}}$ terminates and outputs a CES document/extracted-signature forgery pair $(\mathbf{M}^*, \sigma^*)$. A_{cr} then searches for $(j^*, i^*) \in [q_s] \times \mathsf{CI}(\mathbf{M}^*)$ satisfying (26) and $\mathbf{c}^*[i^*] \neq \mathbf{c}_{j^*}[i^*]$. Note that if event CollH occurs then such a (j^*, i^*) pair exists and can be found in time $O(q_s \cdot (n_B \cdot \ell + l_{CEAS} + l_H))$. If A_{cr} finds (j^*, i^*) (else A_{cr} fails), it finds a collision pair for $H(.)$ as follows. Let $a^* = A_{n^*}(i^*) = (0, a^*_m, \ldots, a^*_0)$ denote the address of the leaf vertices in the trees T^* and T_{j^*} whose values are $v^*_{a^*} = \mathbf{c}^*[i^*]$ and $v^{j^*}_{a^*} = \mathbf{c}_{j^*}[i^*]$ respectively. Consider the sequence of

vertex values along the path $P_{n^*}(a^*)$ from the root to the leaf at address a^*. In the forgery tree T^* this sequence is $L^* = \langle v^*_0, \ldots, v^*_{0,a^*_m,\ldots,a^*_0}\rangle$, whereas in the j^*th Sig query tree T_{j^*} this sequence is $L_{j^*} = \langle v^{j^*}_0, \ldots, v^{j^*}_{0,a^*_m,\ldots,a^*_0}\rangle$. Note that the sequences L^* and L^{j^*} have a common first (root) value (since $v^*_0 = v^{j^*}_0$) but an unequal last (leaf) value (since $v^*_{0,a^*_m,\ldots,a^*_0} = \mathbf{c}^*[i^*] \neq \mathbf{c}_{j^*}[i^*] = v^{j^*}_{0,a^*_m,\ldots,a^*_0}$). It follows that these sequences must first differ in the value of some internal vertex along the path. Let k^* denote the level of the first internal vertex along the path (starting at the root) where the sequences L^* and L_{j^*} differ. $\mathsf{A_{cr}}$ finds k^* (this takes time $O(\log n_B \cdot l_H)$) and outputs the collision pair (z_1, z_2), where $z_1 = Y_{a^*_{k^*}}(v^*_{0,a^*_m,\ldots,a^*_{k^*}}, v^*_{0,a^*_m,\ldots,\bar{a}^*_{k^*}})$ and $z_2 = Y_{a^*_{k^*}}(v^{j^*}_{0,a^*_m,\ldots,a^*_{k^*}}, v^{j^*}_{0,a^*_m,\ldots,\bar{a}^*_{k^*}})$.

This completes the description of $\mathsf{A_{cr}}$. The claimed RPs are clear from this description. Note that $\mathsf{A_{cr}}$ perfectly simulates the view of $\mathsf{A_{CES}}$ as in Experiment c. Hence $\Pr[\mathsf{CollH}]_{cr} = \Pr[\mathsf{CollH}]_c$. We show below that if CollH occurs then $\mathsf{A_{cr}}$ succeeds. This immediately implies the desired bound (28). If CollH occurs then by definition of k^* we have

$$v^*_{0,a^*_m,\ldots,a^*_{k^*}} \neq v^{j^*}_{0,a^*_m,\ldots,a^*_{k^*}} \text{ but } v^*_{0,a^*_m,\ldots,a^*_{k^*+1}} = v^{j^*}_{0,a^*_m,\ldots,a^*_{k^*+1}}, \tag{31}$$

but on the other hand, from the hash rule used to construct the trees T^* and T_{j^*} we also have

$$v^*_{0,a^*_m,\ldots,a^*_{k^*+1}} = H(Y_{a^*_{k^*}}(v^*_{0,a^*_m,\ldots,a^*_{k^*}}, v^*_{0,a^*_m,\ldots,\bar{a}^*_{k^*}})) \tag{32}$$

and

$$v^{j^*}_{0,a^*_m,\ldots,a^*_{k^*+1}} = H(Y_{a^*_{k^*}}(v^{j^*}_{0,a^*_m,\ldots,a^*_{k^*}}, v^{j^*}_{0,a^*_m,\ldots,\bar{a}^*_{k^*}})). \tag{33}$$

Combining (31), (32) and (33) we see that CollH implies that $\mathsf{A_{cr}}$'s output (z_1, z_2) is a collision for $H(.)$ so $\mathsf{A_{cr}}$ succeeds in the cr sense, as required. This completes the proof of Part (1) of the Theorem.

Proof of (2). We show how to use any efficient CES-Privacy attacker $\mathsf{A_{CES}} = (\mathsf{A_1}, \mathsf{A_2})$ to construct an efficient attacker $\bar{\mathsf{A}}_{\mathsf{com}} = (\bar{\mathsf{A}}_1, \bar{\mathsf{A}}_2)$ to break the hiding property of commitment scheme CM, thus contradicting the assumed hiding property of the latter. More precisely, we show that:

$$\mathbf{Succ}^{\mathrm{CES-PR}}_{\mathsf{A_{CES}},\mathsf{HT}}(k) \leq \mathbf{Succ}^{\mathrm{hide}}_{\bar{\mathsf{A}}_{\mathsf{com}},\mathsf{CM}}(k), \tag{34}$$

where $\mathsf{A_{CES}}$ has Resource Parameters (RPs) (t, n, ℓ), and $\bar{\mathsf{A}}_{\mathsf{com}}$ has RPs $(t[CM], \ell[CM])$ with $t[CM] = t$ and $\ell[CM] = \ell$. The theorem then follows immediately from (34) by taking maximums over all attackers $\bar{\mathsf{A}}_{\mathsf{com}}$ with the given RPs. It remains to describe the attacker $\bar{\mathsf{A}}_{\mathsf{com}}$ and show that it satisfies (34).

We will denote the CES − PR notion experiment **CESPRExp** for HT by c and the hide notion exp. **HideExp** for CM by cm.

We now describe attacker $\bar{\mathsf{A}}_{\mathsf{com}}$. It is invoked by exp. cm on input (k, sp), where k is the security parameter and sp is the commitment scheme parameters from $\mathsf{I}(k)$. The idea is straightforward: $\bar{\mathsf{A}}_{\mathsf{com}}$ will run $\mathsf{A_{CES}}$, incorporate its challenge commitment into the challenge extracted signature given to $\mathsf{A_{CES}}$, so that $\bar{\mathsf{A}}_{\mathsf{com}}$ can just use $\mathsf{A_{CES}}$'s prediction bit to solve its challenge. The details follow.

Setup. $\bar{\mathsf{A}}_1$ runs $\mathsf{K}(k)$ to get a DS key pair (sk_s, pk_s). It defines (pk, sk) for HT as in the GK algorithm for HT. It then runs $\mathsf{A_1}$ on input (k, sk).

End of $\bar{A}_{com}$'s Find Stage. Eventually A_1 terminates and outputs $s' = (s, \mathbf{M}^*, M_0, M_1, i^*, CEAS^*, X^*)$. $\bar{\mathsf{A}}_1$ then saves its state into $\bar{s} = (k, sp, (sk, pk), s')$, and uses (M_0, M_1) as the message pair to be challenged on. It terminates and outputs $(\bar{s}, M_0, M_1)$.

$\bar{\mathsf{A}}_{\mathsf{com}}$'s Challenge. The *cm* experiment chooses a random $b \in \{0, 1\}$, computes challenge commitment $c^* = C(M_b; r^*)$ for a random string r^*, and runs A_2 on input $(\bar{s}, c^*)$.

$\bar{\mathsf{A}}_{\mathsf{com}}$'s Guess Stage. $\bar{\mathsf{A}}_2$ retrieves $(k, sp, (sk, pk), s')$ from $\bar{s}$. It then simulates an extracted signature σ_E^* for input document $\mathbf{M}^*$ and extraction subset X^* (using c^* for the i^*th commitment). The details follow. Let $n^* = \mathsf{len}(\mathbf{M}^*)$. For $i \in [n^*] - \{i^*\}$, $\bar{\mathsf{A}}_2$ computes $\mathbf{c}^*[i] = C(\mathbf{M}^*[i]; \mathbf{r}^*[i])$ using random strings $\mathbf{r}^*[i]$. For $i = i^*$, it sets $\mathbf{c}^*[i^*] = c^*$. Then $\bar{\mathsf{A}}_2$ computes the tree root value v_0^* as done by Sig and computes the signature $\bar{\sigma^*} = \mathsf{S}(sk_s, (CEAS^*, n^*, v_0^*))$. Finally it sets the simulated extracted CES signature to $\sigma_E^* = (CEAS^*, \bar{\sigma^*}, \langle \mathbf{r}[i] \rangle_{i \in [X]}, \langle v_a \rangle_{a \in S_X^* - P_X^*})$ and runs A_2 on input (s, σ_E^*).

Output. Eventually A_2 terminates and outputs a guess bit b' for b. Then $\bar{\mathsf{A}}_2$ just output b' as its guess for b.

This completes the description of $\bar{\mathsf{A}}_{\mathsf{com}}$. To establish (34), we just observe that since $\bar{\mathsf{A}}_{\mathsf{com}}$ in experiment *cm* perfectly simulates (when $i^* \notin X^*$) the view of $\mathsf{A}_{\mathsf{CES}}$ as in experiment *c*, where the bit b which $\mathsf{A}_{\mathsf{com}}$ needs to guess is equal to the bit which $\mathsf{A}_{\mathsf{CES}}$ needs to guess. Therefore $\Pr[b' = b]_{cm} \geq \Pr[b' = b \wedge (i^* \notin X^*)]_{cm} = \Pr[b' = b \wedge (i^* \notin X^*)]_c$, which immediately implies the desired result (34). The stated RPs follow directly from the specification of $\bar{\mathsf{A}}_{\mathsf{com}}$. This completes the proof of part (2). □

Remark: In both $\mathsf{HashTree}$ and $\mathsf{CommitVector}$ 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 PseudoRandom 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 [36]. 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 [17] in the context of 'hierarchical access control'.

5.2 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.

5.2.1 Scheme RSAProd (RSAP)

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 can be considered a modification of an RSA batch "screening" verifier, proposed by Bellare, Garay and Rabin [6]. 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.

The precise definition of the scheme RSAP is given in Fig. 8.

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 [6], 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 [6] in order to avoid the attack described in [19]. 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.

Scheme RSAProd (RSAP)

Algorithm $\mathsf{GK}(k)$
  Pick random $k/2$-bit primes (p, q)
  $N \leftarrow pq$; $\phi \leftarrow (p-1)(q-1)$
  Choose e such that $\gcd(e, \phi) = 1$
  $d \leftarrow e^{-1} \bmod \phi$
  $pk \leftarrow (N, e)$; $sk \leftarrow (N, e, d)$
  `Return` (sk, pk)

Algorithm $\mathsf{Sig}(sk, \mathbf{M}, CEAS)$
  $n \leftarrow \mathsf{len}(\mathbf{M})$
  Parse $pk = (N, e)$; $sk = (N, e, d)$
  $T \xleftarrow{\mathrm{R}} \{0,1\}^{l_{tag}}$
  for $i \in [n]$
    $\mathbf{h}[i] \leftarrow H(CEAS, T, n, i, \mathbf{M}[i])$
    $\sigma[i] \leftarrow \mathbf{h}[i]^d \bmod N$
  $\sigma_F \leftarrow (CEAS, T, \langle \sigma[i] \rangle_{i \in [n]})$
  `Return` σ_F

Algorithm $\mathsf{Ext}(pk, \mathbf{M}, \sigma_F, X)$
  Parse $pk = (N, e)$
  Parse $\sigma_F = (CEAS, T, \langle \sigma[i] \rangle_{i \in [n]})$
  $\sigma \leftarrow \prod_{i \in X} \sigma[i] \bmod N$
  $\sigma_E \leftarrow (CEAS, T, \sigma)$
  `Return` σ_E

Algorithm $\mathsf{Ver}(pk, \mathbf{M}', \sigma_E)$
  $X' \leftarrow \mathsf{CI}(\mathbf{M}')$; $n \leftarrow \mathsf{len}(\mathbf{M}')$
  Parse $pk = (N, e)$; $\sigma_E = (CEAS, T, \sigma)$
  For $i \in X'$
    $\mathbf{h}[i] \leftarrow H(CEAS, T, n, i, \mathbf{M}'[i])$
  If
    $\sigma^e \equiv \prod_{i \in X'} \mathbf{h}[i] \pmod N$ and
    $X' \in CEAS$
  `Return` Acc
  Else `Return` Rej.

Figure 8: Definition of CES scheme RSAP.

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 5.3. *(1) If the Full-Domain-Hash standard signature scheme* $\mathsf{FDH}-\mathsf{RSA}$ *is existentially unforgeable under chosen message attack (*CMA *notion) then CES scheme* RSAP *has the CES-Unforgeability property (*$\mathrm{CES}-\mathrm{UF}$ *notion). Concretely, the following bound holds:*

$$\mathbf{InSec}^{\mathrm{CES}-\mathrm{UF}}_{\mathsf{RSAP}}(t, q_s, n_B, \ell, q_H[F]) \leq \mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{FDH}-\mathsf{RSA}}(t[F], q_s[F], \ell[F], q_H[F]) + \frac{q_s(q_s-1)}{2^{l_{tag}+1}} \tag{35}$$

where $t[F] = t + O(q_s \cdot n_B)$; $q_s[F] = (q_s+1)\cdot n_B$; $\ell[F] = \ell + l_{CEAS} + l_{tag} + 2\log(n_B)$; $q_H[F] = (q_s+1)\cdot n_B$.

(2) Scheme RSAP *has the CES-Privacy property (perfectly and unconditionally). That is,* $\mathbf{InSec}^{\mathrm{CES}-\mathrm{PR}}_{\mathsf{RSAP}}(\infty, n_B, \ell) = 0$.

Proof. Proof of (1). First, we observe that scheme RSAP is obtained by applying the generic security-enhancing transform $T_{WO}(.)$ (see Definition A.2 in Appendix) to the simpler CES scheme RSAP_1 which does not append $(CEAS, T, n)$ to the submessages during signing. Applying Lemma A.1, we have

$$\mathbf{InSec}^{\mathrm{CES}-\mathrm{UF}}_{\mathsf{RSAP}}(t, q_s, n_B, \ell, q_H) \leq \mathbf{InSec}^{\mathrm{WO}-\mathrm{UF}}_{\mathsf{RSAP}_1}(t', q'_s, n'_B, \ell', q'_H) + \frac{q_s(q_s-1)}{2^{l_{tag}+1}}, \tag{36}$$

where $t' = t$; $q'_s = q_s$; $n'_B = n_B$; $\ell' = \ell + l_{CEAS} + l_{tag} + \log(n_B)$.

To establish (35), it therefore suffices to bound the 'Weak-Ordered' ($\mathrm{WO}-\mathrm{UF}$ notion) insecurity of RSAP_1 in terms of $\mathsf{FDH}-\mathsf{RSA}$, namely to show that

$$\mathbf{InSec}^{\mathrm{WO}-\mathrm{UF}}_{\mathsf{RSAP}_2}(t', q'_s, n'_B, \ell', q'_H) \leq \mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{FDH}-\mathsf{RSA}}(t[F], q_s[F], n_B[F], \ell[F], q_H[F]), \tag{37}$$

where $t[F] = t' + O(q'_s \cdot n'_B)$; $q_s[F] = (q'_s+1)\cdot n'_B$; $\ell[F] = \ell' + \log(n'_B)$; $q_H[F] = (q'_s+1)\cdot n'_B$. We show (37) by using an efficient attacker $\mathsf{A_C}$ for breaking CES scheme $\mathsf{RSAP}_1 = (\mathsf{GK}_1, \mathsf{Sig}_1, \mathsf{Ext}_1, \mathsf{Ver}_1)$ in the sense of $\mathrm{WO}-\mathrm{UF}$ to construct an efficient attacker $\mathsf{A_F}$ for breaking scheme $\mathsf{FDH}-\mathsf{RSA} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ in the sense of CMA. By lower bounding the success probability of $\mathsf{A_F}$ in terms of that of $\mathsf{A_C}$, we will derive the desired bound (37).

Attacker $\mathsf{A_F}$ works as follows. $\mathsf{A_F}$ is invoked by the CMA experiment on input $(k, (N, e))$, where k is the security parameter and (N, e) is a public key for $\mathsf{FDH}-\mathsf{RSA}$, with (N, e, d) denoting the corresponding secret key. $\mathsf{A_F}$ is able to query messages to be signed by the signing oracle S for scheme $\mathsf{FDH}-\mathsf{RSA}$.

Setup. $\mathsf{A_F}$ runs $\mathsf{A_C}$ on input $(k, (N, e))$.

Sig **Queries.** When $\mathsf{A_C}$ makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its Sig_1 oracle, $\mathsf{A_F}$ answers it by simulating the action of the Sig_1 algorithm, using in turn the S oracle. The details follow. Let $n_j = \mathsf{len}(\mathbf{M}_j)$. For each $i \in [n_j]$, $\mathsf{A_F}$ defines the message $\bar{\mathbf{M}}_j[i] = (i, \mathbf{M}_j[i])$ and queries this message to S to obtain signature $\bar{\sigma}_j[i]$. Finally, $\mathsf{A_F}$ sets the simulated signature to $\sigma_j = (\langle \bar{\sigma}_j \rangle_{i \in [n_j]})$. We assume that $\mathsf{A_F}$ stores the documents $\mathbf{M}_j$ in memory.

Output. Eventually $\mathsf{A_C}$ terminates and outputs a forgery pair $(\mathbf{M}^*, \sigma^*)$. Let $n^* = \mathsf{len}(\mathbf{M}^*)$ and $X^* = \mathsf{CI}(\mathbf{M}^*)$. $\mathsf{A_F}$ now searches through the (stored) queried documents $\mathbf{M}_j$ and tries to find a submessage index $i^* \in X^*$ such that $\mathbf{M}^*[i^*] \neq \mathbf{M}_j[i^*]$ for all $j \in [q_s]$ (this takes time $O(q_s \cdot n)$). If $\mathsf{A_F}$ finds such an i^* (else $\mathsf{A_F}$ fails), it queries, for each $i \in X^* - \{i^*\}$, the message $\bar{\mathbf{M}}^*[i] = (i, \mathbf{M}^*[i])$ to the signing oracle S to get the signature $\sigma^*[i] = H(\bar{\mathbf{M}}^*[i])^d \bmod N$. Then $\mathsf{A_F}$ computes the product $R = \prod_{i \in X^* - \{i^*\}} \sigma^*[i] \bmod N$ and computes the forged signature $\bar{\sigma}^* = \sigma^*/R \bmod N$. Finally, $\mathsf{A_F}$ outputs the forgery pair $(\bar{\mathbf{M}}^*[i^*], \bar{\sigma}^*)$ for scheme $\mathsf{FDH-RSA}$.

This completes the description of $\mathsf{A_F}$. The claimed RPs of $\mathsf{A_F}$ are easily verified from this description. To complete the proof of the lemma, it remains to evaluate the success (in the CMA sense) probability of $\mathsf{A_F}$.

We use the notation $\Pr[.]_{exp}$ to denote probability in experiment *exp*. The attack experiments for CES scheme $\mathsf{RSAP_1}$(WO − UF notion), and $\mathsf{FDH-RSA}$ (CMA notion) are denoted by subscripts c and d, respectively.

We define the following events. Event $\mathsf{Succ_C}$ means that $\mathsf{A_C}$ succeeds in the sense of WO − UF. Recall that $\mathsf{Succ_C}$ means that $(\bar{\mathbf{M}}^*, \bar{\sigma}^*)$ is a valid pair for $\mathsf{RSAP_1}$ (call this event $\mathsf{Valid_C}$), and also there exists $i^* \in X^*$ such that $\mathbf{M}^*[i^*] \neq \mathbf{M}_j[i^*]$ for all $j \in [q_s]$ (call this event $\mathsf{NewM_C}$). Event $\mathsf{Succ_F}$ means that $\mathsf{A_F}$ succeeds in the sense of CMA. Recall that $\mathsf{Succ_F}$ means that $(\bar{\mathbf{M}}^*, \bar{\sigma}^*)$ is a valid pair for $\mathsf{FDH-RSA}$ (call this event $\mathsf{Valid_F}$), and also the message $\bar{\mathbf{M}}^*$ was never queried to S (call this event $\mathsf{NewM_F}$).

We show below that if $\mathsf{Succ_C}$ occurs then $\mathsf{Succ_F}$ occurs. Since $\mathsf{A_F}$ simulates the view of $\mathsf{A_C}$ in exp. d exactly as in the real attack exp. c, it follows that the event $\mathsf{Succ_C}$ has the same probability in both experiments. Therefore $\mathbf{Succ}^{\mathrm{CMA}}_{A_F, FDH-RSA}(k) \geq \mathbf{Succ}^{\mathrm{CES-UF}}_{A_C, RSAP}(k)$, which implies (37), as required.

To complete the proof, it remains to show that $\mathsf{Succ_C}$ implies $\mathsf{Succ_F}$. First $\mathsf{Succ_C}$ implies $\mathsf{Valid_C}$, meaning that $\sigma^* = \prod_{i \in X^*} H(\bar{\mathbf{M}}^*[i])^d \bmod N$ so $\bar{\sigma}^* = \sigma^*/R \bmod N = H(\bar{\mathbf{M}}^*[i^*])^d \bmod N$ is a valid $\mathsf{FDH-RSA}$ signature on message $\bar{\mathbf{M}}^*[i^*]$ and $\mathsf{Valid_F}$ occurs. To show that $\mathsf{NewM_F}$ also occurs, note that if $\mathsf{NewM_C}$ occurs then $\mathbf{M}^*[i^*] \neq \mathbf{M}^*_j[i^*]$ for all $j \in [q_s]$, which implies that $\bar{\mathbf{M}}^*[i^*] \neq \bar{\mathbf{M}}_j[i^*]$ for all $j \in [q_s]$, and all other queries of $\mathsf{A_F}$ to S (namely $\bar{\mathbf{M}}_j[i]$ and $\bar{\mathbf{M}}^*[i]$ for $i \neq i^*$ and $j \in [q_s]$) differ from $\bar{\mathbf{M}}^*[i^*]$ in the submessage index prefix. So if $\mathsf{Succ_C}$ occurs the forgery message was never queried to S, i.e. $\mathsf{NewM_F}$ occurs, as required. This completes the proof of claim (1).

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

If we use the random oracle model [10] for the hash function $H(.)$, then Theorem 5.3 and Lemma 3.1 in § 3.3.2 immediately give the following result.

Corollary 5.1. *If the* RSA *function is One-Way (*OW *notion) then CES scheme* RSAP *has the CES-Unforgeability property (*CES − UF *notion) in the random-oracle model. Concretely:*

$$\mathbf{InSec}^{\mathrm{CES-UF}}_{\mathsf{RSAP}}(t, q_s, n_B, \ell, q_H) \leq \quad \exp(1) \cdot (q_s + 1) \cdot n_B \cdot \mathbf{InSec}^{\mathrm{OW}}_{\mathsf{RSA}}(t') \tag{38}$$

where $t' = t + O([n_B(q_s+1)+q_H]^2 \cdot (\ell + l_{CEAS} + l_{tag} + \log n_B + 1) + [1 + n_B(q_s+1)(n_B(q_s+1)+q_H)]k^3)$.

Remark 1: To achieve CES-unforgeability it is essential that CES tags are never re-used by the Sign algorithm. In the unstateful version of Sig described above, the CES tag T is chosen uniformly at random in $\{0,1\}^{l_{tag}}$ for each signed document. Hence one needs a tag length of $l_{tag} = 160$ bit for up to 2^{80} CES signatures to avoid birthday collisions with high probability. It is also possible to use a

stateful version of Sig which keeps a counter for T which is incremented by 1 after each document is signed — in this case a tag length of $l_T = 80$ bit suffices for up to 2^{80} CES signatures.

Remark 2: This scheme does *not* have the optional 'Iterative Extraction' property described in §4. 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 5.3 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.

5.2.2 A Variant: Scheme MERSAProd (MERP)

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 MERP 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 [21] 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 [21] discovered that while standard RSA is hard to batch (see [35]), '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 $\mathbf{e}[i]$, pairwise relatively prime and prime to $(p-1)(q-1)$ (for efficiency these n_B public exponents are normally chosen as the first n_B odd prime numbers). To each public exponent $\mathbf{e}[i]$ there exists a corresponding secret exponent $\mathbf{d}[i] \equiv \mathbf{e}[i]^{-1} \bmod (p-1)(q-1)$. Given a sequence $\mathbf{h}[1], ..., \mathbf{h}[n]$ 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 $\mathbf{h}[i]^{\mathbf{d}[i]} \bmod N$. Fiat's algorithm is divided into 3 stages. Stage 1 outputs the product $r \stackrel{\text{def}}{=} \Pi_{i=1}^{n} \mathbf{h}[i]^{e/\mathbf{e}[i]} \bmod N$, where $e \stackrel{\text{def}}{=} \prod_{i=1}^{n} \mathbf{e}[i]$. Stage 2 signs the product and outputs $r^{1/e} = \prod_{i=1}^{n} \mathbf{h}[i]^{1/\mathbf{e}[i]} \bmod N$. Stage 3 'separates' the product into the n indvidual multi-exponent signatures $\mathbf{h}[i]^{1/\mathbf{e}[i]} \bmod N$. 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 MERP is a natural extension of scheme RSAP to the ME-RSA setting, 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 $\mathbf{e}[i]$ $(i = 1, ..., n_B)$ as public exponents, keeping the corresponding $\mathbf{d}[i]$ as secret exponents. The Sign algorithm runs only stages 1 and 2 of Fiat's algorithm to compute the product $r = \prod_{i \in [n]} \mathbf{h}[i]^{\mathbf{d}[i]} \bmod N$, where $\mathbf{h}[i]$ denotes the hash of the ith submessage (with appended CEAS, CES Tag and submessage sequence number i, as in scheme RSAP). Only the product r (single RSA signature length) is sent to the user (together with CEAS and CES Tag). The Ext algorithm runs stage 3 of Fiat's algorithm to break up the product r into the individual signatures $\mathbf{h}[i]^{\mathbf{d}[i]}$ and then multiplies the ones corresponding to the extracted submessages in the extraction subset X as in scheme RSAP, to compute σ_E to achieve the same single RSA signature length of extracted signatures. The Ver algorithm verifies σ_f on document $\mathbf{M}'$ by checking if $\sigma_E^e = \prod_{i \in X} \mathbf{h}[i]^{e/\mathbf{e}[i]} \bmod N$, where $e \stackrel{\text{def}}{=} \prod_{i \in X} \mathbf{e}[i]$, and $X = \mathsf{CI}(\mathbf{M}')$. Fiat's stage 1 algorithm is again used to perform the the verification efficiently (see the precise definition for details).

The precise definition of the scheme MERP is given in Fig. 9.

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. As for scheme RSAP, we first examine the security of MERP in the standard model (no random oracle assumption on $H(.)$). We will show that the CES-unforgeability of MERP follows from the standard (CMA) unforgeability of a 'Multiple-Exponent' variant of the $\mathsf{FDH-RSA}$ signature scheme, which we call $\mathsf{MER-FDH}$ (see § 3.3.3 for a precise definition and analysis of this scheme). The proof is analogous to the proof of Theorem 5.3.

Theorem 5.4. *(1) If the Multiple-Exponent RSA variant of the Full-Domain-Hash standard signature scheme* $\mathsf{FDH-MER}$ *is existentially unforgeable under chosen message attack (*CMA *notion) then CES scheme* MERP *has the CES-Unforgeability property (*$\mathrm{CES-UF}$ *notion). Concretely, the following bound holds:*

$$\mathbf{InSec}^{\mathrm{CES-UF}}_{\mathsf{MERP}}(t, q_s, n_B, \ell, q_H) \leq \mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{FDH-MER}}(t[F], q_s[F], \ell[F], q_H[F]) + \frac{q_s(q_s-1)}{2^{l_{tag}+1}} \tag{39}$$

where $t[F] = t + O(q_s \cdot n_B)$; $q_s[F] = (q_s+1) \cdot n_B$; $\ell[F] = \ell + l_{CEAS} + l_{tag} + 2\log(n_B)$; $q_H[F] = (q_s+1) \cdot n_B + q_H$.

(2) Scheme MERP *has the CES-Privacy property (perfectly and unconditionally). That is,* $\mathbf{InSec}^{\mathrm{CES-PR}}_{\mathsf{MERP}}(\infty, n_B, \ell) = 0$.

Proof. Proof of (1). First, we observe that scheme MERP is obtained by applying the generic security-enhancing transform $T_{WO}(.)$ (see Definition A.2) to the simpler CES scheme $\mathsf{MERP_1}$ which does not append $(CEAS, T, n)$ to the submessages during signing. Applying Lemma A.1, we have

$$\mathbf{InSec}^{\mathrm{CES-UF}}_{\mathsf{MERP}}(t, q_s, n_B, \ell, q_H) \leq \mathbf{InSec}^{\mathrm{WO-UF}}_{\mathsf{MERP_1}}(t', q'_s, n'_B, \ell', q'_H) + \frac{q_s(q_s-1)}{2^{l_{tag}+1}}, \tag{40}$$

where $t' = t$; $q'_s = q_s$; $n'_B = n_B$; $\ell' = \ell + l_{CEAS} + l_{tag} + \log(n_B)$.

Scheme MERSAProd (MERP)

Algorithm $\mathsf{GK}(k)$
for $i \in [n_B]$, let $\mathbf{e}[i]$ denote ith odd prime
Pick random $k/2$-bit primes (p, q)
such that $(p-1)/2$ and $(q-1)/2$ are $\mathbf{e}[n_B]$-strong
$N \leftarrow pq$; $\phi \leftarrow (p-1)(q-1)$
for $i \in [n_B]$, $\mathbf{d}[i] \leftarrow \mathbf{e}[i]^{-1} \bmod \phi$
$d \leftarrow \prod_{i \in [n_B]} \mathbf{d}[i] \bmod \phi$
$pk \leftarrow (N, \mathbf{e})$; $sk \leftarrow (N, \mathbf{e}, d)$
`Return` (sk, pk)

Algorithm $\mathsf{Sig}(sk, \mathbf{M}, CEAS)$
$n \leftarrow \mathsf{len}(\mathbf{M})$;
Parse $pk = (N, \mathbf{e})$; $sk = (N, \mathbf{e}, d)$
$T \xleftarrow{\mathrm{R}} \{0,1\}^{l_{tag}}$
for $i \in [n]$
$\mathbf{h}[i] \leftarrow H(CEAS, T, n, i, \mathbf{M}[i])$
$r \leftarrow \prod_{i \in [n]} \mathbf{h}[i]^{e/\mathbf{e}[i]} \bmod N$
(Use Fiat 'Stage 1' alg., $e \stackrel{\text{def}}{=} \prod_{i \in [n]} \mathbf{e}[i]$).
$\sigma \leftarrow r^d \bmod N$
$\sigma_F \leftarrow (CEAS, T, \sigma)$
`Return` σ_F

Algorithm $\mathsf{Ext}(pk, \mathbf{M}, \sigma_F, X)$
$n \leftarrow \mathsf{len}(\mathbf{M})$
Let $\mathbf{c}$ CRT coeff. vec. for moduli $\mathbf{e}$
(i.e. $\mathbf{c}[k] \equiv 1 \bmod \mathbf{e}[k]$ and
$\mathbf{c}[k] \equiv 0 \bmod \mathbf{e}[i]$ for $i \in [n] - \{k\}$)
Parse $pk = (N, \mathbf{e})$
Parse $\sigma_F = (CEAS, T, \sigma)$
for $i \in [n]$
$\mathbf{h}[i] \leftarrow H(CEAS, T, n, i, \mathbf{M}[i])$
for $k \in [n]$
$\sigma[k] \leftarrow \sigma^{\mathbf{c}[k]} / \prod_{i \in [n]} \mathbf{h}[i]^{\lfloor \mathbf{c}[k]/\mathbf{e}[i] \rfloor} \bmod N$
(Use Fiat 'Stage 3' alg.)
$\bar{\sigma} \leftarrow \prod_{i \in X} \sigma[i] \bmod N$
$\sigma_E \leftarrow (CEAS, T, \bar{\sigma})$
`Return` σ_E

Algorithm $\mathsf{Ver}(pk, \mathbf{M}', \sigma_E)$
$X' \leftarrow \mathsf{CI}(\mathbf{M}')$; $n \leftarrow \mathsf{len}(\mathbf{M}')$
Parse $pk = (N, \mathbf{e})$; $\sigma_E = (CEAS, T, \sigma)$
$i_m \leftarrow \min_{i \in X'} i$
for $i \in X' - \{i_m\}$
$\bar{\mathbf{h}}[i] \leftarrow H(CEAS, T, n, i, \mathbf{M}'[i])$
$\mathbf{h}[i_m] \leftarrow H(CEAS, T, n, i_m, \mathbf{M}'[i_m])$
$\bar{\mathbf{h}}[i_m] \leftarrow \mathbf{h}[i_m] / \bar{\sigma}^{\mathbf{e}[i_m]} \bmod N$
$\bar{r} \leftarrow \prod_{i \in X'} \bar{\mathbf{h}}[i]^{e/\mathbf{e}[i]} \bmod N$
(Use Fiat 'Stage 1' alg., $e \stackrel{\text{def}}{=} \prod_{i \in [X']} \mathbf{e}[i]$)
if $\bar{r} = 1$ and $X' \in CEAS$
`Return` Acc
else `Return` Rej.

Figure 9: Definition of CES scheme MERP.

To establish (39), it therefore suffices to bound the 'Weak-Ordered' ($\mathrm{WO}-\mathrm{UF}$ notion) insecurity of $\mathsf{MERP_1}$ in terms of $\mathsf{FDH}-\mathsf{MER}$, namely to show that

$$\mathbf{InSec}^{\mathrm{WO-UF}}_{\mathsf{MERP_1}}(t', q'_s, n'_B, \ell', q'_H) \leq \mathbf{InSec}^{\mathrm{CMA}}_{\mathsf{FDH-MER}}(t[F], q_s[F], n_B[F], \ell[F], q_H[F]), \tag{41}$$

where $t[F] = t' + O(q'_s \cdot n'_B)$; $q_s[F] = (q'_s + 1) \cdot n'_B$; $\ell[F] = \ell' + \log(n'_B)$; $q_H[F] = (q'_s + 1) \cdot n'_B$. We show (41) by using an efficient attacker $\mathsf{A_C}$ for breaking CES scheme $\mathsf{MERP_1} = (\mathsf{GK_1}, \mathsf{Sig_1}, \mathsf{Ext_1}, \mathsf{Ver_1})$ in the sense of $\mathrm{WO}-\mathrm{UF}$ to construct an efficient attacker $\mathsf{A_F}$ for breaking scheme $\mathsf{FDH}-\mathsf{MER} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ in the sense of CMA. By lower bounding the success probability of $\mathsf{A_F}$ in terms of that of $\mathsf{A_C}$, we will derive the desired bound (41).

Attacker $\mathsf{A_F}$ works as follows. $\mathsf{A_F}$ is invoked by the CMA experiment on input $(k, (N, \mathbf{e}))$. $\mathsf{A_F}$ is able to query messages to be signed by the sig. oracle S for $\mathsf{FDH}-\mathsf{MER}$.

Setup. $\mathsf{A_F}$ runs $\mathsf{A_C}$ on input $(k, (N, \mathbf{e}))$.

Sig **Queries.** When $\mathsf{A_C}$ makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its $\mathsf{Sig_1}$ oracle, $\mathsf{A_F}$ answers it by simulating the action of the $\mathsf{Sig_1}$ algorithm, using in turn the S oracle. The details follow. Let $n_j = \mathsf{len}(\mathbf{M}_j)$. For each $i \in [n_j]$, $\mathsf{A_F}$ defines the message $\bar{\mathbf{M}}_j[i] = (i, \mathbf{M}_j[i])$ and queries this message to S to obtain signature $\bar{\sigma}_j[i]$. Finally, $\mathsf{A_F}$ sets the simulated signature to $\sigma_j = \prod_{i \in [n_j]} \bar{\sigma}_j[i] \bmod N$. We assume that $\mathsf{A_F}$ stores the documents $\mathbf{M}_j$ in memory.

Output. Eventually $\mathsf{A_C}$ terminates and outputs a forgery pair $(\mathbf{M}^*, \sigma^*)$. Let $n^* = \mathsf{len}(\mathbf{M}^*)$ and $X^* = \mathsf{CI}(\mathbf{M}^*)$. $\mathsf{A_F}$ now searches through the (stored) queried documents $\mathbf{M}_j$ and tries to find a submessage index $i^* \in X^*$ such that $\mathbf{M}^*[i^*] \neq \mathbf{M}_j[i^*]$ for all $j \in [q_s]$ (this takes time $O(q_s \cdot n_B)$). If $\mathsf{A_F}$ finds such an i^* (else $\mathsf{A_F}$ fails), it queries, for each $i \in X^* - \{i^*\}$, the message $\bar{\mathbf{M}}^*[i] = (i, \mathbf{M}^*[i])$ to the signing oracle S to get the signature $\sigma^*[i] = H(\bar{\mathbf{M}}^*[i])^{1/\mathbf{e}[i]} \bmod N$. Then $\mathsf{A_F}$ computes the product $R = \prod_{i \in X^* - \{i^*\}} \sigma^*[i] \bmod N$ and computes the forged signature $\bar{\sigma}^* = \sigma^*/R \bmod N$. Finally, $\mathsf{A_F}$ outputs the forgery pair $(\bar{\mathbf{M}}^*[i^*], \bar{\sigma}^*)$ for scheme $\mathsf{FDH}-\mathsf{MER}$.

This completes the description of $\mathsf{A_F}$. The claimed RPs of $\mathsf{A_F}$ are easily verified from this description. To complete the proof of the lemma, it remains to evaluate the success (in the CMA sense) probability of $\mathsf{A_F}$.

We use the notation $\Pr[.]_{exp}$ to denote probability in experiment exp. The attack experiments for CES scheme $\mathsf{MERP_1}$($\mathrm{WO}-\mathrm{UF}$ notion), and $\mathsf{FDH}-\mathsf{MER}$ (CMA notion) are denoted by subscripts c and d, respectively.

We define the following events. Event $\mathsf{Succ_C}$ means that $\mathsf{A_C}$ succeeds in the sense of $\mathrm{WO}-\mathrm{UF}$. Recall that $\mathsf{Succ_C}$ means that $(\bar{\mathbf{M}}^*, \bar{\sigma}^*)$ is a valid pair for $\mathsf{RSAP_1}$ (call this event $\mathsf{Valid_C}$), and also there exists $i^* \in X^*$ such that $\mathbf{M}^*[i^*] \neq \mathbf{M}_j[i^*]$ for all $j \in [q_s]$ (call this event $\mathsf{NewM_C}$). Event $\mathsf{Succ_F}$ means that $\mathsf{A_F}$ succeeds in the sense of CMA. Recall that $\mathsf{Succ_F}$ means that $(\bar{\mathbf{M}}^*, \bar{\sigma}^*)$ is a valid pair for $\mathsf{FDH}-\mathsf{RSA}$ (call this event $\mathsf{Valid_F}$), and also the message $\bar{\mathbf{M}}^*$ was never queried to S (call this event $\mathsf{NewM_F}$).

We show below that if $\mathsf{Succ_C}$ occurs then $\mathsf{Succ_F}$ occurs. Since $\mathsf{A_F}$ simulates the view of $\mathsf{A_C}$ in exp. d exactly as in the real attack exp. c, it follows that the event $\mathsf{Succ_C}$ has the same probability in both experiments. Therefore $\mathbf{Succ}^{\mathrm{CMA}}_{A_F, FDH-RSA}(k) \geq \mathbf{Succ}^{\mathrm{CES-UF}}_{A_C, RSAP}(k)$, which implies (37), as required.

To complete the proof, it remains to show that $\mathsf{Succ_C}$ implies $\mathsf{Succ_F}$. First $\mathsf{Succ_C}$ implies $\mathsf{Valid_C}$, meaning that $\sigma^* = \prod_{i \in X^*} H(\bar{\mathbf{M}}^*[i])^{1/\mathbf{e}[i]} \bmod N$ so $\bar{\sigma}^* = \sigma^*/R \bmod N = H(\bar{\mathbf{M}}^*[i^*])^{1/\mathbf{e}[i^*]} \bmod N$ is a valid $\mathsf{FDH}-\mathsf{MER}$ signature on message $\bar{\mathbf{M}}^*[i^*]$ and $\mathsf{Valid_F}$ occurs. To show that $\mathsf{NewM_F}$ also occurs, note that if $\mathsf{NewM_C}$ occurs then $\mathbf{M}^*[i^*] \neq \mathbf{M}^*_j[i^*]$ for all $j \in [q_s]$, which implies that $\bar{\mathbf{M}}^*[i^*] \neq$

$\bar{\mathbf{M}}_j[i^*]$ for all $j \in [q_s]$, and all other queries of $\mathsf{A_F}$ to S (namely $\bar{\mathbf{M}}_j[i]$ and $\bar{\mathbf{M}}^*[i]$ for $i \neq i^*$ and $j \in [q_s]$) differ from $\bar{\mathbf{M}}^*[i^*]$ in the submessage index prefix. So if $\mathsf{Succ_C}$ occurs the forgery message was never queried to S, i.e. $\mathsf{NewM_F}$ occurs, as required. This completes the proof of claim (1).

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

Using the random-oracle model for $H(.)$ we immediately obtain a stronger result, namely a reduction with respect to the One-Wayness of RSA, by combining the insecurity bound of Theorem 5.4 for MERP in terms of $\mathsf{FDH-MER}$ with that of Lemma 3.2 in § 3.3.3 for $\mathsf{FDH-MER}$ in terms of RSA_i for $i \in [n_B]$.

Corollary 5.2. *If the RSA function is One-Way for all n_B first odd primes as public exponents (i.e. RSA_i satisfies* OW *notion for all $i \in [n_B]$) then CES scheme MERP has the CES-Unforgeability property (*CES − UF *notion) in the random-oracle model. Concretely, the following bound holds:*

$$\mathbf{InSec}^{\mathrm{CES-UF}}_{\mathsf{MERP}}(t, q_s, n_B, \ell, q_H) \leq \exp(1) \cdot (q_s+1) \cdot n_B \cdot \left(\sum_{i\in[n_B]} \mathbf{InSec}^{\mathrm{OW}}_{\mathsf{RSA}_i}(t')\right) + \frac{q_s(q_s-1)}{2^{l_{tag}+1}} \quad (42)$$

where $t' = t + O([n_B(q_s+1)+q_H]^2 \cdot (\ell + l_{CEAS} + l_{tag} + \log n_B + 1) + [1 + n_B(q_s+1)(n_B(q_s+1)+q_H)]k^3)$.

5.3 Performance Summary

A summary of the performance parameters of the proposed CES schemes is given in Tables 1 to 4. 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_H (Hashing time), T_M^{RSA} (time to perform multiplication modulo N). For all these algorithm running time functions, the argument in brackets denotes the input length to the algorithm. We refer to the following lengths in the tables: l_{S} (signature length), l_H (hash length), l_C (commitment length), l_r (commitment randomness length), l_m (submessage length- assumed identical for all submessages). We also define the function $f(n,m) \stackrel{\text{def}}{=} \min(m \log_2(n/m), n-m)$.

Scheme	Sign Time	Saving Ratio	Typ. Saving ($n = 100$)
CV	$T_{\mathsf{S}}(n \cdot l_C) + n \cdot T_C(l_m)$	$n/\left\lceil 1 + \frac{2n \cdot T_C(l_m)}{T_{\mathsf{S}}(l_m)} \right\rceil$	91
HT	$T_{\mathsf{S}}(l_H) + 3nT_C(l_m)$	$n/\left\lceil 1 + \frac{3n \cdot T_C(l_m)}{T_{\mathsf{S}}(l_m)} \right\rceil$	88
RSAP	$n \cdot T_{\mathsf{S}}^{RSA}$	1	1
MERP	$O(n \log_2^2 n) \cdot T_M^{RSA} + T_{\mathsf{S}}^{RSA}$	$n/\left\lceil 1 + \frac{3n \log_2^2 n \cdot T_M^{RSA}}{T_{\mathsf{S}}^{RSA}} \right\rceil$	17
Trivial	$n \cdot T_{\mathsf{S}}(l_m)$	1	1

Table 1: Comparison of signer computation time for proposed CES schemes. Column 'Time Saving Ratio' gives estimated ratio of sign time of the trivial multiple signature scheme denoted Trivial to the sign time of each scheme. Column 'Typ. Saving ($n = 100$)' evaluates this saving ratio for a typical example with $n = 100$ submessages (see text).

The 'Typ. Saving' figures for the saving ratios were estimated for the following practical example. Note that all the typical timing parameters assumed below were taken from Wei Dai's Benchmarks

Scheme	Verify Time	Saving Ratio	Typ. Saving ($n = 100$)
CV	$T_{\mathsf{V}}(n \cdot l_C) + m \cdot T_C(l_m)$	$m/\left[1 + \frac{(n+m)\cdot T_C(l_m)}{T_{\mathsf{V}}(l_m)}\right]$	91
HT	$T_{\mathsf{V}}(l_H) + (2n+m) \cdot T_C(l_m)$	$m/\left[1 + \frac{(2n+m)\cdot T_C(l_m)}{T_{\mathsf{V}}(l_m)}\right]$	88
RSAP	$T_{\mathsf{V}}^{RSA} + m \cdot T_M^{RSA}$	$m/\left[1 + \frac{m\cdot T_M^{RSA}}{T_{\mathsf{V}}^{RSA}}\right]$	12
MERP	$O(m \log_2 m \log_2 n) \cdot T_M^{RSA}$	$T_{\mathsf{V}}^{RSA}/\left[3 \log_2 m \log_2 n \cdot T_M^{RSA}\right]$	0.1
Trivial	$m \cdot T_{\mathsf{V}}(l_m)$	1	1

Table 2: Comparison of verifier computation time for proposed CES schemes. Column 'Time Saving Ratio' gives estimated ratio of verify time of the trivial multiple signature scheme denoted Trivial to the verify time of each scheme. Column 'Typ. Saving ($n = 100$)' evaluates this saving ratio for a typical example with $n = 100$ and $m = 99$ (see text).

Scheme	Sign Length	Saving Ratio	Typ. Saving ($n = 100$)
CV	$l_{\mathsf{S}} + l_r$	$n/\left[1 + \frac{l_r}{l_{\mathsf{S}}}\right]$	16
HT	$l_{\mathsf{S}} + l_r$	$n/\left[1 + \frac{l_r}{l_{\mathsf{S}}}\right]$	16
RSAP	$n \cdot l_{\mathsf{S}}^{RSA}$	1	1
MERP	l_{S}^{RSA}	n	100
Trivial	$n \cdot l_{\mathsf{S}}^{RSA}$	1	1

Table 3: Comparison of signature length (signer to user communication) for proposed CES schemes. Column 'Saving Ratio' gives estimated ratio of signature length of the trivial multiple signature scheme denoted Trivial to the signature length of each scheme. Column 'Typ. Saving ($n = 100$)' evaluates this saving ratio for a typical example with $n = 100$ (see text).

Scheme	Ext. Sig. Length	Saving Ratio	Typ. Saving ($n = 100$)
CV	$l_{\mathsf{S}} + m \cdot l_r + (n-m) \cdot l_C$	$m/\left[1 + \frac{m\cdot l_r + (n-m)\cdot l_C}{l_{\mathsf{S}}}\right]$	0.19
HT	$l_{\mathsf{S}} + m \cdot l_r + f(n,m) \cdot l_C$	$m/\left[1 + \frac{m\cdot l_r + f(n,m)\cdot l_C}{l_{\mathsf{S}}}\right]$	0.19
RSAP	l_{S}^{RSA}	n	100
MERP	l_{S}^{RSA}	n	100
Trivial	$n \cdot l_{\mathsf{S}}^{RSA}$	1	1

Table 4: Comparison of *extracted* signature length (user to verifier communication) for proposed CES schemes. Column 'Saving Ratio' gives estimated ratio of extracted signature length of the trivial multiple signature scheme denoted Trivial to the extracted signature length of each scheme. Column 'Typ. Saving ($n = 100$)' evaluates this saving ratio for a typical example with $n = 100$ and $m = 99$ (see text).

web page [20] for the 'Crypto++ 4.0' library of cryptographic routines. These timings were obtained on a Celeron 850MHz processor under Windows 2000 SP 1.

We assumed a submessage length $l_m = 1024$ bit.

For the underlying standard signature scheme used by CV and HT, we assumed the Digital Signature Standard DSS [30] with 160-bit exponents and 1024-bit modulus, giving a signature length $l_{\mathsf{S}} = 320$ bit, signing time $T_{\mathsf{S}}(l) = T_H(l) + 5.7$ms, and verifying time $T_{\mathsf{V}}(l) = T_H(l) + 6.4$ms. For the commitment scheme we assumed the scheme of [26], using SHA-1 [31] as the collision-resistant hash function ($k = 160$ bit), and an affine universal hash family $h_{A,b} : \{0,1\}^L \to \{0,1\}^k$ 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 [26] 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. We assumed that the running time to compute the commitment $T_C(l_m)$ is equal to the running time $T_H(l_m) = (l_m/384)\text{Mbs}^{-1}$ of the collision-resistant hash function SHA-1 (since it would normally dominate the time to compute the universal hashing), and the hash length is of course $l_H = 160$. We also assumed that a pseudorandom bit generator is used to generate the n randomness strings for the commitments, so only the pseudorandom generator seed of length l_r need be sent from the signer to the user, as explained in the remark following Theorem 5.2.

For the RSA-based schemes RSAP and MERP, we assumed a modulus length $l_{\mathsf{S}}^{RSA} = 1024$ bit, and hence a signing time $T_{\mathsf{S}}^{RSA} = 10.2$ ms and a verifying time $T_{\mathsf{V}}^{RSA} = 0.32$ ms using a small public-exponent of 17. The time for a multiplication modulo N was taken as $T_M^{RSA} = 24\mu$s.

Note that we did not include the lengths of CEAS and tag in the length comparisons since they have little effect on the result (and are also somewhat application dependant).

The results in Tables 1 to 4 clearly illustrate 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.

6 Implementation of CES Schemes Within the XML Signature Framework

The XML-Signature Recommendation (XMLsig) [3] appears likely to be widely adopted as a standard framework for the exchange of signed digital objects over the Internet. Therefore, for the practical adoption of CES schemes, it is important to examine how CES schemes may be implemented within the XMLsig framework and in particular into Core Generation and Core Validation behaviour [3, §3.0]. In this section, we summarize recent work we have done in this direction [15, 16]. Our work shows, on the negative side, that the built-in functionality of XMLsig alone is not sufficient for implementing secure CES schemes, but on the positive side, that XMLsig is sufficiently flexible to be augmented with external custom transforms which implement our secure CES schemes into XMLsig Core Generation and Core Validation behaviour so that it is application independent.

6.1 The Necessity of Extending XMLSig to Support CES Schemes

The XMLsig specification defines a scheme for creating digital signatures that can be applied to digital content, or data objects, which may be within the same XML document or externally in other documents located on different sites across the web. The XMLsig enables the signer to sign anything that can be referenced by a URI along with any transforms of the information so that some other user can see what is signed and verify the signature.

The thrust of the XMLsig is to enable a signer to efficiently sign multiple objects that may be located on physically different servers and to enable a verifier to verify the available objects even though the entire set of signed objects may not be available. In particular, the XMLsig specification includes a *manifest* item that could be used to indicate whether it is "ok" for a fragment to be missing, although this is left for application logic to perform [3, §5.1].

We observe that the above built-in functionality of XMLsig comes close to implementing our first proposed CES scheme CommitVector. Unfortunately, it is not sufficient, for two main reasons that we now explain.

First, the built-in XMLsig functionality is not designed to provide our *privacy* security against leakage of information on blinded (missing) submessages, and indeed it is *completely insecure* in this sense — Given a standard XML signature on a missing data object having only two possible values, it is easy to recover from the signature the value of the missing object (the reason is that standard XML signatures use a deterministic hash function in place of our commitment scheme).

Second, the built-in XMLsig functionality does not allow the signer to specify flexible extraction policies (CEAS), which allow, for instance, certain objects to be removed only if other specified objects are also removed. Such flexible policies may be crucial in many real-world applications. Whilst the XMLsig's *manifest* element may be considered to handle the signer's extraction policy, it relies on the application to perform the checking. For CES functionality we require extraction policy verification to be included in the XMLsig Core Validation behaviour so that we can achieve application independence.

We conclude that the built-in XMLsig functionality does not suffice for implementing secure and useful CES schemes. Fortunately, in the following section we show that the XMLsig specification is sufficiently flexible to allow the built-in XMLsig functionality to be augmented with external *custom transforms* which can be used to incorporate CES schemes into the XMLsig framework.

6.2 Extending XMLSig to Support CES

Whilst the XMLsig specification defines a scheme for producing digital signatures, it also provides a framework to support variations to a basic digital signature. Some of this flexibility is achieved through the *Reference Processing Model* [3, §4.3.3.2] and its support for the implementation of custom transforms.

We have developed our own custom transforms, along with a corresponding design for the XMLsig structure, to achieve CES functionality using the *CommitVector* scheme in XMLsig Core Generation and Core Validation behaviour for application independence [15]. This implementation included a CEAS that only allowed a simple single-dimensional signer extraction policy. More complex CEAS encodings have been developed to enable support for an extraction policy that allows fragment groupings to be specified and we have shown how to implement into XMLsig Core behaviour [16]. The XMLsig specification in §4.3.3.4 states:

> "However, applications should refrain from using application-specific transforms if they wish their signatures to be verifiable outside of their application domain."

Notwithstanding this, our design and implementation using the XML Signature implementation in the .Net Framework downloads the custom transforms from a Web-server on-demand independent of the application [15]. This demonstrates that the use of custom transforms with XML signatures need not be constrained to a specific application domain, although it is constrained to homogenous platforms at this time.

7 Summary and Open Problems

Motivated by emerging needs in online interactions and the need to embrace a minimal information model through the use of smaller verifiable content granularity, 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 provably secure CES schemes with various performance tradeoffs.

Open Problems. An interesting problem for future research is to find new CES schemes with performance advantages over the schemes presented here, possibly using other cryptographic assumptions. For example, our schemes achieve either high communication savings (RSA schemes) or high computation savings (commitment-based schemes). Is it possible to achieve simultaneously a high computation saving ratio *and* a high communication saving ratio in the worst case over the multiple signature scheme (meaning that both saving factors in the order of the number of submessages n)? Is it possible to achieve high communication saving ratio in the worst case using more general cryptographic primitives than homomorphic trapdoor one-way functions?

Acknowledgement

We wish to thank Dr David McG. Squire and Dr Peter Stanski for their involvement and contribution with the work presented in §6.

A Appendix

The following definition and Lemmas are required for security results stated in the paper.

The following definition is for a weaker notion of unforgeability than the full one defined in the main body of the paper.

Definition A.1 (Weak Ordered CES Unforgeability: $\mathrm{WO}-\mathrm{UF}$ Notion). *A CES scheme* CES *is said to have the* Weak Ordered *(*$\mathrm{WO}-\mathrm{UF}$*) CES-unforgeability notion if the following holds: It is infeasible for an attacker* A*, 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. The attacker's Resource Parameters (RPs) are* (t, q_s, n, ℓ) *as for the*

Scheme CES_2

Algorithm $\mathsf{GK}_2(k)$
$(sk, pk) \leftarrow \mathsf{GK}_1(k)$
Return (sk, pk)

Algorithm $\mathsf{Ext}_2(pk, \mathbf{M}, \sigma_F, X)$
Parse $\sigma_F = (CEAS, T, \bar{\sigma_F})$
for $i \in [n]$
$\bar{\mathbf{M}}[i] \leftarrow (CEAS, T, n, \mathbf{M}[i])$
$\bar{\sigma_E} \leftarrow \mathsf{Ext}_1(pk, \bar{\mathbf{M}}, \bar{\sigma_F}, X)$
$\sigma_E \leftarrow (CEAS, T, \bar{\sigma_E})$
Return σ_E

Algorithm $\mathsf{Sig}_2(sk, \mathbf{M}, CEAS)$
$n \leftarrow \mathsf{len}(\mathbf{M})$
$T \xleftarrow{R} \{0,1\}^{l_{tag}}$
for $i \in [n]$
$\bar{\mathbf{M}}[i] \leftarrow (CEAS, T, n, \mathbf{M}[\mathbf{i}])$
$\bar{\sigma_F} \leftarrow \mathsf{Sig}_1(sk, \bar{\mathbf{M}}, CEAS)$
$\sigma_F \leftarrow (CEAS, T, \bar{\sigma_F})$
Return σ_F

Algorithm $\mathsf{Ver}_2(pk, \mathbf{M}', \sigma_E)$
$X' \leftarrow \mathsf{CI}(\mathbf{M}')$; $n \leftarrow \mathsf{len}(\mathbf{M}')$
Parse $\sigma_E = (CEAS, T, \bar{\sigma_E})$
for $i \in [n]$
$\bar{\mathbf{M}}'[i] \leftarrow (CEAS, T, n, \mathbf{M}'[\mathbf{i}])$
If
$\mathsf{Ver}_1(pk, \bar{\mathbf{M}}', \bar{\sigma_E}) = Acc$ and
$X' \in CEAS$
Return Acc
Else Return Rej.

Figure 10: Definition of transformed CES scheme $\mathsf{CES}_2 = T_{WO}(\mathsf{CES}_1)$.

full $\mathrm{CES}-\mathrm{UF}$ *notion. The attacker's success probability and the scheme's insecurity function are similarly denoted* $\mathbf{Succ}^{\mathrm{WO-UF}}_{\mathsf{A,CES}}(k)$ *and* $\mathbf{InSec}^{\mathrm{WO-UF}}_{\mathsf{CES}}(t, q_s, n, \ell)$*, respectively.*

We now give a generic security-enhancing transformation $T_{WO}(.)$ to upgrade any CES scheme CES_1 which is only weakly unforgeable (namely $\mathrm{WO}-\mathrm{UF}$-secure) into another CES scheme $\mathsf{CES}_2 = T_{WO}(\mathsf{CES}_1)$ which is strongly unforgeable (namely $\mathrm{CES}-\mathrm{UF}$-secure). The construction is straightforward and involves choosing a unique 'Document tag' for each signed document and appending a CEAS encoding, the tag, and the document length to each submessage before applying the original CES Sign algorithm. The CEAS and tag are appended in the signature and the verifying algorithm also enforces the CEAS restriction. A precise specification of the transform T_{WO} follows.

Definition A.2 (Weak-to-Full UF Transform $T_{WO}(.)$). *The transform $T_{WO}(.)$ transforms any input CES scheme $\mathsf{CES}_1 = (\mathsf{GK}_1, \mathsf{Sig}_1, \mathsf{Ext}_1, \mathsf{Ver}_1)$ into a new CES scheme $\mathsf{CES}_2 = T_{WO}(\mathsf{CES}_1) = (\mathsf{GK}_2, \mathsf{Sig}_2, \mathsf{Ext}_2, \mathsf{Ver}_2)$ defined in Fig. 10.*

The following lemma proves that the transform T_{WO} achieves its goal if the tag length l_{tag} is sufficiently long to avoid tag collisions with high probability. We denote by l_{CEAS} the bit-length bound for the CEAS encoding, and assume that the length field n appended by CES_2 is $\log(n)$-bits long.

Lemma A.1. *The transform T_{WO} converts any CES scheme CES_1 which has the Weak-Ordered Unforgeability property ($\mathrm{WO}-\mathrm{UF}$ notion) into a CES scheme $\mathsf{CES}_2 = T_{WO}(\mathsf{CES}_1)$ which has the*

full CES- Unforgeability property ($\mathrm{CES}-\mathrm{UF}$ *notion). Concretely:*

$$\mathbf{InSec}^{\mathrm{CES}-\mathrm{UF}}_{\mathsf{CES}_2}(t, q_s, n, \ell) \leq \mathbf{InSec}^{\mathrm{WO}-\mathrm{UF}}_{\mathsf{CES}_1}(t', q'_s, n'.\ell') + \frac{q_s(q_s-1)}{2^{l_{tag}+1}} \tag{43}$$

where $t' = t$; $q'_s = q_s$; $n' = n$; $\ell' = \ell + l_{CEAS} + l_{tag} + \log(n)$.

Proof. We show how to use an efficient attacker A_2 breaking scheme CES_2 in the sense of $\mathrm{CES}-\mathrm{UF}$ to construct an efficient attacker A_1 for breaking scheme CES_1 in the sense of $\mathrm{WO}-\mathrm{UF}$. By lower bounding the success probability of A_1 in terms of that of A_2, we will derive the desired bound (43).

Attacker A_1 works as follows. A_1 is invoked by the $\mathrm{WO}-\mathrm{UF}$ experiment on input (k, pk), where k is the security parameter and pk is a public key for CES_1, with sk denoting the corresponding secret key. A_1 is able to query CEAS/document pairs to be signed by the signing oracle $\mathsf{Sig}_2(sk_s, ., .)$ for scheme CES_1.

Setup. A_1 runs A_2 on input (k, pk).

Sig **Queries.** When A_2 makes the jth CES sign query $(CEAS_j, \mathbf{M}_j)$ to its Sig_2 oracle, A_1 answers it by simulating the action of the Sig_1 algorithm, using the $\mathsf{Sig}_1(sk, ., .)$ oracle. The details follow. Let $n_j = \mathsf{len}(\mathbf{M}_j)$. First, A_1 chooses a random document tag $T_j \xleftarrow{\mathrm{R}} \{0,1\}^{l_{tag}}$. It defines document $\bar{\mathbf{M}}$ by setting $\bar{\mathbf{M}}_j[i] = (CEAS_j, T_j, n_j, \mathbf{M}_j[i])$ for all $i \in [n_j]$. It queries $(CEAS_j, \bar{\mathbf{M}}_j)$ to Sig_2 to get $\bar{\sigma}_j$. Finally it sets the simulated signature to $\sigma_j = (CEAS_j, T_j, \bar{\sigma}_j)$.

Output. Eventually A_2 terminates and outputs a forgery pair $(\mathbf{M}^*, \sigma^*)$. Let $n^* = \mathsf{len}(\mathbf{M}^*)$ and $X^* = \mathsf{CI}(\mathbf{M}^*)$. A_1 parses $\sigma^* = (CEAS^*, T^*, \bar{\sigma}^*)$, and defines document $\bar{\mathbf{M}}^*$ by setting $\bar{\mathbf{M}}^*[i] = (CEAS^*, T^*, n^*, \mathbf{M}^*[i])$ for all $i \in [n^*]$. Finally, it outputs the forgery pair $(\bar{\mathbf{M}}^*, \bar{\sigma}^*)$ for scheme CES_1.

This completes the description of A_1. The claimed RPs of A_1 are easily verified from this description. To complete the proof of the lemma, it remains to evaluate the success (in the $\mathrm{WO}-\mathrm{UF}$ sense) probability of A_1.

We use the notation $\Pr[.]_{exp}$ to denote probability in experiment *exp*. The attack experiments for CES scheme CES_1($\mathrm{WO}-\mathrm{UF}$ notion), and CES_2 ($\mathrm{CES}-\mathrm{UF}$ notion) are denoted by subscripts 1 and 2, respectively.

We define the following events. Event Succ_1 means that A_1 succeeds in the sense of $\mathrm{WO}-\mathrm{UF}$. Recall that Succ_1 means that $(\bar{\mathbf{M}}^*, \bar{\sigma}^*)$ is a valid pair for CES_1 (call this event Valid_1), and also there exists $i^* \in \mathsf{CI}(\bar{\mathbf{M}}^*)$ such that $\bar{\mathbf{M}}^*[i^*] \neq \bar{\mathbf{M}}_j[i^*]$ for all $j \in [q_s]$ (call this event NewM_1). Event Succ_2 means that A_2 succeeds in the sense of $\mathrm{CES}-\mathrm{UF}$. Recall that Succ_2 means that $(\mathbf{M}^*, \sigma^*)$ is a valid pair for CES_2 (call this event Valid_2), and also for each $j \in [q_s]$ either $X^* \notin CEAS_j$ or $\mathbf{M}^* \not\leq \mathbf{M}_j$ (call this event NewM_2). Denote by DistT the event that all q_s tags T_j for $j \in [q_s]$ are distinct.

We will show below that if $\mathsf{Succ}_2 \wedge \mathsf{DistT}$ occurs then Succ_1 occurs. Since A_1 simulates the view of A_2 in exp. 1 exactly as in the real attack exp. 2, it follows that the event $\mathsf{Succ}_2 \wedge \mathsf{DistT}$ has the

same probability in both experiments. Therefore:

$$\begin{aligned}
\mathbf{Succ}^{\mathrm{WO-UF}}_{A_1,CES_1}(k) &\geq \Pr[\mathsf{Succ}_2 \wedge \mathsf{DistT}]_1 \\
&= \Pr[\mathsf{Succ}_2 \wedge \mathsf{DistT}]_2 \\
&\geq \Pr[\mathsf{Succ}_2]_2 - \Pr[\neg\mathsf{DistT}]_2 \\
&\geq \mathbf{Succ}^{\mathrm{CES-UF}}_{A_2,CES_2}(k) - \frac{q_s(q_s-1)}{2^{l_{tag}+1}}, \qquad (44)
\end{aligned}$$

where we have used the birthday bound Lemma A.2 to get $\Pr[\neg\mathsf{DistT}] \leq \frac{q_s(q_s-1)}{2^{l_{tag}+1}}$. Rearranging (44) and taking maximums we immediately obtain the desired insecurity bound (43).

To complete the proof, it remains to show that if $\mathsf{Succ}_2 \wedge \mathsf{DistT}$ occurs then Succ_1 occurs. First $\mathsf{Succ}_2 \wedge \mathsf{DistT}$ implies Valid_2, which directly implies Valid_1 by definition of CES_2. Now it remains to show $\mathsf{Succ}_2 \wedge \mathsf{DistT}$ implies NewM_1. If $\mathsf{Succ}_2 \wedge \mathsf{DistT}$ occurs there are two possible subcases.

In the first subcase, $T^* \notin \{T_1, \ldots, T_{q_s}\}$, which immediately implies NewM_1 since $\bar{\mathbf{M}}^*[i]$ and $\bar{\mathbf{M}}_j[i]$ have unequal tags T^* and T_j respectively, for any $i \in X^*$.

The second subcase is that $T^* = T_{j^*}$ for some unique $j^* \in [q_s]$. In this case, because j^* is unique we still have $\bar{\mathbf{M}}^*[i] \neq \bar{\mathbf{M}}_j[i]$ for $j \neq j^*$ and any $i \in X^*$. So we just need to show that there exists $i^* \in X^*$ such that $\bar{\mathbf{M}}^*[i^*] \neq \bar{\mathbf{M}}_{j^*}[i^*]$. But Succ_2 implies that either $X^* \notin CEAS_{j^*}$ and $X^* \in CEAS^*$ (which implies $CEAS_{j^*} \neq CEAS^*$ and hence i^* exists because $\bar{\mathbf{M}}^*[i^*]$ and $\bar{\mathbf{M}}_{j^*}[i^*]$ contain different CEAS tags for any $i^* \in X^*$) or that $\mathbf{M}^* \not\leq \mathbf{M}_{j^*}$, in which case there are two further subcases. The first is $n^* \neq n_{j^*}$ in which case i^* exists because $\bar{\mathbf{M}}^*[i^*]$ and $\bar{\mathbf{M}}_{j^*}[i^*]$ contain different length fields for any $i^* \in X^*$. The second directly states the existence of i^*. This completes the proof that $\mathsf{Succ}_2 \wedge \mathsf{DistT}$ implies Succ_1, which completes the proof of the lemma. □

We use the following result in the proof of Lemma A.1.

Lemma A.2. *[Birthday Bound] Let $(T_1, \ldots, T_q)$ denote a vector of q independent random strings, each uniformly distributed in $\{0,1\}^l$. The probability of a collision between some pair of strings (i.e. $T_i = T_j$ for some $i \neq j$) is at most $\frac{q(q-1)}{2^{l+1}}$.*

Proof. Let $p(q)$ denote the probability of a collision among $(T_1, \ldots, T_q)$. We use induction on q to show $p(q) \leq f(q) = \frac{q(q-1)}{2^{l+1}}$. The basis case $q = 1$ is trivial since $p(1) = 0$. For the induction step, we suppose that $p(q) \leq f(q) = \frac{q(q-1)}{2^{l+1}}$ for some q and we show that $p(q+1) \leq f(q+1) = \frac{q(q+1)}{2^{l+1}}$. If a collision occurs among $(T_1, \ldots, T_{q+1})$, then either a collision occurs among $(T_1, \ldots, T_q)$ (this has probability at most $f(q)$ by the induction hypothesis) or $T_{q+1} = T_i$ for some $i \in \{1, \ldots, q\}$ (this has probability at most $q/2^l$ since T_{q+1} is uniformly distributed in $\{0,1\}^l$ independent of $(T_1, \ldots, T_q)$). Hence $p(q+1) \leq f(q) + q/2^l = \frac{q(q-1)+2q}{2^{l+1}} = f(q+1)$, as required. This completes the proof. □

References

[1] J. An, Y. Dodis, and T. Rabin. On the Security of Joint Signature and Encryption. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 83–107, Berlin, 2002. Springer-Verlag.

[2] S.A. Baker. Don't worry be happy. Available online, June 1994. [Last accessed: July 27, 2002]. URI: http://www.wired.com/wired/archive/2.06/nsa.clipper_pr.html.

[3] 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. Available online. [Last accessed: September 18, 2002]. URI: http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/.

[4] M. Bellare. Practice-Oriented Provable-Security. In *Lectures on Data Security*, volume 1561 of *LNCS*, pages 1–15, Berlin, 1998. Springer-Verlag.

[5] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 26–45, Berlin, 1998. Springer-Verlag.

[6] M. Bellare, J.A. Garay, and T. Rabin. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In *EUROCRYPT '98*, volume 1403 of *LNCS*, pages 236–250, Berlin, 1998. Springer-Verlag.

[7] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental Cryptography: The Case of Hashing and Signing. In *CRYPTO '94*, volume 0839 of *LNCS*, pages 216–233, Berlin, 1994. Springer-Verlag.

[8] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental Cryptography and Application to Virus Protection. In *Proc. 27-th ACM Symp. on Theory Of Comput.*, pages 45–56, New York, 1995. ACM Press.

[9] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1-st ACM Conf. on Comp. and Comm. Security*, pages 62–73, New York, November 1993. ACM.

[10] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416, Berlin, 1996. Springer-Verlag.

[11] 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.

[12] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Proceedings of Eurocrypt 2003*, Warsaw, Poland, 4–8 May 2003. Springer-Verlag. (to appear).

[13] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, 2000.

[14] S. Brands. A technical overview of digital credentials. Available online, February 20 2002. [Last accessed: February 18, 2003]. URI: http://www.credentica.com/technology/overview.pdf.

[15] L. Bull, P. Stanski, and D. 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)*, Budapest, Hungary, 20–24 May 2003. (to appear).

[16] Laurence Bull, David McG. Squire, Jan Newmarch, and Yuliang Zheng. Grouping verifiable content for selective disclosure using XML signatures. Technical report, School of Computer Science and Software Engineering, Monash University, 900 Dandenong Road, Caulfield East, Victoria 3145 Australia, April 2003.

[17] G.C. Chick and S.E. Tavares. Flexible Access Control with Master Keys. In *CRYPTO '89*, volume 0435 of *LNCS*, pages 316–323, Berlin, 1990. Springer-Verlag.

[18] J-S. Coron. On the Exact Security of Full Domain Hash. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235, Berlin, 2000. Springer-Verlag.

[19] J.S. Coron and D. Naccache. On the Security of RSA Screening. In *PKC '99*, volume 1560 of *LNCS*, pages 197–203, Berlin, 1999. Springer-Verlag.

[20] W. Dai. Crypto++ 4.0 Benchmarks, 2000. http://www.eskimo.com/ weidai/benchmarks.html.

[21] A. Fiat. Batch RSA. *J. of Cryptology*, 10:75–88, 1997.

[22] R. Gennaro, S. Halevi, and T. Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. In *EUROCRYPT '99*, volume 1592 of *LNCS*, pages 123–139, Berlin, 1999. Springer-Verlag.

[23] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4):792–807, 1986.

[24] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. of Computer and System Sciences*, 28(2):270–299, 1984.

[25] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptively Chosen Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[26] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *CRYPTO '96*, volume 1109 of *LNCS*, pages 201–215, Berlin, 1996. Springer-Verlag.

[27] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic Signature Schemes. In *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262, Berlin, 2002. Springer-Verlag.

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

[29] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. Discrete mathematics and its applications. CRC Press, 1997.

[30] National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186*, November 1994.

[31] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-1*, April 1995.

[32] C.J. Pavlovski and C. Boyd. Efficient Batch Signature Generation Using Tree Structures. In *CrypTEC '99*, pages 70–77. City University of Hong Kong Press, 1999.

[33] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–128, 1978.

[34] J.B. Rosser and L. Schoenfeld. Approximate Formulas for Some Functions of Prime Numbers. *Illinois. J. Math.*, 6:64–94, 1962.

[35] H. Shacham and D. Boneh. Improving SSL Handshake Performance via Batching. In *CT-RSA 2001*, volume 2020 of *LNCS*, pages 28–43, Berlin, 2001. Springer-Verlag.

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

[37] R. Steinfeld, L. Bull, and Y. Zheng. Content Extraction Signatures. In *International Conference on Information Security and Cryptology ICISC 2001*, volume 2288 of *LNCS*, pages 285–304, Berlin, 2001. Springer-Verlag.