J Cryptol (2015) 28:351–395
DOI: 10.1007/s00145-014-9182-0

Journal of
CRYPTOLOGY

# Computing on Authenticated Data

Jae Hyun Ahn · Susan Hohenberger*
Johns Hopkins University, Baltimore, MD, USA
indrazit@gmail.com; susan@cs.jhu.edu

Dan Boneh†
Stanford University, Stanford, CA, USA
dabo@cs.stanford.edu

Jan Camenisch‡
IBM Research – Zurich, Ruschlikon, Switzerland
jca@zurich.ibm.com

Abhi Shelat§
University of Virginia, Charlottesville, VA, USA
abhi@cs.virginia.edu

Brent Waters¶
University of Texas at Austin, Austin, TX, USA
bwaters@cs.utexas.edu

Communicated by Tal Rabin.

Received 13 June 2012
Online publication 18 April 2014

**Abstract.** In tandem with recent progress on computing on encrypted data via fully homomorphic encryption, we present a framework for computing on *authenticated* data

* Supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-11-2-0211, the Office of Naval Research under contract N00014-11-1-0470, NSF CNS 1154035 and 1228648, a Microsoft Faculty Fellowship and a Google Faculty Research Award.

† Supported by NSF, DARPA, and AFOSR. Applying to all authors, the views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US government.

‡ This work has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement No. 216483 (PrimeLife).

§ Supported by NSF CNS-0845811 and TC-1018543, Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-11-2-0211, and a Microsoft New Faculty Fellowship.

¶ Supported by NSF CNS-0915361, CNS-0952692 and CNS-1228599, AFOSR Grant No. FA9550-08-1-0352, DARPA PROCEED, DARPA N11AP20006, Google Faculty Research Award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

via the notion of slightly homomorphic signatures, or $P$-homomorphic signatures. With such signatures, it is possible for a third party to *derive* a signature on the object $m'$ from a signature of $m$ as long as $P(m, m') = 1$ for some predicate $P$ which captures the "authenticatable relationship" between $m'$ and $m$. Moreover, a derived signature on $m'$ reveals *no extra information* about the parent $m$. Our definition is carefully formulated to provide one unified framework for a variety of distinct concepts in this area, including arithmetic, homomorphic, quotable, redactable, transitive signatures, and more. It includes being unable to distinguish a derived signature from a fresh one *even when given the original signature*. The inability to link derived signatures to their original sources prevents some practical privacy and linking attacks, which is a challenge not satisfied by most prior works. Under this strong definition, we then provide generic constructions for all univariate and closed predicates, and specific efficient constructions for a broad class of natural predicates such as quoting, subsets, weighted sums, averages, and Fourier transforms. To our knowledge, these are the first efficient constructions for these predicates (excluding subsets) that provably satisfy this strong security notion.

**Keywords.** Authentication, Homomorphic signatures, Quotable signatures.

## 1. Introduction

In tandem with recent progress on computing *any function* on encrypted data, e.g., [33, 56, 59], this work explores computing on unencrypted signed data. In the recent years, several independent lines of research touched on this area:

- Quoting/redacting: [1, 20–22, 36, 39, 46, 58] Given Alice's signature on some message $m$ anyone should be able to derive Alice's signature on a subset of $m$. Quoting typically applies to signed text messages where one wants to derive Alice's signature on a substring of $m$. Quoting can also apply to signed images where one wants to derive a signature on a subregion of the image (say, a face or an object) and to data structures where one wants to derive a signature of a subset of the data structure such as a sub-tree of a tree.
- Arithmetic: [15–17, 26, 32, 40, 63, 65] Given Alice's signature on vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{F}_p^n$ anyone should be able to derive Alice's signature on a vector $\mathbf{v}$ in the linear span of $\mathbf{v}_1, \ldots, \mathbf{v}_k$. Arithmetic on signed data is motivated by applications to secure network coding [31]. We show that these schemes can be used to compute authenticated linear operations such as computing an authenticated weighted sum of signed data and an authenticated Fourier transform. As a practical consequence of this, we show that an untrusted database storing signed data (e.g., employee salaries) can publish an authenticated average of the data without leaking any other information about the stored data. Recent constructions go beyond linear operations and support low degree polynomial computations [15].
- Transitivity: [8, 9, 37, 45, 50, 51, 54, 64] Given Alice's signature on edges in a graph $G$ anyone should be able to derive Alice's signature on a pair of vertices $(u, v)$ if and only if there is a path in $G$ from $u$ to $v$. The derived signature on the pair $(u, v)$ must be indistinguishable from a fresh signature on $(u, v)$ had Alice generated one herself [45]. This requirement ensures that the derived signature on $(u, v)$ reveals no information about the path from $u$ to $v$ used to derive the signature.

In this paper, we put forth a general framework for computing on authenticated data that encompasses these lines of research and much more. While prior definitions mostly contained artifacts specific to the type of malleability they supported, and thus, were hard to compare to one another, we generalize and strengthen these disparate notions into a single definition. This definition can be instantiated with any predicate, and we allow repeated computation on the signatures (e.g., it is possible to quote from a quoted signature). During our study, we realized that the "privacy" notions offered by many existing definitions are, in our view, insufficient for some practical applications. We therefore require a stronger (and seemingly a significantly more challenging to achieve) property called *context hiding*. Under this definition, we provide two generic solutions for computing signatures on any univariate, closed predicate; however, these generic constructions are not efficient. We also present efficient constructions for three problems: quoting substrings in Sect. 4, a subset predicate in Sect. 5, and a weighted average over data in Sect. 6 (which captures weighted sums and Fourier transforms). Our quoting substring construction is novel and significantly more efficient than the generic solutions. For the problems of subsets and weighted averages, we show somewhat surprising connections to respective existing solutions in attribute-based encryption and network coding signatures.

After the appearance of the conference version of this work, this concept was explored further. Attrapadung, Libert, and Peters [3] studied an even stronger notion of privacy, which they called *complete context-hiding*, and later [4] provided efficient quotable and linearly homomorphic signatures satisfying this strong notion. Chase, Kohlweiss, Lysyanskaya, and Meiklejohn [27] extended the context-hiding definition of Attrapadung et al. to allow for adversarially generated keys and signatures and studied applications to anonymous credentials. Recently, Deiseroth, Fehr, Fischlin, Maasz, Reimers, and Stein [29] explored computing on authenticated data when the predicates may be adjusted or combined.

### 1.1. *Overview*

*A general framework* Let $\mathcal{M}$ be some message space and let $2^{\mathcal{M}}$ be its powerset. Consider a predicate $P : 2^{\mathcal{M}} \times \mathcal{M} \rightarrow \{0, 1\}$ mapping a set of messages and a message to a bit. Loosely speaking we say that a signature scheme supports computations with respect to $P$ if the following holds:

> Let $M \subset \mathcal{M}$ be a set of messages and let $m'$ be a *derived* message, namely $m'$ satisfies $P(M, m') = 1$. Then there exists an efficient procedure that can derive Alice's signature on $m'$ from Alice's independent signatures on all of the messages in $M$.

For the quoting application, the predicate $P$ is defined as $P(M, m') = 1$ iff $m'$ is a quote from the set of messages $M$. Here, we focus on quoting from a single message $m$ so that $P$ is false whenever $M$ contains more than one component[1], and thus use the notation $P(m, m')$ as shorthand for $P(\{m\}, m')$. The predicate $P$ for arithmetic computations is defined in Appendix 2.3 and essentially says that $P\big((\mathbf{v}_1, \ldots, \mathbf{v}_k), \mathbf{v}\big)$ is true whenever $\mathbf{v}$ is in the span of $\mathbf{v}_1, \ldots, \mathbf{v}_k$.

[1] We leave it for future work to construct systems for securely quoting from two messages (or possibly more) as defined next.

We emphasize that signature derivation can be iterative. For example, given a message-signature pair $(m, \sigma)$ from Alice, Bob can publish a derived message-signature pair $(m', \sigma')$ for an $m'$ where $P(m, m')$ holds. Charlie, using $(m', \sigma')$, may further derive a signature $\sigma''$ on $m''$. In the quoting application, Charlie is quoting from a quote which is perfectly fine.

*Security* We give a clean security definition that captures two properties: unforgeability and context hiding. We briefly discuss each in turn and give precise definitions in the next section.

- Unforgeability captures the idea that an attacker may be given various derived signatures (perhaps iteratively derived) on messages of his choice. The attacker should be unable to produce a signature on a message that is not derivable from the set of signed messages in his possession. For example, suppose Alice generates $(m, \sigma)$ and gives it to Bob who then publishes a derived signature $(m', \sigma')$. Then an attacker given $(m', \sigma')$ should be unable to produce a signature on $m$ or on any other message $m''$ such that $P(m', m'') = 0$.
- Context hiding captures an important privacy property: a signature should reveal nothing more than the message being signed. In particular, if a signature on $m'$ was derived from a signature on $m$, an attacker should not learn anything about $m$ other than what can be inferred from $m'$. This should be true even if the original signature on $m$ is revealed. For example, a signed quote should not reveal anything about the message from which it was quoted, including its length, the position of the quote, whether its parent document is the same as another quote, whether it was derived from a given signed message or generated freshly, etc.

Defining context hiding is an interesting and subtle task. In the next section, we give a definition that captures a very strong privacy requirement. We discuss earlier attempts at defining privacy following our definition in Sect. 2.4; while many prior works use a similar sounding *intuition* as we give above, most contain a fundamental difference to ours in their *formalization*.

We note that notions such as group or ring signatures [7,13,23,28,53] have considered the problem of hiding the identity of a signer among a set of users. Context hiding ensures privacy for the data rather than the signer. Our goal is to hide the legacy of how a signature was created.

*Efficiency* We require that the size of a signature, whether fresh or derived, depend only on the size of the object being signed. This rules out solutions where the signature grows with each derivation.

*Generic Approaches* We begin with two generic constructions that can be inefficient. They apply to *closed, univariate* predicates, namely predicates $P(M, m')$ where $M$ contains a single message ($P$ is false when $|M| > 1$) and where if $P(a, b) = P(b, c) = 1$ then $P(a, c) = 1$. The first construction uses any standard signature scheme $S$ where the signing algorithm is deterministic. (One can enforce determinism using PRFs [34]). To sign a message $m \in \mathcal{M}$, one uses $S$ to sign each message $m'$ such that $P(m, m') = 1$. The signature consists of all these signature components. To verify a signature for $m$, one checks the signature component corresponding to the message $m$. To derive a signature

$m'$ from $m$, one copies the signature components for all $m''$ such that $P(m', m'') = 1$. Soundness of the construction follows from the security of the underlying standard scheme $S$ and context hiding from the fact that signing in $S$ is deterministic.

Unfortunately, these signatures may become large consisting up to $|\mathcal{M}|$ signature components—impacting both the signing time and signature size. Our second generic construction alleviates the space burden by using an RSA accumulator. The construction works in a similar brute force fashion where a signature on $m$ is an accumulator value on all $m'$ such that $P(m, m') = 1$. While this produces short signatures, the time component of both verification and derivation are even worse than the first generic approach. Thus, these generic approaches are too expensive for most interesting predicates. We detail these generic approaches and proofs in Sect. 3, where we also discuss a generic construction using NIZK.

*Our Quoting Construction.* We turn to more efficient constructions. First, we set out to construct a signature for quoting *substrings*[2], which although conceptually simple is non-trivial to realize securely. As an efficiency baseline, we note that the brute force generic construction of the quoting predicate would result in $n^2$ components for a signature on $n$ characters. So any interesting construction must perform more efficiently than this. We prove our construction selectively secure.[3] In addition, we give some potential future directions for achieving adaptive security and removing the use of random oracles.

Our construction uses bilinear groups to link different signature components together securely, but in such a way that the context can be hidden by a re-randomizing step in the derivation algorithm. A signature in our system on a message of length $n$ consists of $n \lg n$ group elements; intuitively organized as $\lg n$ group elements assigned to each character. To derive a new signature on a substring of $\ell$ characters, one roughly removes the group elements not associated with the new substring and then re-randomizes the remaining part of the signature. This results in a new signature of $\ell \lg \ell$ group elements. The technical challenge consists in simultaneously allowing re-randomization and preserving the "linking" between successive characters. In addition, there is a second option in our derive algorithm that allows for the derivation of a short signature of $\lg \ell$ group elements; however, the derive procedure cannot be applied again to this short signature. *Thus, we support quoting from quotes, and also provide a compression option which produces a very short quote, but the price for this is that it cannot be quoted from further.*

*Computing Signatures on Subsets and Weighted Averages* Our final two contributions are schemes for deriving signatures on subsets and weighted averages on signatures. Rather than create entirely new systems, we show connections to existing Attribute-Based Encryption schemes and Network Coding Signatures. Briefly, our subset construction extends the concept of Naor [14] who observed that every IBE scheme can be transformed into a standard signature scheme by applying the IBE KeyGen algorithm as a signing

[2] A substring of $x_1 \dots x_n$ is some $x_i \dots x_j$ where $i, j \in [1, n]$ and $i \leq j$. We emphasize that we are not considering *subsequences*. Thus, it is *not* possible, in this setting, to extract a signature on "I like fish" from one on "I do not like fish".

[3] Following an analog of [24], selective security for signatures requires the attacker to give the forgery message before seeing the verification key.

algorithm. Here we show an analog for known Ciphertext-Policy (CP) ABE schemes. The KeyGen algorithm which generates a key for a set $S$ of attributes can be used as a signing algorithm for the set $S$. For known CP-ABE systems [10,41,62] it is straightforward to derive a key for a subset $S'$ of $S$ and to re-randomize the signature/key. To verify a signature on $S$ we can apply Naor's signature-from-IBE idea and encrypt a random message $X$ to a policy that is an AND of all the attributes in $S$ and see if the signature can be used as an ABE key to decrypt to $X$. Signatures for subsets have been previously considered in [37, §6.4], but without context hiding requirements. We provide further details in Sect. 5. Our construction for weighted sums is presented in Sect. 6, where we discuss how this applies to Fourier transforms.

## 2. Definitions

**Definition 2.1** (*Derived messages*) Let $\mathcal{M}$ be a message space and let $P : 2^{\mathcal{M}} \times \mathcal{M} \to \{0, 1\}$ be a predicate from sets over $\mathcal{M}$ and a message in $\mathcal{M}$ to a bit. We say that a message $m'$ is **derivable** from the set $M \subseteq \mathcal{M}$ if $P(M, m') = 1$. We denote by $P^*(M)$ the set of messages derivable from $M$ by repeated derivation. That is, let $P^0(M)$ be the set of messages derivable from $M$ and for $i > 0$ let $P^i(M)$ be the set of messages derivable from $P^{i-1}(M)$. Then $P^*(M) := \cup_{i=0}^{\infty} P^i(M)$.

We define the closure of $P$, denoted $P^*$, as the predicate defined by $P^*(M, m) = 1$ iff $m \in P^*(M)$.

A $P$-homomorphic signature scheme $\Pi$ for message space $\mathcal{M}$ and predicate $P$ is a triple of PPT algorithms:

**KeyGen**$(1^\lambda)$: the key generation algorithm outputs a key pair $(pk, sk)$. We treat the secret key $sk$ as a signature on the empty tuple $\varepsilon \in \mathcal{M}^*$. We also assume that $pk$ is embedded in $sk$.

**SignDerive**$(pk, (\{\sigma_m\}_{m\in M}, M), m', w)$: the algorithm takes as input the public key, a set of messages $M \subseteq \mathcal{M}$ and corresponding signatures $\{\sigma_m\}_{m\in M}$, a derived message $m' \in \mathcal{M}$, and possibly some auxiliary information $w$. It produces a new signature $\sigma'$ or a special symbol $\perp$ to represent failure. For complicated predicates $P$, the auxiliary information $w$ serves as a witness that $P(M, m') = 1$. To simplify the notation we often drop $w$ as an explicit argument.

As shorthand we write **Sign**$(sk, m)$ := **SignDerive**$(pk, (sk, \varepsilon), m, \cdot)$ to denote that any message can be derived when the original signature is the signing key. For a set of messages $M = \{m_1, \ldots, m_k\} \subset \mathcal{M}^*$ it is convenient to let **Sign**$(sk, M)$ denote independently signing each of the $k$ messages, namely:

$$\mathbf{Sign}(sk, M) := \big(\mathbf{Sign}(sk, m_1), \ldots, \mathbf{Sign}(sk, m_k)\big).$$

**Verify**$(pk, m, \sigma)$: given a public key, message, and purported signature $\sigma$, the algorithm returns 1 if the signature is valid and 0 otherwise.

We assume that testing $m \in \mathcal{M}$ can be done efficiently, and that **Verify** returns 0 if $m \notin \mathcal{M}$.

*Correctness* We require that for all key pairs $(sk, pk)$ generated by $\mathbf{KeyGen}(1^n)$ and for all $M \in \mathcal{M}^*$ and $m' \in \mathcal{M}$ we have:

- if $P(M, m') = 1$ then $\mathbf{SignDerive}(pk, (\mathbf{Sign}(sk, M), M), m') \neq \perp$, and
- for all signature tuples $\{\sigma_m\}_{m \in M}$ such that $\sigma' \leftarrow \mathbf{SignDerive}(pk, (\{\sigma_m\}_{m \in M}, M), m') \neq \perp$, we have $\mathbf{Verify}(pk, m', \sigma') = 1$.

In particular, correctness implies that a signature generated by **SignDerive** can be used as an input to **SignDerive** so that signatures can be further derived from derived signatures, if allowed by $P$.

*Derivation efficiency* In many cases it is desirable that the size of a derived signature depend only on the size of the derived message. This rules out signatures that expand as one iteratively calls **SignDerive**. All the constructions in this paper are derivation efficient in this sense.

**Definition 2.2.** (*Derivation-Efficient*) A signature scheme is derivation-efficient if there exists a polynomial $p$ such that for all $(pk, sk) \leftarrow \mathbf{KeyGen}(1^\lambda)$, set $M \subseteq \mathcal{M}^*$, signatures $\{\sigma_m\}_{m \in M} \leftarrow \mathbf{Sign}(sk, M)$ and derived messages $m'$ where $P(M, m') = 1$, we have

$$|\mathbf{SignDerive}(pk, \{\sigma_m\}_{m \in M}, M, m')| = p(\lambda, |m'|).$$

## 2.1. *Security: Unforgeability*

To define unforgeability, we extend the basic notion of existential unforgeability with respect to adaptive chosen-message attacks [35]. The definition captures the idea that if the attacker is given a set of signed messages (either primary or derived) then the only messages he can sign are derivations of the signed messages he was given. This is defined using a game between a challenger and an adversary $\mathcal{A}$ with respect to scheme $\Pi$ over message space $\mathcal{M}$.

— Game $\mathbf{Unforg}(\Pi, \mathcal{A}, \lambda, P)$:

**Setup:** The challenger runs $\mathbf{KeyGen}(1^\lambda)$ to obtain $(pk, sk)$ and sends $pk$ to $\mathcal{A}$. The challenger maintains two sets $T$ and $Q$ that are initially empty.

**Queries:** Proceeding adaptively, the adversary issues the following queries to the challenger:

- *Sign*$(m \in \mathcal{M})$: the challenger generates a unique handle $h$, runs $\mathbf{Sign}(sk, m) \rightarrow \sigma$ and places $(h, m, \sigma)$ into a table $T$. It returns the handle $h$ to the adversary.
- *SignDerive*$(\boldsymbol{h} = (h_1, \ldots, h_k),\ m')$: the oracle retrieves the tuples $(h_i, \sigma_i, m_i)$ in $T$ for $i = 1, \ldots, k$, returning $\perp$ if any of them do not exist. Let $M := (m_1, \ldots, m_k)$ and $\{\sigma_m\}_{m \in M} := \{\sigma_1, \ldots, \sigma_k\}$. If $P(M, m')$ holds, then the oracle generates a new unique handle $h'$, runs $\mathbf{SignDerive}(pk, (\{\sigma_m\}_{m \in M}, M), m') \rightarrow \sigma'$ and places $(h', m', \sigma')$ into $T$, and returns $h'$ to the adversary.
- *Reveal*$(h)$: Returns the signature $\sigma$ corresponding to handle $h$, and adds $(\sigma', m')$ to the set $Q$.

**Output:** Eventually, the adversary outputs a pair $(\sigma', m')$. The output of the game is 1 (i.e., the adversary wins the game) if:

- **Verify**$(pk, m', \sigma') = 1$ and,
- let $M \subseteq \mathcal{M}$ be the set of messages in $Q$ then $P^*(M, m') = 0$ where $P^*$ is the closure of $P$ from Definition 2.1.

Else, the output of the game is 0. Define $\mathbf{Forg}_{\mathcal{A}}$ as the probability that Pr[**Unforg**$(\Pi, \mathcal{A}, \lambda, P) = 1$].

Interestingly, for some predicates it may be difficult to test if the adversary won the game. For all the predicates we consider in this paper, this will be quite easy.

**Definition 2.3.** (*(Unforgeability*) A $P$-homomorphic signature scheme $\Pi$ is **unforgeable** with respect to adaptive chosen-message attacks if for all PPT adversaries $\mathcal{A}$, the function $\mathbf{Forg}_{\mathcal{A}}$ is negligible in $\lambda$.

A $P$-homomorphic signature scheme $\Pi$ is **selective unforgeable** with respect to adaptive chosen-message attacks if for all PPT adversaries $\mathcal{A}$ who begin the above game by announcing the message $m'$ on which they will forge, $\mathbf{Forg}_{\mathcal{A}}$ is negligible in $\lambda$.

*Properties of the definition* By taking $P$ to be the equality oracle, namely $P(x, y) = 1$ iff $x = y$, we obtain the standard unforgeability requirement for signatures.

Notice that *Sign* and *SignDerive* queries return handles, but do not return the actual signatures. A system proven secure under this definition adequately rules out the following attack: suppose $(m, \sigma)$ is a message signature pair and $(m', \sigma')$ is a message-signature pair derived from it, namely $\sigma' = \mathbf{SignDerive}(pk, \sigma, m, m')$. For example, suppose $m'$ is a quote from $m$. Then given $(m', \sigma')$ it should be difficult to produce a signature on $m$ and indeed our definition treats a signature on $m$ as a valid forgery.

The unforgeability game imposes some constraints on $P$: (1) $P$ must be reflexive, i.e., $P(m, m) = 1$ for all $m \in \mathcal{M}$, (2) $P$ must be monotone, i.e., $P(M, m') \Rightarrow P(M', m')$ where $M \subseteq M'$. It is easy to see that predicates that do not satisfy these requirements cannot be realized under Definition 2.3.

### 2.2. *Security: Context Hiding (a.k.a., Privacy)*

Let $M$ be some set and let $m'$ be a derived message from $M$ (i.e., $P(M, m') = 1$). Context hiding captures the idea that a signature on $m'$ derived from signatures on $M$ should reveal no information about $M$ beyond what is revealed by $m'$. For example, in the case of quoting, a signature on a quote from $m$ should reveal nothing more about $m$: not the length of $m$, not the position of the quote in $m$, etc. The same should hold even if the attacker is given signatures on multiple quotes from $m$.

We put forth the following powerful *statistical* definition of context hiding and discuss its implications following the definition. We were most easily able to leverage a statistical definition for our proofs, although we also give an alternative *computational* definition in Sect. 2.3.

**Definition 2.4.** (*Strong Context Hiding*) Let $M \subseteq \mathcal{M}^*$ and $m' \in \mathcal{M}$ be messages such that $P(M, m') = 1$. Let $(pk, sk) \leftarrow \mathbf{KeyGen}(1^{\lambda})$ be a key pair. A signature scheme (**KeyGen**, **SignDerive**,, **Verify**) is strongly context hiding (for predicate $P$) if for all such triples $((pk, sk), M, m')$, the following two distributions are statistically close:

$$\left\{\left(sk, \{\sigma_m\}_{m\in M} \leftarrow \mathbf{Sign}(sk, M),\ \mathbf{Sign}(sk, m')\right)\right\}_{sk,M,m'},$$
$$\left\{\left(sk, \{\sigma_m\}_{m\in M} \leftarrow \mathbf{Sign}(sk, M),\ \mathbf{SignDerive}(pk, (\{\sigma_m\}_{m\in M}, M), m')\right)\right\}_{sk,M,m'}.$$

The distributions are taken over the coins of **Sign** and **SignDerive**. Without loss of generality, we assume that $pk$ can be computed from $sk$.

The definition states that a derived signature on $m'$, from an honestly-generated original signature, is statistically indistinguishable from a fresh signature on $m'$. This implies that a derived signature on $m'$ is indistinguishable from a signature generated independently of $M$. Therefore, the derived signature cannot (provably) reveal any information about $M$ beyond what is revealed by $m'$. By a simple hybrid argument the same holds even if the adversary is given multiple derived signatures from $M$.

Moreover, Definition 2.4 requires that a derived signature look like a fresh signature even if the original signature on $M$ is known. Hence, if for example someone quotes from a signed recommendation letter and somehow the original signed recommendation letter becomes public, it would be impossible to link the signed quote to the original signed letter. The same holds even if the signing key $sk$ is leaked.

Thus, Definition 2.4 captures a broad range of privacy requirements for derived signatures. Earlier work in this area [19,20,22,39] only considered weaker privacy requirements using more complex definitions. The simplicity and breadth of Definition 2.4 is one of our key contributions.

Definition 2.4 uses statistical indistinguishability meaning that even an unbounded adversary cannot distinguish derived signatures from newly created ones. In Sect. 2.3, we give a definition using computational indistinguishability which is considerably more complex since the adversary needs to be given signing oracles. In the unbounded case of Definition 2.4 the adversary can simply recover a secret key $sk$ from the public key and answer its own signature queries which greatly simplifies the definition of context hiding. All the signature schemes in this paper satisfy the statistical Definition 2.4.

As mentioned above, the context-hiding guarantee applies to all derivations that begin with an honestly-generated signature. One might imagine a scenario where a malicious signer creates a signature that passes the verification algorithm, but contains a "watermark" that allows the signer to detect if other signatures are derived from it. To prevent such attacks from malicious signers, we could alter the definition so that indistinguishability holds for any derivative that results from a signature that passed the verification algorithm.

### 2.3. *A Computational Definition of Context Hiding*

For systems that are strongly context hiding, unforgeability follows from a simpler game than that of Sect. 2.1. In particular, it suffices to just give the adversary the ability to obtain top level signatures signed by $sk$. In this section, we define this simpler unforgeability game and prove equivalence to Definition 2.3 using strong context hiding.

Let (**KeyGen**, **SignDerive**, **Verify**) be a $P$-homomorphic signature scheme for predicate $P$ and message $\mathcal{M}$. Consider the following game to model context hiding:

**Setup:** The challenger runs the algorithm $(pk, sk) \leftarrow \mathbf{KeyGen}(1^\lambda)$ to obtain the public key $pk$ and the secret key $sk$, and gives $pk$ to the adversary.

**Query Phase 1:** Proceeding adaptively, the adversary may query any of the three oracles from the unforgeability game:

- *Sign*$(m \in \mathcal{M})$: (same as in the unforgeability game)
- *SignDerive*$(i \in \mathbb{Z}, m')$: (same as in the unforgeability game)
- *Reveal*$(i \in \mathbb{Z})$: (same as in the unforgeability game)

**Challenge:** At some point, the adversary issues a challenge $(m, m')$ where $P(m, m') = 1$ for any $m, m' \in \mathcal{M}$. The challenger computes the following three values: $\sigma \leftarrow$ **Sign**$(sk, m)$, $\sigma_0 \leftarrow$ **Sign**$(sk, m')$, and $\sigma_1 \leftarrow$ **SignDerive**$(pk, \sigma, m, m')$. The challenger then picks a random $b \in \{0, 1\}$ and returns $(\sigma, \sigma_b)$ to the adversary. Note: there are no restrictions on $m, m'$ other than that they be in the message space; in particular, they could be equal and one or both could have been previously signed.
**Query Phase 2:** Proceeding adaptively, the adversary may query the oracles from Phase 1.
**Output:** Eventually, the adversary will output a bit $b'$ and is said to win if $b = b'$.

We define $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{CH}}$ to be the probability that adversary $\mathcal{A}$ wins in the above game minus $\frac{1}{2}$.

**Definition 2.5.** (*Context Hiding*) For a predicate $P$ and message space $\mathcal{M}$, a $P$-homomorphic signature scheme (**KeyGen**, **Sign**, **SignDerive**, **Verify**) is context hiding if for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{CH}}$ is negligible in $\lambda$.

2.3.1. *Relation to Strong Context Hiding*

**Lemma 2.6.** *A homomorphic signature scheme that is strongly context hiding is context hiding.*

*Proof.* Let $\Pi =$ (**KeyGen**, **SignDerive**, **Verify**) be a homomorphic signature scheme and let $A$ be an adversary that has advantage $\mathbf{Adv}_A^{CH}(\Pi) = p(\lambda)$ in the context-hiding game. The advantage probability for $A$ is taken over the random coins of the key generation, random coins of the *Sign* and *SignDerive* operations used in the first query phase, the random coins used by algorithm $A$, and the random coins used by the rest of the experiment. Therefore by an averaging argument, there must exist *some particular* key pair $(PK, SK) \leftarrow$ **KeyGen**$(1^\lambda; z_1)$, some *particular* random tape $z_q$ for the **Sign** and **SignDerive** operations used in the first query phase, some *particular* random coins $z_A$ for $A$, and some *particular* message pair $(m, m')$ output by $A$ over which the probability of $A$ winning the context-hiding game in this case is at least $p(\lambda)$. Let the values of the random tapes be given as non-uniform advice.

We show how this information can be used to construct a (non-uniform) adversary $A'$ that distinguishes $\{(SK, \sigma, \mathbf{Sign}(SK, m')\}$ from $\{(SK, \sigma, \mathbf{SignDerive}(PK, \sigma, m, m')\}$ with probability $p(\lambda)$ for the triple $((PK, SK), m, m')$. Thus, if $\Pi$ is strongly context hiding, then $p(\lambda)$ must be exponentially small, and so $\Pi$ must also be context-hiding.

The adversary $A'$ works as follows: On input the challenge tuple $(SK, \sigma, \sigma')$, $A'$ begins to run the context-hiding experiment for $A(PK; z_A)$. $A'$ answers the queries that $A$ asks by using $SK$ and the random tape $z_q$ to run **Sign** and **SignDerive**. When $A$

outputs a challenge message pair $(m, m')$ (which must occur by construction), then $A'$ answers with $(\sigma, \sigma')$. $A'$ answers the second-phase queries of $A$ using $SK$ and fresh random coins. Finally, when $A$ outputs $b'$, $A'$ echoes this answer as output and halts.

First observe that $A'$ performs a perfect simulation of the context-hiding game. When the input pair $(\sigma, \sigma')$ corresponds to $(\mathbf{Sign}(SK, m), \mathbf{Sign}(SK, m'))$, then $A'$ simulates the context-hiding game for $b = 0$. In the other case, $A'$ simulates the context-hiding game for $b = 1$. Therefore, $A'$ distinguishes

$$\{(SK, \mathbf{Sign}(SK, m), \mathbf{Sign}(SK, m'))\}_{SK,m,m'}$$
$$\{(SK, \mathbf{Sign}(SK, m), \mathbf{SignDerive}(PK, \sigma, m, m'))\}_{SK,m,m'}$$

with probability $p(\lambda)$. □

2.3.2. *Simplified Unforgeability Under Strong Context Hiding*

We now show how the strong context hiding property can help simplify the security argument for unforgeability. In particular, we introduce a weaker notion of unforgeability in which the adversary only makes calls to the **Sign** oracle and immediately receives a signature.

— Game $\mathbf{NHU}(\Pi, \mathcal{A}, \lambda, P)$: This game is the same as the $\mathbf{Unforg}(\Pi, \mathcal{A}, \lambda, P)$ game with the exception that only the following query is allowed:

— *Sign*$(m \in \mathcal{M})$: the oracle computes $\sigma \leftarrow \mathbf{Sign}(SK, m)$, adds $m$ to $Q$ and returns $\sigma$.

Note, the only difference between game **NHU** and the standard unforgeability game for a signature scheme is that in this game, the adversary only wins if it produces a forgery on a signature $m^*$ such that for all $m \in Q$, $P(m, m^*) = 0$, whereas in the standard unforgeability game, the adversary wins if it produces a signature on *any* message that is not in $Q$.

**Definition 2.7.** A quoteable signature scheme $\Pi$ is **NHU**-unforgeable if for all efficient adversaries $\mathcal{A}$, it holds that $\Pr[\mathbf{NHU}(\Pi, \mathcal{A}, \lambda, P) = 1] < negl(\lambda)$ for some negligible function $\lambda$.

**Lemma 2.8.** *A signature scheme that is* **NHU**-*unforgeable and strongly context hiding is* **Unforg**-*unforgeable.*

*Proof.* Our plan is to present a series of hybrid experiments that are meant to simplify the quotable unforgeability game.

**Hybrid** $H_1(\Pi, \mathcal{A}, \lambda, P)$ Consider the first hybrid experiment $H_1$ which is the same as the unforgeability game $\mathbf{Unforg}(\Pi, \mathcal{A}, \lambda, P)$, with the exception that all *Sign* and *SignDerive* queries are lazily evaluated. That is, when $\mathcal{A}$ makes a query, the experiment responds in the following way:

— *Sign*$(m)$: generate a handle $i$ and record information $(i, ?, m, \epsilon)$ in $T$ and return $i$

— *SignDerive*$(i, m')$: retrieve $(i, z, m, \cdot)$ from $T$, return $\perp$ if it does not exist or if $P(m, m') \neq 1$, generate a new handle $i'$, record $(i', ?, m', i)$ in $T$, and return $i'$
— *Reveal*$(i)$: retrieve $(i, z, m, i_1)$ from $T$ (returning $\perp$ if it does not exist). If $z \neq ?$, then return $z$. Otherwise, if $i_1 = \epsilon$, then compute $\sigma \leftarrow \mathbf{Sign}(SK, m)$, replace the entry $(i, z, m, \epsilon)$ with $(i, \sigma, m, \epsilon)$, and return $\sigma$. Finally, if $i_1 \neq \epsilon$, then recursively call $z_1 \leftarrow$ *Reveal*$(i_1)$, obtain $(i_1, \cdot, m_1, \cdot)$ from $T$ and compute $\sigma \leftarrow \mathbf{SignDerive}(PK, z_1, m_1, m)$. Replace the entry with $(i, \sigma, m, i_1)$, and return $\sigma$.

□

**Claim 2.9.** $\Pr[H_1(\Pi, \mathcal{A}, \lambda, P) = 1] = \Pr[\mathbf{Unforg}(\Pi, \mathcal{A}, \lambda, P) = 1]$.

This claim follows by inspection. For any query that is eventually revealed, the same operations are performed in both $H_1$ and the original game. For any query that is never revealed, no operation in $H_1$ is performed; but this does not affect the view of the adversary, and therefore does not affect the output of the adversary.
**Hybrid** $H_{2,i}(\Pi, \mathcal{A}, \lambda, P)$ The second hybrid is the same as $H_1$ except that the first $i$ queries to *Reveal* are answered using *Reveal*$_2$ described below, and the remaining queries are answered as per $H_1$: (*The only difference is that* $\mathbf{Sign}(SK, m_1)$ *is used in place of* $\mathbf{SignDerive}(PK, z_1, m_1, m)$ *in the second to last sentence.*)

— *Reveal*$_2(i)$: retrieve $(i, z, m, i_1)$ from $T$ (returning $\perp$ if it does not exist). If $z \neq ?$, then return $z$. Otherwise, if $i_1 = \epsilon$, then compute $\sigma \leftarrow \mathbf{Sign}(SK, m)$, replace the entry $(i, z, m, \epsilon)$ with $(i, \sigma, m, \epsilon)$, and return $\sigma$. Finally, if $i_1 \neq \epsilon$, then recursively call $z_1 \leftarrow$ *Reveal*$(i_1)$, obtain $(i_1, \cdot, m_1, \cdot)$ from $T$ and compute $\sigma \leftarrow \mathbf{Sign}(SK, m_1)$. Replace the entry with $(i, \sigma, m, i_1)$, and return $\sigma$.

**Claim 2.10.** *$H_{2,0}(\Pi, \mathcal{A}, \lambda, P)$ is identically distributed to $H_1(\Pi, \mathcal{A}, \lambda, P)$.*

By inspection.

**Claim 2.11.** *$H_{2,i}(\Pi, \mathcal{A}, \lambda, P)$ is identically distributed to $H_{2,i-1}(\Pi, \mathcal{A}, \lambda, P)$ for $i \geq 1$.*

This claim follows via the strong context-hiding property of the signature scheme because this property guarantees $\mathbf{Sign}(SK, m')$ and $\mathbf{SignDerive}(PK, \sigma, m, m')$ are statistically close.

Suppose that $\mathcal{A}$ makes $\ell = poly(\lambda)$ queries. Observe that $H_{2,\ell}(\Pi, \mathcal{A}, \lambda, P)$ only evaluates **Sign**, and only does so on messages that are immediately returned to the adversary. Thus, $H_{2,\ell}$ is syntactically equivalent to the **NHU** game. Since the $H_{2,\ell}$ game enables $\mathcal{A}$ to produce a forgery with the same probability as $\mathbf{Unforg}(\Pi, \mathcal{A}, \lambda, P)$, we have that $\mathbf{Unforg}(\Pi, \mathcal{A}, \lambda, P) = \mathbf{NHU}(\Pi, \mathcal{A}, \lambda, P)$ which completes the lemma. □

### 2.4. *Related Work*

Early work on quotable signatures [19,21,25,36,39,47,48,58] supports quoting from a single document, but does not achieve the privacy or unforgeability properties we are aiming for. For example, if *simple quoting* of messages is all that is desired, then the following folklore solution would suffice: simply sign the Merkle hash of a document. A quote represents some sub-tree of the Merkle hash; so a quoter could include enough intermediate hash nodes along with the original signature in any quote. A verifier could simply hash the quote, and then build the Merkle hash tree using the computed hash and the intermediate hashes, and compare with the original signature. Notice, however, that every quote in this scheme reveals information about the original source document. In particular, each quote reveals information about *where in the document* it appears. Thus, this simple quoting scheme is not *context hiding* in our sense.

The work whose definition is closest to what we envision is the recent work on redacted signatures of Chang et al. [25] and Brzuska et al. [19] (see also Naccache [49, p. 63] and Boneh-Freeman [15, 16]).[4] However, there is a subtle, but fundamental difference between their definition and the privacy notion we are aiming for. In our formulation, a quoted signature should be indistinguishable from a fresh signature, even when the distinguisher is given the original signature. (We capture this by an even stronger game where a derived signature is distributed statistically close to a fresh signature). In contrast, the definitions of [15, 16, 19, 25] do not provide the distinguisher with the original signature. Thus, it may be possible to link a quoted document to its original source (and indeed it is in the constructions of [15, 16, 19, 25]), which can have negative privacy implications. Overcoming such document linkage while maintaining unforgeability is a real technical challenge. This requires moving beyond techniques that use *nonces* to link parts of messages.

Indeed, in most prior constructions, such as [19, 25], nonces are used to prevent "mix-and-match" attacks (e.g., forming a "quote" using pieces of two different messages). Unfortunately, these nonces reveal the history of derivation, since they cannot change during each derivation operation. Arguably, much of the technical difficulty in our current work comes precisely from the effort to meet our definition and hide the lineage. We introduce new techniques in this work which link pieces together using randomness that can be re-randomized in controlled ways.

Another line of work studies computing on authenticated data by holders of secret information. Examples include *sanitizable* signatures [1, 20, 22, 46, 48] that allow a proxy to compute signatures on related messages, but requires the proxy to have a secret key, and *incremental* signatures [6], where the signer can efficiently make small edits to his signed data. In contrast, our proposal is more along the lines of homomorphic encryption and Rivest's vision [51], where *anyone* can compute on the authenticated data.

---

[4] As acknowledged in Sect. 2.2 of Boneh-Freeman [15], our definitional notion is stronger than and predates the "weak context hiding" notion of [15]. Indeed, the fact that [15] uses our framework lends support to its generality, and the fact that they could not achieve our context- hiding notion highlights its difficulty. Their "weak" definition, which is equivalent to [19], only ensures privacy when the original signatures remain hidden. In their system, signature derivation is deterministic and therefore once the original signatures become public it is easy to tell where the derived signature came from. Our signatures achieve full context hiding so that derived signatures remain private no matter what information is revealed. This is considerably harder and is not known how to do for the lattice-based signatures in Boneh-Freeman.

## 3. Generic Constructions for Simple Predicates

Let $\mathcal{M}$ be a finite message space. We say that a predicate $P : \mathcal{M}^* \times \mathcal{M} \to \{0, 1\}$ is a *simple* predicate if the following properties hold:

1. $P$ is false whenever its left input is a tuple of length greater than 1,
2. $P$ is a closed predicate (i.e., $P$ is equal to its closure $P^*$; see Sect. 2.1).
3. For all $m \in \mathcal{M}$, $P(m, m) = 1$.

In this section, we present and discuss generic approaches for computing on authenticated data with respect to *any* simple predicate $P$. Note that the quoting of substrings or subsequences (i.e., redacting) are examples of simple predicates.

We begin with two inefficient constructions. The first takes a brute force approach that constructs long signatures that are easy to verify. The second takes an accumulator approach that constructs shorter signatures at the cost of less efficient verification. We conclude by discussing the limitations of a generic NIZK proof of knowledge approach.

### 3.1. *A Brute Force Construction From Any Signature Scheme*

Let $(G, S, V)$ be a signature scheme with a deterministic signing algorithm.[5] One can construct a $P$-homomorphic signature scheme for any simple predicate $P$ as follows:

**KeyGen**$(1^\lambda)$: The setup algorithm runs $G(1^\lambda) \to (pk, sk)$ and outputs this key pair.
**Sign**$(sk, m \in \mathcal{M})$: While **Sign** is simply a special case of the **SignDerive** algorithm, we will explicitly provide both algorithms here for clarity purposes.
The signature $\sigma$ is the tuple $(S(sk, m), U = \{S(sk, m') \mid m' \in P^0(\{m\})\})$.
**SignDerive**$(pk, \sigma, m, m')$: The derived signature is computed as follows. First check that $P(m, m') = 1$. If not, then output $\perp$. Otherwise, parse $\sigma = (\sigma_1, \ldots, \sigma_k)$ where $\sigma_i$ corresponds to message $m_i$. If for any $i$, $V(pk, m_i, \sigma_i) = 0$, then output $\perp$. Otherwise, the signature is comprised as the set containing $\sigma_i$ for all $m_i$ such that $P(m', m_i) = 1$. Again, by default, let the first sub-signature of the output be the signature on $m'$.
**Verify**$(pk, m, \sigma)$: Parse $\sigma = (\sigma_1, \ldots, \sigma_k)$. Output $V(pk, m, \sigma_1)$.

*Efficiency Discussion* The efficiency of the above approach depends on the message space and the predicate $P$. For instance, the brute force approach for signing a message of $n$ characters, where $P(m, m')$ outputs 1 if and only if $m'$ is a substring of $m$, will result in $O(n^2)$ sub-signatures (one for each of the $O(n^2)$ substrings). If one wanted to “quote" subgraphs from a graph, this approach is intractable, as a graph of $n$ nodes will generate an exponential in $n$ number of subgraphs.

**Theorem 3.1.** (Security from Any Signature) *If $(G, S, V)$ is a secure deterministic signature scheme, then the above signature scheme is unforgeable and context-hiding.*

Proof of the above theorem is rather straightforward. The context-hiding property follows from the uniqueness of the signatures generated by the honest signing algorithms.

[5]Given a signature scheme with a probabilistic signing algorithm, one can convert it to a scheme with a deterministic signing algorithm by: (1) including a pseudorandom function (PRF) seed as part of the secret key, and (2) during the signing algorithm, applying this PRF to the message and using the output as the randomness in the signature. Given any signature scheme, one can also construct a PRF.

The unforgeability property follows from the fact that an adversary cannot obtain a signature on any message not derivable from those she queried or one could use this signature to directly break the regular unforgeability of the underlying signature scheme. The correctness property is actually the most complex to verify: it requires the two restrictions on the predicate $P$ made above.

### 3.2. *An Accumulator-based Construction*

**Assumption 3.2.** (*RSA* [52]) Let $k$ be the security parameter. Let a positive integer $N$ be the product of two random $k$-bit primes $p, q$. Let $e$ be a randomly chosen positive integer less than and relatively prime to $\phi(N) = (p-1)(q-1)$. Then no PPT algorithm given $(N, e)$ and a random $y \in \mathbb{Z}_N^*$ as input can compute $x$ such that $x^e \equiv y \mod N$ with non-negligible probability.

**Lemma 3.3.** (Shamir [55]) *Given $x, y \in \mathbb{Z}_n$ together with $a, b \in \mathbb{Z}$ such that $x^a = y^b$ and $\gcd(a, b) = 1$, there is an efficient algorithm for computing $z \in \mathbb{Z}_n$ such that $z^a = y$.*

**Theorem 3.4.** (Prime Number Theorem) *Define $\pi(x)$ as the number of primes no larger than $x$. For $x > 1$,*

$$\pi(x) > \frac{x}{\lg x}.$$

Consider the following RSA accumulator solution which supports short signatures, but the computation required to derive a new signature is expensive. Let $P$ be any univariate predicate with the above restrictions.

We now describe the algorithms. While **Sign** is simply a special case of the **SignDerive** algorithm, we will explicitly provide both algorithms here for clarity purposes.

**KeyGen**$(1^\lambda)$: The setup algorithm chooses $N$ as a $20\lambda$-bit RSA modulus and a random value $a \in \mathbb{Z}_N$. It also chooses a hash function $H_p$ that maps arbitrary strings to $2\lambda$-bit prime numbers, e.g., [38], which we treat as a random oracle.[6] Output the public key $pk = (H_p, N, a)$ and keep as the secret key $sk$, the factorization of $N$.
**Sign**$(sk, m \in \mathcal{M})$: Let $U = P^0(\{m\}) = \{m' \mid m' \in \mathcal{M} \text{ and } P(m, m') = 1\}$. Compute and output the signature as

$$\sigma := a^{1/(\prod_{u_i \in U} H_p(u_i))} \mod N.$$

**SignDerive**$(pk, \sigma, m, m')$: The derivation is computed as follows. First check that $P(m, m') = 1$. If not, then output $\perp$. Otherwise, let $U' = P^0(\{m'\})$. Compute and output the signature as

$$\sigma' := \sigma^{\prod_{u_i \in U - U'} H_p(u_i)} \mod N.$$

[6] We choose our modulus and hash output lengths to obtain $\lambda$-bit security based on the recent estimates of [57].

Thus, the signature is of the form $a^{1/\prod_{u_i \in U'} H_p(u_i)} \mod N$.

**Verify**$(pk, m, \sigma)$: Accept if and only if $a = \sigma^{\prod_{u_i \in U} H_p(u_i)} \mod N$ where $U = P^0(m)$.

*Efficiency Discussion* In the above scheme, signatures require only one element in $\mathbb{Z}_N^*$. However, the cost of signing depends on $P$ and the size of the message space. For example, computing an $\ell$-symbol quote from an $n$-symbol message requires $O(n(n-\ell))$ evaluations of $H_p$ and $O(n(n-\ell))$ modular exponentiations. The prime search component of $H_p$ will likely be the dominating factor. Verification requires $O(\ell^2)$ evaluations of $H_p$ and $O(\ell^2)$ modular exponentiations, for an $\ell$-symbol quote. Thus, this scheme optimizes on space, but may require significant computation.

**Theorem 3.5.** (Security under RSA) *If the RSA assumption holds, then the above signature scheme is unforgeable and context-hiding in the random oracle model.*

We provide a proof of above theorem by showing the following lemmas.

**Lemma 3.6.** (Context-Hiding) *The homomorphic signature scheme from §3.2 is strongly context-hiding.*

*Proof.* This property is derived from the fact that a signature on any given message is deterministic. Let the public key $PK$ be $(H_p, N, a)$ and challenge be any $m, m'$ where $P(m, m') = 1$. Let $U = P^0(m)$ and $U' = P^0(m')$. Observe that

$$\begin{aligned}
\textbf{Sign}(sk, m) &= \sigma = a^{1/\prod_{u \in U} H_p(u)} \mod N \\
\textbf{Sign}(sk, m') &= \sigma_0 = a^{1/\prod_{u' \in U'} H_p(u')} \mod N \\
\textbf{SignDerive}(pk, (\sigma, m), m') &= \sigma^{\prod_{u \in U-U'} H_p(u)} \mod N \\
&= \left[a^{1/\prod_{u \in U} H_p(u)}\right]^{\prod_{u \in U-U'} H_p(u)} \mod N \\
&= a^{1/\prod_{u' \in U'} H_p(u')} \mod N \\
&= \sigma_0
\end{aligned}$$

Because **Sign**$(sk, m')$ and **SignDerive**$(pk, (\sigma, m), m')$ are identical, for any adversary $\mathcal{A}$, the probability that $\mathcal{A}$ distinguishes the two is exactly $1/2$, and so the advantage in the strong context hiding game is 0. □

**Lemma 3.7.** (Unforgeability) *If the RSA assumption holds, then the Sect. 3.2 homomorphic signature scheme is unforgeable in the* **Unforg** *game in the random oracle model.*

*Proof.* Our reduction only works on certain types of RSA challenges, as in [38]. In particular, this reduction only attempts to solve RSA challenges $(N, e^*, y)$ where $e^*$ is an odd prime. Fortunately, good challenges will occur with non-negligible probability. We know that $e^*$ is less than and relatively prime to $\phi(N) < N$, which implies it cannot be 2. We also know, by Theorem 3.4, that the number of primes that are less than $N$ is at least $\frac{N}{\lg N}$. Thus, a loose bound on the probability of $e^*$ being a prime is $\geq (\frac{N}{\lg N})/N = \frac{1}{\lg N} = \frac{1}{20\lambda}$. □

Now, we describe the reduction. Our proof first applies Lemma 2.8, which allows us to only consider adversaries $\mathcal{A}$ that ask queries to *Sign* oracle in the **NHU** game. Moreover, suppose adversary $\mathcal{A}$ queries the random oracle $H_p$ on at most $s$ unique inputs. Without loss of generality, we will assume that all queries to this deterministic oracle are unique and that whenever *Sign* is called on message $M$, then $H_p$ is automatically called with all unique substrings of $M$. Suppose an adversary $\mathcal{A}$ can produce a forgery with probability $\epsilon$ in the **NHU** game; then we can construct an adversary $\mathcal{B}$ that breaks the RSA assumption (with odd prime $e^*$) with probability $\epsilon/s$ minus a negligible amount as follows.

On input an RSA challenge $(N, e^*, y)$, $\mathcal{B}$ proceeds as follows:

**Setup** $\mathcal{B}$ chooses $2\lambda$-bit distinct prime numbers $e_1, e_2, \ldots, e_{s-1}$ at random, where all $e_i \neq e^*$. Denote this set of primes as $E$. Next, $\mathcal{B}$ makes a random guess of $i^* \in [1, s]$ and saves this value for later. Then it sets

$$a := y^{\prod_{e_i \in E} e_i}.$$

Finally, $\mathcal{B}$ give the public key $PK = (N, a)$ to $\mathcal{A}$ and will answer its queries to random oracle $H_p$ interactively as described below.

**Queries** Proceeding adaptively, $\mathcal{B}$ answers the oracle and sign queries made by $\mathcal{A}$ as follows:

1. $H_p(x)$ : When $\mathcal{A}$ queries the random oracle for the $j$th time, $\mathcal{B}$ responds with $e^*$ if $j = i^*$, with $e_j$ if $j < i^*$ and $e_{j-1}$ otherwise. Recall that we stipulated that each call to $H_p$ was unique. Denote $x^*$ as the input where $H_p(x^*) = e^*$.
2. $Sign(M)$: Let $U = P^0(M)$. If $x^* \in U$, then $\mathcal{B}$ aborts the simulation. Otherwise, $\mathcal{B}$ calls $H_p$ on all elements of $U$ not previously queried to $H_p$. Let **primes**$(U)$ denote the set of primes derived by calling $H_p$ on the strings of $U$. Then, it computes the signature as $\sigma := y^{\prod_{e_i \in (E - \mathbf{primes}(U))} e_i} \mod N$ and returns $(M, \sigma)$.

**Response** Eventually, $\mathcal{A}$ outputs a valid message-signature pair $(M, \sigma)$, where $M$ is not a derivative of an element returned by *Sign*. If $M$ was not queried to $H_p$ or if $M \neq x^*$, then $\mathcal{B}$ aborts the simulation. Otherwise, let $U = P^0(x^*) - \{x^*\}$ and **primes**$(U)$ denote the set of primes derived by calling $H_p$ on the strings of $U$. It holds that $a^{1/\prod_{e_i \in \mathbf{primes}(U)} e_i} = y^{\prod_{e_i \in E - \mathbf{primes}(U)} e_i} = \sigma^{e^*} \mod N$. Since $y, \sigma \in \mathbb{Z}_N$ and $\gcd(e^*, \prod_{e_i \in E - \mathbf{primes}(U)} e_i) = 1$ (recall, they are all distinct primes), then $\mathcal{B}$ can apply the efficient algorithm from Lemma 3.3 to obtain a value $z \in \mathbb{Z}_N$ such that $z^{e^*} = y \mod N$. $\mathcal{B}$ outputs $z$ as the solution to the RSA challenge.

**Analysis** We now argue that any successful adversary $\mathcal{A}$ against our scheme will have success in the game presented by $\mathcal{B}$. To do this, we first define a sequence of games, where the first game models the real security game and the final game is exactly the view of the adversary when interacting with $\mathcal{B}$. We then show via a series of claims that if $\mathcal{A}$ is successful against Game $j$, then it will also be successful against Game $j + 1$.

**Game 1:** The same as Game **NHU**, with the exception that at the beginning of the game $\mathcal{B}$ guesses an index $1 \leq i^* \leq s$ and $e^*$ is the response of the $i^*$th query to $H_p$.
**Game 2:** The same as Game 1, with the exception that $\mathcal{A}$ fails if any output of $H_p$ is repeated.
**Game 3:** The same as Game 2, with the exception that $\mathcal{A}$ fails if it outputs a valid forgery $(M, \sigma)$ where $M$ was not queried to $H_p$.
**Game 4:** The same as Game 3, with the exception that $\mathcal{A}$ fails if it outputs a valid forgery $(M, \sigma)$ where $M \neq x^*$.

Notice that Game 4 is exactly the view of the adversary when interacting with $\mathcal{B}$. We complete this argument by linking the probability of $\mathcal{A}$'s success in these games via a series of claims. The only non-negligible probability gap comes between Games 3 and 4, where there is a factor $1/s$ loss.

Define $\mathbf{Adv}_{\mathcal{A}}[\text{Game } x]$ as the advantage of adversary $\mathcal{A}$ in Game $x$.

**Claim 3.8.** *If $H_p$ is a truly random function, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 1}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game } \mathbf{NHU}].$$

*Proof.* The value $e^*$ was chosen independently at random by the RSA challenger, just as $H_p$ would have done. □

**Claim 3.9.** *If $H_p$ is a truly random function, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 2}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 1}] - \frac{2s^2\lambda}{2^{2\lambda}}.$$

*Proof.* Consider the probability of a repeat occurring when $s$ $2\lambda$-bit primes are chosen at random. By Theorem 3.4, we know that there are at least $2^{2\lambda}/(2\lambda)$ $2\lambda$-bit primes. Thus, a repeat will occur with probability $< \sum^s s/(2^{2\lambda}/2\lambda) = 2s^2\lambda/2^{2\lambda}$, which is negligible since $s$ must be polynomial in $\lambda$. □

**Claim 3.10.** *If $H_p$ is a truly random function, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 3}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 2}] - \frac{2\lambda}{2^{2\lambda}}.$$

*Proof.* If $M$ was never queried to $H_p$, then $\sigma$ can only be a valid forgery if $\mathcal{A}$ guessed the $2\lambda$-bit prime that $H_p$ would respond with on input $M$. By Theorem 3.4, there are at least $2^{2\lambda}/2\lambda$ such primes and thus the probability of $\mathcal{A}$'s correct guess is at most $2\lambda/2^{2\lambda}$, which is negligible. □

**Claim 3.11.**

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 4}] = \frac{\mathbf{Adv}_{\mathcal{A}}[\text{Game 3}]}{s}.$$

*Proof.* At this point in our series of games, we conclude that $\mathcal{A}$ forges on one of the $s$ queries to $H_p$ and that $1 \le i^* \le s$ was chosen at random. Thus, the probability that $\mathcal{A}$ forges on the $i^*$th query is $1/s$. □

This completes our proof. □

### 3.3. *On the Limitations of Using a Generic NIZK Proof of Knowledge Approach*

Another general approach that one might be tempted to try is to use an NIZK [11] proof of knowledge system to generate a signature on $m'$ by proving that one knows a signature on some $m$ such that $P(m, m')$ holds. Unfortunately, this approach has the standard drawback of generality in that it requires circuit-based (nonblack-box) reductions. In particular, generic NIZK proof systems would require expressing the signature verification method and quoting predicate into, for example, a boolean circuit, a 3-SAT formula, or a Hamiltonian-circuit representation. Even if one were to tailor an NIZK proof of knowledge for these specific statements and therefore avoid costly reductions, another problem emerges with re-quoting. When a quote is re-quoted, then the same process happens for both the original signature scheme circuit, the predicate, *and* the proof system. Aside from the inefficiency, using standard NIZKPoK systems would leak information about the size of the original message and quotes, and therefore would not satisfy our context-hiding property.[7]

## 4. A Powers-of-2 Construction for Quoting Substrings

We begin by describing our algebraic setting.

### 4.1. *Bilinear Groups and the CDH Assumption*

*Bilinear Groups and the CDH Assumption* Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order $p$. A *bilinear map* is an efficient mapping $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ which is both: (*bilinear*) for all $g \in \mathbb{G}$ and $a, b \leftarrow \mathbb{Z}_p$, $\mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{ab}$; and (*non-degenerate*) if $g$ generates $\mathbb{G}$, then $\mathbf{e}(g, g) \neq 1$. We will focus on the Computational Diffie-Hellman assumption in these groups.

**Assumption 4.1.** (*CDH* [30]) Let $g$ generate a group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$. For all PPT adversaries $\mathcal{A}$, the following probability is negligible in $\lambda$: $\Pr[a, b, \leftarrow \mathbb{Z}_p; z \leftarrow \mathcal{A}(g, g^a, g^b) : z = g^{ab}]$.

### 4.2. *The Quoting Construction*

We now provide our main construction for quoting substrings in a text document. It achieves the best time/space efficiency trade-off to our knowledge for this problem. We

[7] Using non-interactive CS-proofs [44] in the random oracle model may reduce the size of the proof, but we do not know how to avoid leaking the size of the theorem statement which also violates the context-hiding property.

will have two different types of signatures called Type I and Type II, where a Type I signature can be quoted down to another Type I or Type II signature. A Type II signature cannot be quoted any further, but will be a shorter signature. The quoting algorithm will allow us to quote anything that is a substring of the original message. We point out that the Type I, II signatures of this system conform to the general framework given in Sect. 2. In particular, we can view a message $M$ as a pair $(t, m) \in \{0, 1\}, \{0, 1\}^*$, where an upper-bound on the length of $m$ is fixed at key generation time. The bit $t$ will identify the message as being Type I or Type II (assume $t = 1$ signifies Type I signatures), and $m$ will be the quoted substring. The predicate

$$P(M = (t, m), M' = (t', m')) = \begin{cases} 1 & \text{if } t = 1 \text{ and } m' \text{ is a substring of } m; \\ 0 & \text{otherwise.} \end{cases}$$

The bit $t'$ will indicate whether the new message is Type I or II (i.e., whether the system can quote further). We note that this description allows an attacker to distinguish between any Type I signature from any Type II signature since the “type bit" of the messages will be different, and thus they will technically be two different messages even if the substring components are equal. For this reason, we will only need to prove context hiding between messages of Type I or Type II, but not across types. In general, flipping the bit $t$ will not result in a valid signature of a different type on the same core message, because the format will be wrong; however, moving from a Type I to a Type II on the same core message is not considered a forgery since Type II signatures can be legally derived from Type I.

For presentational clarity, we will split the description of our quoting algorithm into two quoting algorithms for quoting to Type I and to Type II signatures; likewise we will split the description of our verification algorithm into two separate verification algorithms, one for each type of signature. The type of signature used or created (i.e., bit $t$) will be implicit in the description.

**Notation.** We use notation $m_{i,j}$ to denote the substring of $m$ of length $j$ starting at position $i$.

**Intuition:** We begin by giving some intuition. We design Type I signatures that allow re-quoting and Type II signatures that cannot be further quoted, but are ultra-short. For an original message of length $n$, our signature structure should be able to accommodate starting at any position $1 \leq i \leq n$ and quoting any length $1 \leq \ell \leq (n - i + 1)$ substring.[8]

To (roughly) see how this works for a message of length $n$, visualize $(n+1)$ columns with $(\lfloor \lg n \rfloor + 2)$ rows as in Fig. 1. The columns correspond to the characters of the message, so if the 14-character message is “abcdefghijklmn" then there are 15 columns, with a character in between each column. The rows correspond to the numbers $\lg n$

[8]Technically, our predicate $P(m, m')$ will take the quote from the first occurrence of substring $m'$ in $m$, but for the moment imagine that we allowed quoting from anywhere in $m$.

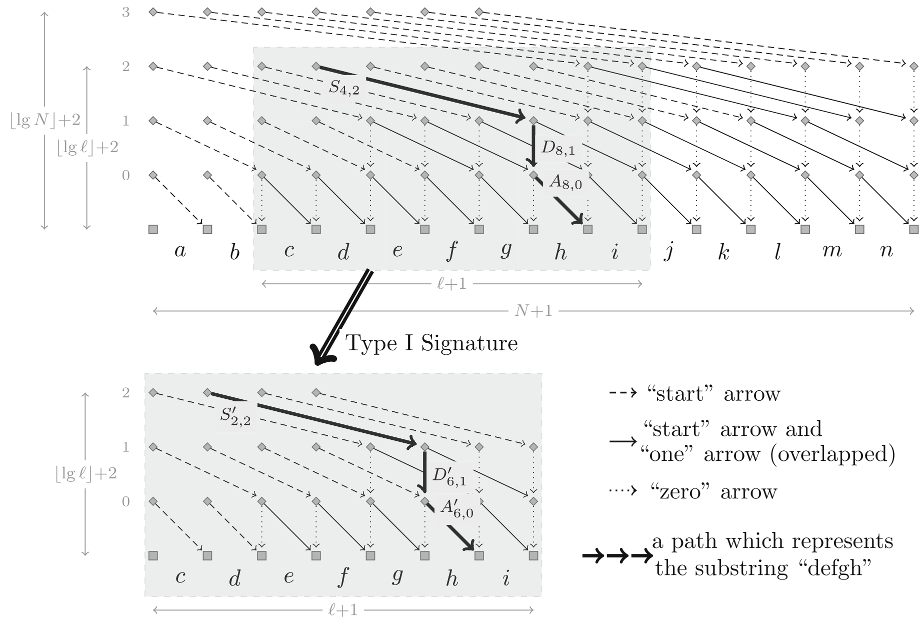


**Fig. 1.** The *top diagram* represents a signature on *abcdefghijklmn* with length $N = 14$. Each *arrow* corresponds to some group elements in the construction. Logically, whenever the elements corresponding to an arrow are included in a quoted signature, the characters underneath this arrow are included in the quoted message. The *bold* path through the *top diagram* shows how to construct a Type II signature on *defgh*; it is very short, but cannot be re-quoted. The *gray box* in this figure shows how to construct a Type I signature on *cdefghi* of length $\ell = 7$; it includes all the *arrows* in the lower figure and can be re-quoted. A technical challenge is to enforce that following the *arrows* is the only way to form a valid signature.

down to 0, plus an extra row at the bottom.[9] Each location in the matrix (except along the bottom-most row) contains one or more out-going arrows. We'll establish rules for when these arrows exist and where each arrow ends shortly.

A Type II quote will trace a ($\lg n$+1)-length path on these arrows through this matrix starting in a row (with outgoing arrows) of the column that begins the quote and ending in the lowest row of the first column after the quote ends. The starting row corresponds to the largest power of two less than or equal to the length of the desired quote. For example, to quote “bcdef", start in row 2 immediately to the left of “b” (because $2^2 = 4$ is the largest power of two less than 5) and end in row 0 immediately to the right of “f”. Intuitively, taking an arrow over a character includes it in the quote. A Type II quote on “defgh" is illustrated in Fig. 1.

A technical challenge is to make this a $O(\lg n)$-length path rather than a $O(n)$-length path. To do this, the key insight is to view the length of any possible quote as the sum of powers of two and to allow arrows that correspond to covering the quote in pieces of size corresponding to one operand of the sum at a time. Each location $(i_c, i_r)$ in the matrix (except the bottom-most row) contains:

[9] The lowest row is intentionally not assigned a number. The second lowest row is row 0. We do this so that row $i$ can correspond to a jump of length $2^i$.

- *a "start" arrow*: an arrow that goes down one row and over $2^{i_r}$ columns ending in $(i_c + 2^{i_r}, i_r - 1)$, if this end point is in the matrix. This adds all characters from position $i_c$ to $i_c + 2^{i_r} - 1$ to the quoted substring; effectively adding the largest power-of-two-length prefix of the quote characters. This arrow indicates that the quote starts here. These are represented as $S_{i,j}, \widetilde{S_{i,j}}$ pairs in our construction.
- *a "one" arrow*: an arrow that operates similarly to start arrows and is used to include characters after a start arrow has been used. These are represented as $A_{i,j}, \widetilde{A_{i,j}}$ pairs in our construction.
- *a "zero" arrow*: an arrow that goes straight down one row ending in $(i_c, i_r - 1)$. This does not add any characters to the quoted substring. These are represented as $D_{i,j}, \widetilde{D_{i,j}}$ pairs in our construction.

A Type II quote always starts with a start arrow and then contains one and zero arrows according to the binary representation of the length of the quote. In our example of original message "abcdefghijklmn", we have 15 columns and 5 rows. We will logically divide our desired substring of "bcdef" (length $5 = 2^2 + 2^0 = 4 + 1$) into its powers-of-two components "bcde"(length $4 = 2^2$) and "f" (length $1 = 2^0$). To form the Type II quote, we start in row 2 (since $4 = 2^2$) of column 2 (to the left of 'b') and take the start arrow ($S_{2,2}$) to row 1 of column 7, take the zero arrow ($D_{7,1}$) to row 0 of column 7, and then take the one arrow ($A_{7,0}$) to the lowest row of column 8. The arrows "pass over" the characters "bcdef". Figure 1 illustrates this for quote "defgh".

For a quote of length $\ell$, the elements on this $O(\lg \ell)$-length path of arrows form a very short Type II signature. For Type I signatures, we include all the elements corresponding to all arrows that make connections within the columns corresponding to the quote. We illustrate this in Fig. 1. This allows quoting of quotes with a signature size of $O(\ell \lg \ell)$.

It is essential for security that the signature structure and data algorithm enforce that the quoting algorithm be used and not allow an attacker to "splice" together a quote from different parts of the signature. We realize this by adding in random "chaining" variables. In order to cancel these out and get a well-formed Type II quote a user must intuitively follow the prescribed procedure (i.e., following the arrows is the only way to form a valid quote).

**The Construction:** We now describe our algorithms. While **Sign** is simply a special case of the **SignDerive** algorithm, we will explicitly provide both algorithms here for clarity purposes.

**KeyGen**$(1^\lambda)$: The algorithm selects a bilinear group $\mathbb{G}$ of prime order $p > 2^\lambda$ with generator $g$. Let $L$ be the maximum message length supported and denote $n = \lfloor \lg(L) \rfloor$. Let $H : \{0,1\}^* \to \mathbb{G}$ and $H_s : \{0,1\}^* \to \mathbb{G}$ be the description of two hash functions that we model as random oracles. Choose random $z_0, \ldots, z_{n-1}, \alpha \in \mathbb{Z}_p$. The secret key is $(z_0, \ldots, z_{n-1}, \alpha)$ and the public key is:

$$PK = (H, H_s, g, g^{z_0}, \ldots, g^{z_{n-1}}, \mathbf{e}(g, g)^\alpha).$$

**Sign**$(sk, M = (t, m) \in \{0,1\} \times \Sigma^{\ell \leq L})$: If $t = 1$, signatures produced by this algorithm are Type I as described below. If $t = 0$, the Type II signature can be obtained by running this algorithm and then running the Quote-Type II algorithm below to obtain a quote on the entire message. The message space is treated as $\ell \leq L$ symbols from alphabet $\Sigma$.

Recall: we use notation $m_{i,j}$ to denote the substring of $m$ of length $j$ starting at position $i$.
For $i = 3$ to $\ell + 1$ and $j = 0$ to $\lfloor \lg(i-1) - 1 \rfloor$, choose random values $x_{i,j} \in \mathbb{Z}_p$. These will serve as our random "chaining" variables, and they should all "cancel" each other out in our short Type II signatures. By definition, set $x_{i,-1} := 0$ for all $i = 1$ to $\ell + 1$.
A signature is comprised of the following values for $i = 1$ to $\ell$ and $j = 0$ to $\lfloor \lg(\ell - i + 1) \rfloor$, for randomly chosen values $r_{i,j} \in \mathbb{Z}_p$:

[start arrow: start and include power $j$]

$$S_{i,j} = g^{\alpha} g^{-x_{i+2^j,j-1}} H_s(m_{i,2^j})^{r_{i,j}} \quad , \quad \widetilde{S_{i,j}} = g^{r_{i,j}}$$

Together with the following values for $i = 3$ to $\ell$ and $j = 0$ to $\min(\lfloor \lg(i-1) - 1 \rfloor, \lfloor \lg(\ell - i + 1) \rfloor)$, for randomly chosen values $r'_{i,j} \in \mathbb{Z}_p$:

[one arrow: include power $j$ and decrease $j$]

$$A_{i,j} = g^{x_{i,j}} g^{-x_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}} \quad , \quad \widetilde{A_{i,j}} = g^{r'_{i,j}}.$$

Together with the following values for $i = 3$ to $\ell + 1$ and $j = 0$ to $\lfloor \lg(i-1) - 1 \rfloor$, for randomly chosen values $r''_{i,j} \in \mathbb{Z}_p$:

[zero arrow: decrease $j$]

$$D_{i,j} = g^{x_{i,j}} g^{-x_{i,j-1}} g^{z_j r''_{i,j}} \quad , \quad \widetilde{D_{i,j}} = g^{r''_{i,j}}$$

We provide an example of how to form Type II signatures from this construction shortly. To see why our $A_{i,j}$ and $D_{i,j}$ values start at $i = 3$, note that Type II quotes at position $i$ of length $2^0 = 1$ symbol include only the $S_{i,0}$ value, where the $x_{\cdot,0-1}$ term is 0 by definition. Type II quotes at position $i$ of length $2^1 = 2$ symbols include the $S_{i,1}$ value plus an additional $D_{i+2,0}$ term to cancel out the $x_{i+2,0}$ value (leaving only $x_{i+2,-1} = 0$). Quotes at position $i$ of length $2^1 + 1 = 3$ symbols include the $S_{i,1}$ value plus an additional $A_{i+2,0}$ term to cancel out the $x_{i+2,0}$ value (leaving only $x_{i+3,-1} = 0$). Since we index strings from position 1, the first position to include an $A_{i,j}$ or $D_{i,j}$ value is $i + 2 = 3$.
**SignDerive**$(pk, \sigma, M = (t, m), M' = (t', m'))$: If $P(M, M') = 0$, output $\perp$. Otherwise, if $t' = 1$, output Quote-Type I$(PK, \sigma, m, m')$; if $t' = 0$, output Quote-Type II$(PK, \sigma, m, m')$, where these algorithms are defined below.
**Quote-Type I**$(pk, \sigma, m, m')$: The quote algorithm takes a Type I signature and produces another Type I signature that maintains the ability to be quoted again. Intuitively, this operation will simply find a substring $m'$ in $m$, keep only the components associated with this substring and re-randomize them all (both the $x_{i,j}$ and $r_{i,j}$ terms in every component).
If $m'$ is not a substring of $m$, then output $\perp$. Otherwise, let $\ell' = |m'|$. Determine the first index $k$ at which substring $m'$ occurs in $m$. Parse $\sigma$ as a collection of $S_{i,j}$, $\widetilde{S_{i,j}}$, $A_{i,j}$, $\widetilde{A_{i,j}}$, $D_{i,j}$, $\widetilde{D_{i,j}}$ values, exactly as would come from **Sign** with $\ell = |m|$.
First, we choose re-randomization values (to re-randomize the $x_{i,j}$ terms of $\sigma$). For $i = 2$ to $\ell' + 1$ and $j = 0$ to $\lfloor \lg(i-1) - 1 \rfloor$, choose random values $y_{i,j} \in \mathbb{Z}_p$. Set $y_{i,-1} := 0$

for all $i = 1$ to $\ell' + 1$. Later, we will choose $t_{i,j}$ values to re-randomize the $r_{i,j}$ terms of $\sigma$.
The quote signature $\sigma'$ is comprised of the following values:
For $i = 1$ to $\ell'$ and $j = 0$ to $\lfloor \lg(\ell' - i + 1) \rfloor$, for randomly chosen $t_{i,j} \in \mathbb{Z}_p$:

$$S'_{i,j} = S_{i+k-1,j} \cdot g^{-y_{i+2^j,j-1}} H_s(m_{i+k-1,2^j})^{t_{i,j}}, \quad \widetilde{S'_{i,j}} = \widetilde{S_{i+k-1,j}} \cdot g^{t_{i,j}}.$$

Together with the following values for $i = 3$ to $\ell'$ and $j = 0$ to $\mathsf{min}(\lfloor \lg(i-1) - 1 \rfloor, \lfloor \lg(\ell' - i + 1) \rfloor)$, for randomly chosen $t'_{i,j} \in \mathbb{Z}_p$:

$$A'_{i,j} = A_{i+k-1,j} \cdot g^{y_{i,j}} g^{-y_{i+2^j,j-1}} H(m_{i+k-1,2^j})^{t'_{i,j}}, \quad \widetilde{A'_{i,j}} = \widetilde{A_{i+k-1,j}} \cdot g^{t'_{i,j}}.$$

Together with the following values for $i = 3$ to $\ell' + 1$ and $j = 0$ to $\lfloor \lg(i-1) - 1 \rfloor$, for randomly chosen $t''_{i,j} \in \mathbb{Z}_p$:

$$D'_{i,j} = D_{i+k-1,j} \cdot g^{y_{i,j}} g^{-y_{i,j-1}} g^{z_j t''_{i,j}}, \quad \widetilde{D'_{i,j}} = \widetilde{D_{i+k-1,j}} \cdot g^{t''_{i,j}}$$

**Quote-Type II**$(pk, \sigma, m, m')$: The quote algorithm takes a Type I signature and produces a Type II signature. If $P(m, m') \neq 1$, then output $\perp$.
A quote is computed from one start value and logarithmically many subsequent pieces depending on the bits of $|m'|$. All signature pieces must be re-randomized to prevent context-hiding attacks.
Consider the length $\ell'$ written as a binary string. Let $\beta'$ be the largest index of $\ell' = |m'|$ that is set to 1, where *we start counting with zero as the least significant bit*. That is, set $\beta' = \lfloor \lg(\ell') \rfloor$. Select random values $v, v_{\beta'-1}, \ldots, v_0 \in \mathbb{Z}_p$. Set the start position as $B := S_{k,\beta'}$ and $k' := k + 2^{\beta'}$. Then, from $j = \beta' - 1$ down to 0, proceed as follows:

- If the $j$th bit of $\ell'$ is 1, set $B := B \cdot A_{k',j} \cdot H(m_{k',2^j})^{v_j}$, set $k' := k' + 2^j$, and $Z_j := \widetilde{A_{k',j}} \cdot g^{v_j}$;
- If the $j$th bit of $\ell'$ is 0, set $B := B \cdot D_{k',j} \cdot g^{z_j v_j}$ and $Z_j := \widetilde{D_{k',j}} \cdot g^{v_j}$.

To end, re-randomize as $B := B \cdot H_s(m_{k,2^\beta})^v$ and $\widetilde{S} := \widetilde{S_{k,\beta}} \cdot g^v$; output the quote as

$$\sigma' = (B, \widetilde{S}, Z_{\beta-1}, \ldots, Z_0).$$

**Verify**$(pk, M = (t, m), \sigma)$: If $t = 1$, output Verify-Type I$(pk, m, \sigma)$. Otherwise, output Verify-Type II$(pk, m, \sigma)$, where these algorithms are defined immediately below.
**Verify-Type I**$(pk, m, \sigma)$: Parse $\sigma$ as the set of $S_{i,j}, \widetilde{S_{i,j}}, A_{i,j}, \widetilde{A_{i,j}}, D_{i,j}, \widetilde{D_{i,j}}$. Let $\ell = |m|$.
Let $X_{i,j}$ denote $e(g, g)^{x_{i,j}}$. We can compute these values as follows. The value $X_{i,-1} = 1$, since for all $i = 1$ to $\ell + 1$, $x_{i,-1} = 0$. For $i = 3$ to $\ell + 1$ and $j = 0$ to $\lfloor \lg(i-1) - 1 \rfloor$, we compute $X_{i,j}$ in the following manner: Let $I = i - 2^{j+1}$ and $J = j + 1$. Next, compute $X_{i,j} = \left(e(g, g)^\alpha \cdot e(H_s(m_{I,2^J}), \widetilde{S_{I,J}})\right) / e(S_{I,J}, g)$. The verification accepts if and only if all of the following hold:

- for $i = 3$ to $\ell$ and $j = 0$ to $\min(\lfloor \lg(i-1) - 1 \rfloor, \lfloor \lg(\ell - i + 1) \rfloor)$,

$$e(A_{i,j}, g) = X_{i,j} / X_{i+2^j, j-1} \cdot e(H(m_{i,2^j}), \widetilde{A_{i,j}})$$

- and for $i = 3$ to $\ell + 1$ and $j = 0$ to $\lfloor \lg(i-1) - 1 \rfloor$, $e(D_{i,j}, g) = X_{i,j} / X_{i,j-1} \cdot e(g^{z_j}, \widetilde{D_{i,j}})$.

**Verify-Type II**$(pk, m, \sigma)$: We give the verification algorithm for Type II signatures. Parse $\sigma$ as $(B, \widetilde{S}, Z_{\beta-1}, \ldots, Z_0)$. Let $\ell = |m|$ and $\beta$ be the index of the highest bit of $\ell$ that is set to 1. If $\sigma$ does not include exactly $\beta$ $Z_i$ values, reject. Set $C := 1$ and $k = 1$. From $j = \beta - 1$ down to 0, proceed as follows:

- If the $j$th bit of $\ell$ is 1, set $C := C \cdot \mathbf{e}(H(m_{k,2^j}), Z_j)$ and $k := k + 2^j$;
- If the $j$th bit of $\ell$ is 0, set $C := C \cdot \mathbf{e}(g^{z_j}, Z_j)$.

Accept if and only if $\mathbf{e}(B, g) = \mathbf{e}(g, g)^\alpha \cdot \mathbf{e}(H_s(m_{1,2^\beta}), \widetilde{S}) \cdot C$.

**Theorem 4.2.** (Security under CDH) *If the CDH assumption holds in $\mathbb{G}$, then the above quotable signature scheme is selectively quote unforgeable and context-hiding in the random oracle model.*

*Efficiency Discussion* This construction presents the best known balance between time and space complexities. The quotable (Type I) signatures require $O(\ell \lg \ell)$ elements in $\mathbb{G}$ for a message of length $\ell$. The group elements in both types of signatures are elements of $\mathbb{G}$, and not the target group $\mathbb{G}_T$. Typically, elements of the base group are significantly smaller than elements of the target group. Computing quotes requires $O(\ell \lg \ell)$ modular exponentations for a quote of length $\ell$ for re-randomization. Similarly, verification also requires $O(\ell \lg \ell)$ pairings.

The non-quotable (Type II) signatures require only $O(\lg \ell)$ elements in $\mathbb{G}$. Computing quotes is very efficient as it requires only $O(\lg \ell)$ modular exponentiations for a quote of length $\ell$ for re-randomization. Similarly, verification requires only $O(\lg \ell)$ pairings.

*On Removing the Random Oracle and Obtaining Full Security* The quoting construction above is provably selectively secure in the random oracle model. We now suggest a few potential avenues for adapting the above construction to full security in the standard model. First, with an eye to remove the random oracle, we observe that our signatures share many properties with the private keys of hierarchical identity-based encryption (HIBE) schemes. To remove the random oracle, while remaining under a selective definition, one might use the Boneh-Boyen techniques [12] to instantiate $H(m) = g^m h$, where $h \in \mathbb{G}$ is added to the public key, and there is a method for mapping the message space to $\mathbb{Z}_p$. Similarly, one might remove the random oracle by instantiating $H$ with the Waters hash [60] and applying his proof techniques. This can be viewed as a full security construction with a reduction to the concrete security parameter by roughly a factor of $(1/O(q))^{\lg \ell}$, where $q$ is the number of signing queries, and $\ell$ is the length of the quote. A direction for achieving full security could be the recent "Dual System" techniques introduced by Waters [61]. One obstacle in adapting the Waters system is that it contains "tags" in the private key structure, which would likely make our re-randomization step difficult for our context-hiding property. Lewko and Waters [42] recently removed the tags, which may make their techniques and construction more suitable for our application. One drawback in using their HIBE techniques to construct signatures is that even the signatures resulting from their construction require (slightly non-standard) *decisional* complexity assumptions. Thus, it is unknown how to balance time/space efficiently while achieving full security in the standard model from a simple computational assumption such as CDH.

### 4.3. *Security Analysis*

We now provide a proof of Theorem 4.2 by showing the following lemmas.

**Lemma 4.3.** (Strong Context-Hiding) *The Sect. 4 quotable signature scheme is strongly context-hiding.*

*Proof.* Given any two challenge messages $M = (t, m)$, $M' = (t', m')$ such that $P(M, M') = 1$, we claim that whether $t' = 1$ or 0, **SignDerive**$(pk, \sigma, M', M)$ has an identical distribution to that of **Sign**$(sk, M)$, which implies that the two distributions are statistically close.

$$\begin{aligned}&\{(SK, \sigma \leftarrow \mathbf{Sign}(SK, M), \mathbf{Sign}(SK, M')\}_{SK,M,M'}\\&\{(SK, \sigma \leftarrow \mathbf{Sign}(SK, M), \mathbf{SignDerive}(PK, \sigma, M, M')\}_{SK,M,M'}\end{aligned}$$

Let $\ell, \ell'$ denote $|m|$ and $|m'|$ respectively. Let $\Gamma = \mathsf{min}(\lfloor \lg(i-1)-1 \rfloor, \lfloor \lg(\ell-i+1) \rfloor)$. **Sign**$(SK, M)$ is composed of the following values:

$$\begin{aligned}S_{i,j} &= g^{\alpha} g^{-x_{i+2^j,j-1}} H_s(m_{i,2^j})^{r_{i,j}}, & \widetilde{S_{i,j}} &= g^{r_{i,j}}, & &\text{for } i = 1 \text{ to } \ell \text{ and } j = 0 \text{ to } \lfloor \lg(\ell - i + 1) \rfloor\\A_{i,j} &= g^{x_{i,j}} g^{-x_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}}, & \widetilde{A_{i,j}} &= g^{r'_{i,j}}, & &\text{for } i = 3 \text{ to } \ell \text{ and } j = 0 \text{ to } \Gamma\\D_{i,j} &= g^{x_{i,j}} g^{-x_{i,j-1}} g^{z_j r''_{i,j}}, & \widetilde{D_{i,j}} &= g^{r''_{i,j}}, & &\text{for } i = 3 \text{ to } \ell + 1 \text{ and } j = 0 \text{ to } \lfloor \lg(i-1) - 1 \rfloor\end{aligned}$$

for randomly chosen $r_{i,j}, r'_{i,j}, r''_{i,j}, x_{i,j} \in \mathbb{Z}_p$.

**Case where** $t' = 1$ **(Type I Signatures)**. Let $\Gamma' = \mathsf{min}(\lfloor \lg(i-1)-1 \rfloor, \lfloor \lg(\ell'-i+1) \rfloor)$. When $t' = 1$, **Sign**$(SK, M')$ is composed of the following values:

$$\begin{aligned}S''_{i,j} &= g^{\alpha} g^{-x'_{i+2^j,j-1}} H_s(m'_{i,2^j})^{v_{i,j}}, & \widetilde{S''_{i,j}} &= g^{v_{i,j}}, & &\text{for } i = 1 \text{ to } \ell' \text{ and } j = 0 \text{ to } \lfloor \lg(\ell' - i + 1) \rfloor\\A''_{i,j} &= g^{x'_{i,j}} g^{-x'_{i+2^j,j-1}} H(m'_{i,2^j})^{v'_{i,j}}, & \widetilde{A''_{i,j}} &= g^{v'_{i,j}}, & &\text{for } i = 3 \text{ to } \ell' \text{ and } j = 0 \text{ to } \Gamma'\\D''_{i,j} &= g^{x'_{i,j}} g^{-x'_{i,j-1}} g^{z_j v''_{i,j}}, & \widetilde{D''_{i,j}} &= g^{v''_{i,j}}, & &\text{for } i = 3 \text{ to } \ell' + 1 \text{ and } j = 0 \text{ to } \lfloor \lg(i-1) - 1 \rfloor\end{aligned}$$

for randomly chosen $v_{i,j}, v'_{i,j}, v''_{i,j}, x'_{i,j} \in \mathbb{Z}_p$.

And **SignDerive**$(PK, \sigma, M, M')$ is Quote-Type I$(PK, \sigma, m, m')$, which is comprised of the following:

$$\begin{aligned}S'_{i,j} &= g^{\alpha} g^{-w_{i+2^j,j-1}} H_s(m'_{i,2^j})^{r_{I,j}+t_{i,j}}, & \widetilde{S'_{i,j}} &= g^{r_{I,j}+t_{i,j}}, & &\text{for } i = 1 \text{ to } \ell' \text{ and } j = 0 \text{ to } \lfloor \lg(\ell' - i + 1) \rfloor\\A'_{i,j} &= g^{w_{i,j}} g^{-w_{i+2^j,j-1}} H(m'_{i,2^j})^{r'_{I,j}+t'_{i,j}}, & \widetilde{A'_{i,j}} &= g^{r'_{I,j}+t'_{i,j}}, & &\text{for } i = 3 \text{ to } \ell' \text{ and } j = 0 \text{ to } \Gamma'\\D'_{i,j} &= g^{w_{i,j}} g^{-w_{i,j-1}} g^{z_j (r''_{I,j}+t''_{i,j})}, & \widetilde{D'_{i,j}} &= g^{r''_{I,j}+t''_{i,j}}, & &\text{for } i = 3 \text{ to } \ell' + 1 \text{ and } j = 0 \text{ to } \lfloor \lg(i-1) - 1 \rfloor\end{aligned}$$

for randomly chosen $t_{i,j}, t'_{i,j}, t''_{i,j}, y_{i,j} \in \mathbb{Z}_p$, where $m'$ occurs at position $k$ as a substring of $m$, $I = i + k - 1$ and $w_{i,j} = x_{I,j} + y_{i,j}$.

Since all exponents have been independently re-randomized, one can see by inspection that **SignDerive**$(pk, \sigma, M', M)$ has identical distribution as that of **Sign**$(sk, M')$.

**Case where $t' = 0$ (Type II Signatures).** Parse $m' = m'_\beta m'_{\beta-1} \dots m'_0$ where $m'_j$ is of length $2^j$ or a null string where $\beta = \lfloor \lg(\ell') \rfloor$. $\ell'_i$ denotes $i$-th bit of $\ell'$ when we start counting with zero as the least significant bit. $m'$ occurs at position $k$ of $m$. **Sign**$(SK, M') = (B, \widetilde{S}, Z_{\beta-1}, \dots, Z_0)$ is the following, for random $u, u_i \in \mathbb{Z}_p$:

$$B = g^\alpha \cdot H_s(m'_\beta)^u \prod_{j<\beta,\ \ell'_j=1} H(m'_j)^{u_j} \prod_{j'<\beta,\ \ell'_{j'}=0} g^{z_{j'} u_{j'}}$$

$$\tilde{S} = g^u, \quad Z_j = g^{u_j}$$

Let each $m'_j$ start at position $s_j$ in $m'$. **SignDerive**$(PK, \sigma, M, M')$ = Quote-Type II$(PK, \sigma, m, m')$ is $(B', \widetilde{S}', Z'_{\beta-1}, \dots, Z'_0)$ such that

$$B' = g^\alpha \cdot H_s(m'_\beta)^{r_{k,\beta}+v} \prod_{j<\beta,\ \ell'_j=1} H(m'_j)^{r'_{k+s_j-1,j}+v_j} \prod_{j'<\beta,\ \ell'_{j'}=0} g^{z_{j'}(r''_{k+s_{j'}-1,j'}+v_{j'})}$$

$$\tilde{S}' = g^{r_{k,\beta}+v}, \quad Z'_j = g^{r''_{k+s_j-1,j}+v_j}$$

for randomly chosen $v, v_j \in \mathbb{Z}_p$. Since all exponents have been independently re-randomized, one can see by inspection that **SignDerive**$(PK, \sigma, M, M')$ has identical distribution as that of **Sign**$(sk, M')$.

Thus, the powers-of-2 construction is strongly context-hiding. □

**Lemma 4.4.** (Unforgeability) *If the CDH assumption holds in $\mathbb{G}$, then the Sect. 4 quotable signature scheme is selectively unforgeable in the* **Unforg** *game in the random oracle model.*

*Proof.* We first apply Lemma 2.8, which allows us to only consider adversaries $\mathcal{A}$ that asks queries to *Sign* oracle in the simpler **NHU** game.

Suppose an adversary $\mathcal{A}$ can produce a forgery with probability $\epsilon$ in the selective **NHU** unforgeability game; then we can construct an adversary $\mathcal{B}$ that breaks the CDH assumption with probability $\epsilon$ plus a negligible amount.

We are now ready to describe $\mathcal{B}$ which solves the CDH problem. On input the CDH challenge $(g, g^a, g^b)$, $\mathcal{B}$ begins to run $\mathcal{A}$ and proceeds as follows:
*Selective Disclosure* $\mathcal{A}$ first announces the message $M^*$ on which he will forge.
*Setup* Let $L$ be the maximum size of any message and let $n = \lfloor \lg(L) \rfloor$. Let $M^* = (t^*, m^*)$ and $\ell^* = |m^*|$ and let $\beta$ be the highest bit of $\ell^*$ set to 1 (numbering the least significant bit as zero). Set $\mathbf{e}(g, g)^\alpha := \mathbf{e}(g^a, g^b)$, which implicitly sets the secret key $\alpha = ab$.

For $i = 0$ to $n - 1$, choose a random $v_i \in \mathbb{Z}_p$ and set

$$g^{z_i} = \begin{cases} g^{bv_i} & \text{if the } i\text{th bit of } \ell^* \text{ is } 1; \\ g^{v_i} & \text{otherwise.} \end{cases}$$

Finally, $\mathcal{B}$ gives the public key $PK = (g, g^{z_0}, \ldots, g^{z_{n-1}}, \mathbf{e}(g, g)^{\alpha})$ to $\mathcal{A}$ and will answer its queries to random oracles $H$ and $H_s$ interactively as described below.

*Random Oracle Queries* Proceeding adaptively, $\mathcal{A}$ may make any of the following queries which $\mathcal{B}$ will answer as follows:

1. $H(x)$: The random oracle is answered as follows. If the query has been made before, return the same response as before. Otherwise, imagine dividing up $m^*$ into a sequence of segments whose lengths are decreasing powers of two; that is, the first segments would be of length $2^\beta$ where $\beta$ is the largest power of two less than $\ell^*$, the second segment would contain the next largest power of two, etc. Let $m^*_{(j)}$ denote the segment of $m^*$ corresponding to power $j$. If no such segment exists, let $m^*_{(j)} = \perp$. Select a random $\gamma \in \mathbb{Z}_p$ and return the response as:

$$H(x) = \begin{cases} g^{\gamma} & \begin{array}{l}\text{if } |x| = 2^j \text{ and } j < \beta \text{ and } m^*_{(j)} = x \\ (x \text{ is on the selective path});\end{array} \\ g^{b\gamma} & \begin{array}{l}\text{otherwise} \\ (x \text{ is not on the selective path}).\end{array} \end{cases}$$

   Note that $H(m^*_{(j)})$ is set according to the first method for all segments of $m^*$ *except* the first segment $m^*_{(\beta)}$.
2. $H_s(x)$: The random oracle is answered as follows. If the query has been made before, then return the same response as before. Select a random $\delta \in \mathbb{Z}_p$ and return the response as:

$$H_s(x) = \begin{cases} g^{\delta} & \text{if } |x| = 2^\beta \text{ and } m^*_{(\beta)} = x; \\ g^{b\delta} & \text{otherwise.} \end{cases}$$

   Note that $H_s(m^*_{(j)})$ is set according to the first method *only* for the first segment of $m^*$.

*Signature and Quote Queries*
**Sign** $(M)$: Let $M = (t, m)$ and $\ell = |m|$. Recall that $\beta^*$ is the highest bit of $\ell^*$ set to 1 and that we are counting up from zero as the least significant bit.

We describe how to create signatures.

1. When $t = 1$ and $m^*$ is not a substring of $m$ (Type I Signature Generation):
   Here $m_{i,j}$ denotes the substring $m$ of length $j$ starting at position $i$. It will help us to first establish the variables $X_{i,j}$, which will be set to 1 if on the selective forgery path and 0 otherwise. We give a set of "rules" defining terms and make

a few observations. Then we describe how the reduction algorithm creates the signatures.

*Rules*
For $i = 1$ up to $\ell + 1$,
For $j = \lfloor \lg(\ell - i + 1) \rfloor$ down to $-1$,

(a) If $j + 1 = \beta^*$ and $m_{i-2^{j+1},2^{j+1}} = m^*_{(j+1)}$, then set $X_{i,j} = 1$.
(b) Else, if $j + 1 < \beta^*$ and $(j + 1)$th bit of $\ell^*$ is 1 and $m_{i-2^{j+1},2^{j+1}} = m^*_{(j+1)}$ and $X_{i-2^{j+1},j+1} = 1$, then set $X_{i,j} = 1$.
(c) Else if $j + 1 < \beta^*$ and $(j + 1)$th bit of $\ell^*$ is 0 and $X_{i,j+1} = 1$, then set $X_{i,j} = 1$.
(d) Else set $X_{i,j} = 0$.

*Observations* Before we show how $\mathcal{B}$ will simulate the signatures, we make a set of useful observations.

(a) For all $i$ and $j \geq \beta^*$, $X_{i,j} = 0$.
(b) For all $i$, $X_{i,-1} = 0$. Otherwise, $m_{i-\ell^*,\ell^*} = m^*$.
(c) For all $i, j$, if $X_{i,j} = 1$ and $X_{i,j-1} = 0$, then the $j$th bit of $\ell^*$ is 1. If the $j$th bit were 0, then $X_{i,j-1}$ would have been set to 1 by Rule 1c.
(d) For all $i, j$, if $X_{i,j} = 0$ and $X_{i,j-1} = 1$, then the $j$th bit of $\ell^*$ is 1. If the $j$th bit were 0, then the only way to set $X_{i,j-1}$ to 1 would be by Rule 1c, however, $X_{i,j} = 0$ so Rule 1c does not apply.
(e) For all $i, j$, if $X_{i,j} = 1$ and $X_{i+2^j,j-1} = 0$, then $H(m_{i,2^j}) = g^{b\gamma}$ for some known $\gamma \in \mathbb{Z}_p$. Otherwise, $X_{i+2^j,j-1}$ would have been set by Rule 1b to be 1.
(f) For all $i, j$, if $X_{i,j} = 0$ and $X_{i+2^j,j-1} = 1$, then $H(m_{i,2^j}) = g^{b\gamma}$ for some known $\gamma \in \mathbb{Z}_p$. If $X_{i+2^j,j-1} = 1$ and $X_{i,j} = 0$, then $X_{i+2^j,j-1}$ was set to be 1 either by Rule 1a or Rule 1c. If it were Rule 1a, then $j = \beta^*$ and it follows from the programming of the random oracle that $H(m_{i,2^j}) = g^{b\gamma}$. If it were Rule 1c, then the $j$th bit of $\ell^*$ is 0, meaning $m_{(j)}$ cannot be on the selective path and therefore again $H(m_{i,2^j}) = g^{b\gamma}$.
(g) For all $i, j$, if $X_{i+2^j,j-1} = 0$, then $H_s(m_{i,2^j}) = g^{b\delta}$ for some known $\delta \in \mathbb{Z}_p$. If $j \neq \beta^*$, this follows immediately from the programming of the random oracle. Otherwise, if $j = \beta^*$, then the only way for $X_{i+2^j,j-1} = 0$ would be if $m_{(\beta)} \neq m^*_{(\beta)}$ by Rule 1a. Thus, it also follows that $H_s(m_{i,2^j}) = g^{b\delta}$.

*Signature Components* Next, for $i = 1$ to $\ell + 1$ and $j = 0$ to $\lfloor \lg(\ell - i + 1) \rfloor$, choose a random $x'_{i,j} \in \mathbb{Z}_p$ and logically set $x_{i,j} := x'_{i,j} + X_{i,j} \cdot (ab)$. For $i = 1$ to $\ell + 1$, set $x_{i,-1} := 0$ (as consistent with Observation 1b).
A signature is comprised of the following values:
*Start* For $i = 1$ to $\ell$ and $j = 0$ to $\lfloor \lg(\ell - i + 1) \rfloor$:

(a) If $X_{i+2^j,j-1} = 0$, then it follows by Observation 1g that $H_s(m_{i,2^j}) = g^{b\delta}$ for some known $\delta \in \mathbb{Z}_p$, so choose random $s_{i,j} \in \mathbb{Z}_p$, implicitly set $r_{i,j} := -a/\delta + s_{i,j}$ and set

$$\begin{aligned} S_{i,j} &= g^{-x_{i+2^j,j-1}} g^{b\delta s_{i,j}} \\ &= g^{\alpha} g^{-x_{i+2^j,j-1}} H_s(m_{i,2^j})^{r_{i,j}} \\ \widetilde{S_{i,j}} &= g^{-a/\delta + s_{i,j}} = g^{r_{i,j}}. \end{aligned}$$

(b) Else $X_{i+2^j,j-1} = 1$, so choose random $r_{i,j} \in \mathbb{Z}_p$ and with $x_{i+2^j,j-1} := x'_{i+2^j,j-1} + ab$ set

$$\begin{aligned} S_{i,j} &= g^{-x'_{i+2^j,j-1}} H_s(m_{i,2^j})^{r_{i,j}} \\ &= g^{\alpha} g^{-x_{i+2^j,j-1}} H_s(m_{i,2^j})^{r_{i,j}} \\ \widetilde{S_{i,j}} &= g^{r_{i,j}}. \end{aligned}$$

*Across* Together with the following values for $i = 3$ to $\ell$ and $j = 0$ to $\min(\lfloor \lg(i-1) - 1 \rfloor, \lfloor \lg(\ell - i + 1) \rfloor)$:

(a) If $X_{i,j} = 1$ and $X_{i+2^j,j-1} = 1$, choose random $r'_{i,j} \in \mathbb{Z}_p$ with implicitly set $x_{i,j} = x'_{i,j} + ab$ and $x_{i+2^j,j-1} = x'_{i+2^j,j-1} + ab$ and set

$$\begin{aligned} A_{i,j} &= g^{x'_{i,j}} g^{-x'_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}} \\ &= g^{x_{i,j}} g^{-x_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}} \\ \widetilde{A_{i,j}} &= g^{r'_{i,j}}. \end{aligned}$$

(b) Else, if $X_{i,j} = 1$ and $X_{i+2^j,j-1} = 0$, then $H(m_{i,2^j}) = g^{b\gamma}$ for some known $\gamma \in \mathbb{Z}_p$ by Observation 1e. Choose random $s'_{i,j} \in \mathbb{Z}_p$ with implicitly set $x_{i,j} = x'_{i,j} + ab$, $x_{i+2^j,j-1} = x'_{i+2^j,j-1}$ and $r'_{i,j} := -a/\gamma + s'_{i,j}$ and set

$$\begin{aligned} A_{i,j} &= g^{x'_{i,j}} g^{-x_{i+2^j,j-1}} g^{b\gamma s'_{i,j}} \\ &= g^{x_{i,j}} g^{-x_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}} \\ \widetilde{A_{i,j}} &= g^{r'_{i,j}}. \end{aligned}$$

(c) Else, if $X_{i,j} = 0$ and $X_{i+2^j,j-1} = 1$, then $H(m_{i,2^j}) = g^{b\gamma}$ for some known $\gamma \in \mathbb{Z}_p$ by Observation 1f. Choose random $s'_{i,j} \in \mathbb{Z}_p$ with implicitly set $x_{i,j} = x'_{i,j}$, $x_{i+2^j,j-1} = x'_{i+2^j,j-1} + ab$ and $r'_{i,j} := a/\gamma + s'_{i,j}$ and set

$$\begin{aligned} A_{i,j} &= g^{x_{i,j}} g^{-x'_{i+2^j,j-1}} g^{b\gamma s'_{i,j}} \\ &= g^{x_{i,j}} g^{-x_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}} \\ \widetilde{A_{i,j}} &= g^{r'_{i,j}}. \end{aligned}$$

(d) Else, $X_{i,j} = 0$ and $X_{i+2^j,j-1} = 0$, so choose random $r'_{i,j} \in \mathbb{Z}_p$ and set

$$A_{i,j} = g^{x_{i,j}} g^{-x_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}}, \quad \widetilde{A_{i,j}} = g^{r'_{i,j}}.$$

*Down* Together with the following values for $i = 3$ to $\ell + 1$ and $j = 0$ to $\lfloor \lg(i-1) - 1 \rfloor$:

(a) If $X_{i,j} = 1$ and $X_{i,j-1} = 1$, choose random $r''_{i,j} \in \mathbb{Z}_p$ with implicitly set $x_{i,j} = x'_{i,j} + ab$ and $x_{i,j-1} = x'_{i,j-1} + ab$ and set

$$D_{i,j} = g^{x'_{i,j}} g^{-x'_{i,j-1}} g^{z_j r''_{i,j}} = g^{x_{i,j}} g^{-x_{i,j-1}} g^{z_j r''_{i,j}}$$
$$\widetilde{D_{i,j}} = g^{r''_{i,j}}.$$

(b) Else, if $X_{i,j} = 1$ and $X_{i,j-1} = 0$, then the $j$th bit of $\ell^*$ is 1 by Observation 1c. Thus $z_j = bv_j$, so choose random $s''_{i,j} \in \mathbb{Z}_p$ with implicitly set $x_{i,j} = x'_{i,j} + ab$, $x_{i,j-1} = x'_{i,j-1}$ and $r''_{i,j} := -a/v_j + s''_{i,j}$ and set

$$D_{i,j} = g^{x'_{i,j}} g^{-x_{i,j-1}} g^{bv_j s''_{i,j}} = g^{x_{i,j}} g^{-x_{i,j-1}} g^{z_j r''_{i,j}}$$
$$\widetilde{D_{i,j}} = g^{-a/v_j + s''_{i,j}} = g^{r''_{i,j}}.$$

(c) Else, if $X_{i,j} = 0$ and $X_{i,j-1} = 1$, then the $j$th bit of $\ell^*$ is 1 by Observation 1d. Thus $z_j = bv_j$, so choose random $s''_{i,j} \in \mathbb{Z}_p$ with implicitly set $x_{i,j} = x'_{i,j}$, $x_{i,j-1} = x'_{i,j-1} + ab$ and $r''_{i,j} := a/v_j + s''_{i,j}$ and set

$$D_{i,j} = g^{x'_{i,j}} g^{-x_{i,j-1}} g^{bv_j s''_{i,j}} = g^{x_{i,j}} g^{-x_{i,j-1}} g^{z_j r''_{i,j}}$$
$$\widetilde{D_{i,j}} = g^{a/v_j + s''_{i,j}} = g^{r''_{i,j}}.$$

(d) Else, $X_{i,j} = 0$ and $X_{i,j-1} = 0$, so choose random $r''_{i,j} \in \mathbb{Z}_p$ and set

$$D_{i,j} = g^{x_{i,j}} g^{-x_{i,j-1}} g^{z_j r''_{i,j}}, \quad \widetilde{D_{i,j}} = g^{r''_{i,j}}.$$

2. When $t = 0$ and $m \neq m^*$ (Type II Signature Generation):
Let $\ell = |m|$, and $\beta = \lfloor \lg(\ell) \rfloor$. $\ell^*_i$ denotes the $i$-th bit of $\ell^*$ when we start counting with zero as the least significant bit, and $\ell_i$ denotes $i$-th bit of $\ell$.
Parse $m^*$ as $m^*_{\beta^*} m^*_{\beta^*-1} \ldots m^*_0$ where $m^*_i$ is a string of length $2^i$ or a null string. $m_i$ is of length $2^i$ if $\ell_i = 0$, and is null otherwise. Similarly, parse $m$ as $m_\beta m_{\beta-1} \ldots m_0$.

$\mathcal{B}$ constructs $(B, \tilde{S}, Z_{\beta-1}, \ldots, Z_0)$ in the following way:

- If $m_\beta \neq m^*_{\beta^*}$, then $H_s(m_\beta) = g^{b\delta}$ for a $\delta$ which is known to $\mathcal{B}$.
  (a) $\mathcal{B}$ sets $\tilde{S} := g^{-a/\delta + r}$ for a randomly chosen $r$ and $B := g^{b\delta r}$.
  (b) For $j = \beta - 1$ down to 0, $Z_j := g^{r_j}$ for a randomly chosen $r_j$, and
    – If $\ell_j = 1$, then $B := B \cdot H(m_j)^{r_j}$.
    – If $\ell_j = 0$, then $B := B \cdot g^{z_j r_j}$.
- Otherwise, if $\beta = \beta^*$ and $m_\beta = m^*_{\beta^*}$, there exists $j_s < \beta$ such that
  – $\ell_{j_s} \neq \ell^*_{j_s}$, or
  – $\ell_{j_s} = \ell^*_{j_s} = 1$ and $H(m_{j_s}) \neq H(m^*_{j_s})$.

so $\mathcal{B}$ can construct a signature $(B, \tilde{S}, Z_{\beta-1}, \ldots, Z_0)$ in the following way.

(a) $\mathcal{B}$ sets $\tilde{S} := g^{r_c}$ for a randomly chosen $r_c$ and $B := g^{\delta r_c}$.
(b) For $j = \beta - 1$ down to $j_s + 1$ and $j = j_s - 1$ to 0, $Z_j := g^{r_j}$ for randomly chosen $r_j$, and
   - If $\ell_j = 1$, then $B := B \cdot H(m_j)^{r_j}$.
   - If $\ell_j = 0$, then $B := B \cdot g^{z_j r_j}$.

(c) For $j = j_s$,
   - If $\ell_j = 1$, whether $\ell_j^* = 0$ or not, $\mathcal{B}$ knows $\gamma$ such that $H(m_j) = g^{b\gamma}$. $\mathcal{B}$ sets $Z_j = g^{-a/\gamma + r_j}$ for a randomly chosen $r_j$, and $B := B \cdot g^{b\gamma r_j}$.
   - If $\ell_j = 0$ and $\ell_j^* = 1$, then $\mathcal{B}$ knows $v$ such that $g^{z_j} = g^{bv}$. $\mathcal{B}$ sets $Z_j = g^{-a/v + r_j}$ for a randomly chosen $r_j$, and $B := B \cdot g^{bvr_j}$.

$\mathcal{B}$ returns $(B, \tilde{S}, Z_{\beta-1}, \ldots, Z_0)$.
*Response* Eventually, $\mathcal{A}$ outputs a valid signature $\sigma^*$ on $M^* = (t^*, m^*)$. Recall that $\ell^* = |m^*|$ and $\beta = \lfloor \lg(\ell^*) \rfloor$. Here $\ell_i^*$ denotes $i$-th bit of $\ell^*$ when we start counting with zero as the least significant bit. Parse $m^*$ as $m_\beta^* m_{\beta-1}^* \ldots m_0^*$ where $m_i^*$ is a string of length $2^i$ (when $\ell_i^* = 1$) or a null string (when $\ell_i^* = 0$).

Because of the selective disclosure and setup, $\mathcal{B}$ knows the following exponents:

- $\gamma$ such that $H_s(m_\beta^*) = g^\gamma$.
- $\delta_j$ such that $H(m^*_{s_j, 2^j}) = g^{\delta_j}$ when $\ell_j^* = 1$ and $j \neq \beta$.
- $z_j$ when $\ell_j^* = 0$.

$t^*$ is either 1 or 0.

- If $t^* = 1$,
  $s_i$ denotes the position where $m_i^*$ starts. $\mathcal{B}$ can compute the information of some $x_{i,j}$ with the following components of $\sigma^*$.
  - $S_{1,\beta} = g^\alpha g^{-x_{1+2^\beta, \beta-1}} H_s(m_\beta^*)^{r_c}$, $\widetilde{S_{1,\beta}} = g^{r_{1,\beta}}$
    $\mathcal{B}$ knows $\gamma$ such that $H_s(m_\beta^*) = g^\gamma$, so $\mathcal{B}$ can compute $g^\alpha g^{-x_{1+2^\beta, \beta-1}} = S_{1,\beta} / \widetilde{S_{1,\beta}}^\gamma$.
  - For $j = \beta - 1$ down to 0,
    * when $\ell_j = 1$, $A_{s_j, j} = g^{x_{s_j, j}} g^{-x_{s_{j-1}, j-1}} H(m_j^*)^{r'_{s_j, j}}$, $\widetilde{A_{s_j, j}} = g^{r'_{s_j, j}}$
      $\mathcal{B}$ knows $\delta$ such that $H(m_j^*) = g^\delta$, so $\mathcal{B}$ can compute $g^{x_{s_j, j}} g^{-x_{s_{j-1}, j-1}} = A_{s_j, j} / \widetilde{A_{s_j, j}}^\delta$.
    * when $\ell_j = 0$, $D_{s_j, j} = g^{x_{s_j, j}} g^{-x_{s_{j-1}, j-1}} g^{z_j r''_{s_j, j}}$, $\widetilde{D_{s_j, j}} = g^{r''_{s_j, j}}$
      $\mathcal{B}$ knows $z_j$, so $\mathcal{B}$ can compute $g^{x_{s_j, j}} g^{-x_{s_{j-1}, j-1}} = D_{s_j, j} / \widetilde{D_{s_j, j}}^{z_j}$.

  so $\mathcal{B}$ can compute $g^{x_{s_j, j}} g^{-x_{s_{j-1}, j-1}}$.

$\mathcal{B}$ has the values of $g^{x_{s_j, j}} g^{-x_{s_{j-1}, j-1}}$ for $j = \beta - 1$ down to 0 and $g^\alpha g^{-x_{1+2^\beta, \beta-1}}$, so can compute

$$g^\alpha g^{-x_{1+2^\beta, \beta-1}} \prod_{j=0}^{\beta-1} g^{x_{s_j, j}} g^{-x_{s_{j-1}, j-1}} = g^\alpha g^{-x_{s_{-1}, -1}} = g^\alpha$$

- If $t^* = 0$,
  $\mathcal{B}$ parses $\sigma^*$ as $(B, \widetilde{S}, Z_{\beta-1}, \ldots, Z_0)$, with

$$\widetilde{S} = g^c, \quad Z_{\beta-1} = g^{c_{\beta-1}}, \quad \ldots, \quad Z_0 = g^{c_0}$$

  for some $c, c_{\beta-1}, \ldots, c_0 \in \mathbb{Z}_p$.

$$B = g^{\alpha} \cdot H_s(m^*_{\beta})^c \prod_{j<\beta,\ \ell^*_j=1} H(m^*_j)^{c_j} \prod_{j'<\beta,\ \ell^*_{j'}=0} (g^{z_{j'}})^{c_{j'}}$$

  because the signature is valid.
  - $\mathcal{B}$ knows $\gamma$ such that $H_s(m^*_{\beta}) = g^{\gamma}$. $\mathcal{B}$ sets $C := \widetilde{S}^{\gamma}$.
  - From $j = \beta - 1$ down to 0, $\mathcal{B}$ proceeds as:
    - If $\ell_j = 1$, $\mathcal{B}$ knows $\delta_j$ such that $H(m^*_j) = g^{\delta_j}$. $\mathcal{B}$ sets $C := C \cdot Z_j^{\delta_j}$ ;
    - If $\ell_j = 0$, $\mathcal{B}$ knows $z_j$. $\mathcal{B}$ sets $C := C \cdot Z_j^{z_j}$.

  Then

$$C = H_s(m^*_{\beta})^c \prod_{j<\beta,\ \ell^*_j=1} H(m^*_j)^{c_j} \prod_{j'<\beta,\ \ell^*_{j'}=0} (g^{z_{j'}})^{c_{j'}}$$

  so $\mathcal{B}$ can compute $B/C = g^{\alpha}$.

Thus, whether $t^*$ is 1 or 0, $\mathcal{B}$ can solve for $g^{\alpha} = g^{ab}$ and correctly answer to the CDH challenge.

*Analysis* The distribution of the above game and the security game are identical. Thus, whenever $\mathcal{A}$ is successful in a forgery against our scheme, $\mathcal{B}$ will solve the CDH challenge. □

## 5. A Construction for Subset Predicates based on ABE

*The Subset Predicate* We now point out a surprising connection to Attribute-Based Encryption (ABE). We show that existing constructions for Ciphertext-Policy ABE [10, 41,62] naturally lead to context hiding quotable signatures for arbitrary message *subsets* (as opposed to the *substring* predicate considered in the previous section). In particular, let $U$ be a set of strings over an arbitrary alphabet. These strings can be used to encode elements for different types of sets. A message will be a set of strings from $U$. A general way to define the subset predicate would be $P(M, m') = 1$ iff $m' \subseteq m_i$ for some $m_i \in M$. Recall from Sect. 2 that $M$ is a set of messages, which might have been independently authenticated. Here, we want to disallow "collusions" between two different signatures where $m'$ is a subset of the union of multiple messages in $M$, but not any single one. (Otherwise, this would be trivially realizable from standard signatures schemes). In other words, our focus here is extracting a subset from a *single* signed set. Thus, we will restrict our attention in this section to the simple predicate $P(m, m') = 1$ iff $m' \subseteq m$.

*The Construction at a High-Level* Our main tool is an observation of Naor that shows that secret keys in Identity-Based Encryption [14] can function as signatures. Recall that in (ciphertext-policy) *attribute-based encryption* an authority provides secret keys to a user based on the user's list of attributes. The main challenge in building such systems is preventing collusion attacks: two (or more) users with distinct sets of attributes should be unable to create a secret key for a combination of their attributes.

If we treat elements in a message $m \subseteq U$ as attributes, that is, we treat a message $m = \{a_1, \ldots, a_\ell\} \in U^\ell$ as a set of attributes $a_1, \ldots, a_\ell$, then we can define the signature on $m$ as a set of $\ell$ secret keys corresponding to the $\ell$ attributes in the message. Verifying the signature can be done by trying to decrypt some test ciphertext using the secret keys in the signature. Now, given a signature on $m$ we derive a signature on a subset of the elements in $m$ by simply removing the secret keys corresponding to elements not in the subset. For context hiding, we need to re-randomize the resulting set of secret keys. (Not all CP-ABE schemes may support the removal and re-randomization of secret keys in this manner, but the schemes of [10,41,62] do).

Since ABE security prevents collusion attacks, it is straight forward to show that these signatures are unforgeable in the sense of Definition 2.3. Moreover, due to the re-randomization of secret keys, a derived signature is sampled from the same distribution as a fresh signature and is independent from the given signature. This implies strong context hiding in sense of Definition 2.4.

This unexpected connection between quoting and ABE leads to the following theorem, stated first informally.

**Theorem 5.1.** (Informal) *The Ciphertext-Policy ABE systems in [10,41,62] translate using Naor's transformation into a signature scheme supporting quoting for arbitrary subsets of a message. (Selective) security of the CP-ABE systems imply (selective) unforgeability and context hiding.*

In other words, when the ABE scheme provides adaptive (resp, selective) security, then the resulting signature scheme achieves adaptive (resp., selective) unforgeability. The (third) ABE scheme of Waters [62] provides selective security from the Decisional Bilinear Diffie Hellman assumption. Adaptive security is proven for the Bethencourt et al. construction [10], but only in the generic group model. The construction of Lewko et al. [41] proves adaptive security under certain static assumptions using composite order groups.

## 5.1. *The Subset Construction from Existing CP-ABE Schemes*

We now formalize the intuition and claims of the previous section.

### 5.1.1. *Background: Ciphertext-Policy ABE*

**Definition 1.** (*Access Structure* [5]) Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\Gamma \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$ : if $B \in \Gamma$ and $B \subseteq C$ then $C \in \Gamma$. An *access* *structure* (respectively, monotone access structure) is a collection (resp., monotone collection) $\Gamma$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\Gamma \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \backslash \{\emptyset\}$. The sets in $\Gamma$ are called the *authorized sets*, and the sets not in $\Gamma$ are called the *unauthorized sets*.

In the context of CP-ABE, the role of the parties is taken by the attributes. Thus, the access structure $\Gamma$ will contain the authorized sets of attributes. We restrict our attention to monotone access structures.

**Definition 5.2.** (*CP-ABE Algorithm Specification*) A ciphertext-policy attribute-based encryption system for message space $\mathcal{M}$ and access structure space $\mathcal{G}$ is a tuple of the following algorithms:

**Setup**$(\lambda, U) \to$ (PK, MK). The setup algorithm takes as input a security parameter $\lambda$ and a universe description $U$, which defines the set of allowed attributes in the system. It outputs the public parameters PK and the master secret key MK.
**Encrypt**(PK, $m$, $\Gamma$) $\to$ CT. The encryption algorithm takes as input the public parameters PK, a message $m$ and an access structure $\Gamma$ and outputs a ciphertext CT associated with the access structure.
**KeyGen**(MK, $S$) $\to$ *sk*. The key generation algorithm takes as input the master secret key MK and a set of attributes $S$ and outputs a private key *sk* associated with the attributes.
**Decrypt**(*sk*, CT) $\to$ $m$. The decryption algorithm takes as input a secret key *sk* associated with attributes $S$ and a ciphertext CT associated with access structure $\Gamma$ and outputs a message $m$ if $S$ satisfies $\Gamma$ or the error message $\perp$ otherwise.

The correctness property requires that for all sufficiently large $\lambda \in \mathbb{N}$, all universe descriptions $U$, all (PK, MK) $\in$ **Setup**$(\lambda, U)$, all $S \subseteq U$, all *sk* $\in$ **KeyGen**(MK, $S$), all $m \in \mathcal{M}$, all $\Gamma \in \mathcal{G}$ and all CT $\in$ **Encrypt**(PK, $m$, $\Gamma$), if $S$ satisfies $\Gamma$, then **Decrypt**(*sk*, CT) outputs $m$.

*Security Model for CP-ABE* Let $\Pi$ = (**Setup**, **Encrypt**, **KeyGen**, **Decrypt**) be a CP-ABE scheme for message space $\mathcal{M}$ and access structure space $\mathcal{G}$, and consider the following experiment for an adversary **Adv**, parameter $\lambda$, and attribute universe $U$:
*The CP-ABE experiment* $\mathsf{CP\text{-}ABE\text{-}Exp}_{\mathbf{Adv},\Pi}(\lambda, U)$:

**Start.** **Setup**$(\lambda, U)$ is run to obtain the public parameters PK and master secret key MK.
**Phase 1.** Adversary **Adv** is given PK and access to the oracle **KeyGen**(MK, $\cdot$), which generates a private key corresponding to an attribute set of the adversary's choosing.
**Challenge.** The adversary outputs two messages $m_0, m_1 \in \mathcal{M}$ and a challenge access structure $\Gamma^*$ such that none of the sets of attributes queried during Phase 1 satisfy it. A random bit $b$ is chosen and **Encrypt**(PK, $m_b$, $\Gamma^*$) is run to produce CT$^*$, which is then given to the adversary.
**Phase 2.** The adversary is given access to the oracle **KeyGen**(MK, $\cdot$), with the restriction that it cannot query the oracle on any set of attributes that satisfy $\Gamma^*$.
**Guess.** The adversary outputs a guess $b'$ of $b$. The output of the experiment is defined to be 1 if and only if $b' = b$.

**Definition 5.3.** (*CP-ABE Security*) A CP-ABE scheme $\Pi$ is secure for attribute universe $U$ if for all probabilistic polynomial-time adversaries **Adv**, there exists a negligible function *negl* such that:

$$\Pr[\textsf{CP-ABE-Exp}_{\mathbf{Adv},\Pi}(\lambda, U) = 1] \leq negl(\lambda).$$

We say that a system is *selectively* secure if we add an Init stage before Start where the adversary outputs the challenge access structure $\Gamma^*$ (instead of waiting until Challenge to do so).

5.1.2. *CP-ABE with Key Reduction*

Our construction requires that the holder of a private key can efficiently "remove" attributes from his private key and then re-randomize the remaining private key. We formalize this as follows.

**Definition 5.4.** (*CP-ABE with key reduction*) We say that a CP-ABE system for attribute universe $U$ supports *key reduction* if there exists an efficient algorithm

> **KeyReduce**(PK, $sk$, $S$, $S'$) $\rightarrow$ $sk'$. The key reduction algorithm takes as input the public parameters PK with a private key $sk$ associated with attribute set $S$ and outputs a private key $sk'$ associated with attribute set $S'$, if $S' \subseteq S$, and $\perp$ otherwise.

such that if (PK, MK) $\in$ **Setup**($\lambda$, $U$) and $S' \subseteq S \subseteq U$, then for all such tuples (MK, $S$, $S'$), the following two distributions are statistically close:

$$\big\{\big(\mathrm{MK}, sk \leftarrow \mathbf{KeyGen}(\mathrm{MK}, S), \ \mathbf{KeyGen}(\mathrm{MK}, S')\big)\big\}_{\mathrm{MK},S,S'}$$
$$\big\{\big(\mathrm{MK}, sk \leftarrow \mathbf{KeyGen}(\mathrm{MK}, S), \ \mathbf{KeyReduce}(\mathrm{PK}, sk, S, S')\big)\big\}_{\mathrm{MK},S,S'}$$

The distributions are taken over the coins of **KeyGen** and **KeyReduce**.

It is not a coincidence that this definition strongly resembles the context-hiding definition presented earlier. Fortunately, we observed that several existing CP-ABE schemes support key reduction. In fact, we are not aware of any prior bilinear map-based schemes that do not support key reduction, although it is possible to construct contrived counterexamples[10] and it seems plausible that such schemes could naturally arise in the bilinear, lattice, or other settings.

**Claim 5.5.** *The Ciphertext-Policy ABE systems in [10, Sect. 4.2], [41, Sect. 2.3.1], [62, Sect. 6] support key reduction.*

*Proof.* We argue this claim by providing a key reduction algorithm for each scheme. In all cases, the output is perfectly indistinguishable from the normal key generation algorithm.

[10] One can build a contrived example that does not support key reduction. For instance, suppose we took an existing CP-ABE scheme and added a standard signature (from the authority) on the set $S$ of attributes associated with the key. Then add to the decryption algorithm a check for the existence of a valid signature on the key and that the key matches this signature before proceeding with decryption.

*The BSW Construction* [10, Sect. 4.2]

- **Setup**$(\lambda, U) \rightarrow$ (PK, MK): The algebraic setting is a bilinear group $\mathbb{G}$ of prime order $p$ with generator $g$. The public parameters PK are $\mathbb{G}, g, p, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha}$, where $\beta, \alpha \in \mathbb{Z}_p$, and the description of a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$. The master secret key MK is $(\text{PK}, \beta, g^{\alpha})$.
- **KeyGen**(MK, $S$) $\rightarrow$ $sk$: The key generation algorithm chooses random $r, r_i \in \mathbb{Z}_p$ for each attribute $i \in S$. The private key $sk$ is:

$$S,\ D = g^{(\alpha+r)/\beta},\ D_j = g^r H(j)^{r_j}, D'_j = g^{r_j}\ \forall j \in S.$$

- **KeyReduce**(PK, $sk$, $S$, $S'$) $\rightarrow$ $sk'$: The key reduction algorithm chooses random $r', r'_i$ for each attribute $i \in S'$ and outputs the private key $sk'$ as:

$$S',\ D' = Dg^{r'/\beta} = g^{(\alpha+r+r')/\beta},$$
$$D'_j = D_j g^{r'} H(j)^{r'_j} = g^{r+r'} H(j)^{r_j+r'_j}, D''_j = D_j g^{r'_j} = g^{r_j+r'_j}\ \forall j \in S'.$$

*The LOSTW Construction* [41, Sect. 2.3.1]

- **Setup**$(\lambda, U) \rightarrow$ (PK, MK): The algebraic setting is a bilinear group $\mathbb{G}$ of order $N = p_1 p_2 p_3$ (3 distinct primes). We let $\mathbb{G}_{p_i}$ denote the subgroup of order $p_i$ in $\mathbb{G}$. The public parameters PK are $N, g, g^a, e(g,g)^{\alpha}, T_i = g^{s_i}$ for all attributes $i \in U$, where $g \in \mathbb{G}_{p_1}$ and $a, \alpha, s_i \in \mathbb{Z}_N$. The master secret key MK is PK, $\alpha$ and a generator $X_3 \in \mathbb{G}_{p_3}$.
- **KeyGen**(MK, $S$) $\rightarrow$ $sk$: The key generation algorithm chooses a random $t \in \mathbb{Z}_N$ and random elements $R_0, R'_0, R_i \in G_{p_3}$. The private key $sk$ is:

$$S,\ K = g^{\alpha} g^{at} R_0,\ L = g^t R'_0,\ K_i = T_i^t R_i\ \forall i \in S.$$

- **KeyReduce**(PK, $sk$, $S$, $S'$) $\rightarrow$ $sk'$: The key reduction algorithm chooses a random $t' \in \mathbb{Z}_N$ and random elements $Z_0, Z'_0, Z_i \in G_{p_3}$ and outputs the new private key $sk'$ as:

$$S',\ K' = Kg^{at'} Z_0 = g^{\alpha} g^{a(t+t')} R_0 Z_0,\ L' = Lg^{t'} Z'_0 = g^{t+t'} R'_0 Z'_0,$$
$$K'_i = K_i T_i^{t'} Z'_i = T_i^{t+t'} R_i Z_i\ \forall i \in S'.$$

*The Waters Construction* [62, Sect. 6]

- **Setup**$(\lambda, U) \rightarrow$ (PK, MK): The algebraic setting is a bilinear group $\mathbb{G}$ of prime order $p$ with generator $g$. Let $n_m ax$ be the maximum number of nodes in an access formula and let $|U|$ be the number of attributes in $U$. The public parameters PK are $\mathbb{G}, g, p, g^a, e(g,g)^{\alpha}, (h_{1,1}, \ldots, h_{1,U}), \ldots, (h_{n_{max},1}, \ldots, h_{n_{max},U})$, where all $h_{i,j}$ values are elements in $\mathbb{G}$. The master secret key MK is $(\text{PK}, g^{\alpha})$.

- **KeyGen**(MK, $S$) $\rightarrow sk$: The key generation algorithm chooses random $t_1, \ldots, t_{n_{max}} \in \mathbb{Z}_p$. The private key $sk$ is:

$$S,\ K = g^{\alpha} g^{a t_1},\ L_1 = g^{t_1}, \ldots, L_{n_{max}} = g^{t_{n_{max}}},$$
$$\forall x \in S,\ K_x = \prod_{j=1}^{n_{max}} h_{j,x}^{t_j}.$$

- **KeyReduce**(PK, $sk$, $S$, $S'$) $\rightarrow sk'$: The key reduction algorithm chooses random $t'_1, \ldots, t'_{n_{max}} \in \mathbb{Z}_p$. The private key $sk'$ is:

$$S',\ K' = K g^{a t'_1} = g^{\alpha} g^{a(t_1+t'_1)},\ L'_1 = L_1 g^{t'_1} = g^{t_1+t'_1}, \ldots,$$
$$L'_{n_{max}} = L_{n_{max}} g^{t'_{n_{max}}} = g^{t_{n_{max}}+t'_{n_{max}}},$$
$$\forall x \in S',\ K'_x = K_x \prod_{j=1}^{n_{max}} h_{j,x}^{t'_j} = \prod_{j=1}^{n_{max}} h_{j,x}^{t_j+t'_j}.$$

□

### 5.1.3. *The Subset Signature Construction*

Let $\Pi = ($**Setup**$_{ABE}$, **Encrypt**$_{ABE}$, **KeyGen**$_{ABE}$, **Decrypt**$_{ABE})$ be a CP-ABE scheme that supports key reduction with the algorithm **KeyReduce**$_{ABE}$. Let $\Pi$ have an arbitrary, finite and efficiently-samplable[11] message space $\mathcal{M}$ and access structure space $\mathcal{G}$ that supports AND gates. Let $U$ be a set of strings over an arbitrary alphabet. We construct a signature scheme that supports computing on authenticated subsets of $U$ as follows.

**KeyGen**($1^\lambda$): Run **Setup**$_{ABE}(1^\lambda, U)$ to obtain the key pair $(pk, sk)$, which will serve as the public and secret keys of the signature scheme.
**Sign**($sk, m \subseteq U$): Run **KeyGen**$_{ABE}(sk, m)$ to obtain an ABE private key which will be treated as the signature $\sigma$.
**SignDerive**($pk, \sigma, m, m'$): First, check if $P(m, m') = 1$. If not, output $\perp$. Otherwise, run **KeyReduce**$_{ABE}(pk, \sigma, m, m')$ to obtain the new signature $\sigma'$ and output it.
**Verify**($pk, m, \sigma$): Recall that $m$ is a set of attributes. Create an access structure $\Gamma$ that is the AND of all attributes in $m$. Choose a random value $x \in \mathcal{M}$. Run **Encrypt**$_{ABE}(pk, x, \Gamma)$ to obtain CT. Output 1 if and only if **Decrypt**$_{ABE}(\sigma, \text{CT}) = x$.

*Efficiency* The efficiency of the subset signature construction is closely linked to the efficiency of the ABE scheme employed. Recall that messages to be signed are sets of strings. The signing algorithm for a message $m$ corresponds directly to the time for the CP-ABE scheme to produce a private key for the set of attributes in $m$. Thus, signing time is likely to scale with the size of the message. For the CP-ABE schemes discussed above, computing a signed subset requires a re-randomization of the remaining subset elements, and thus scales with the size of the subset. The verification time is dominated by an encryption under the AND of all attributes in $m$ (the message being verified) together

[11] We mean that it is possible to efficiently sample elements from the set uniformly at random.

with a decryption of the resulting ciphertext using the subset signature as the private key. Overall, an efficient CP-ABE scheme will result in a practical subset performance.

5.1.4. *Security Analysis*

**Theorem 5.6.** *If* $\Pi$ *is (resp., selectively) secure for attribute universe* $U$ *with respect to Definition 5.3, then the above subset signature scheme is (resp., selectively) unforgeable with respect to Definition 2.3 and strongly context hiding with respect to Definition 2.4.*

*Proof.* We argue this theorem in two parts. □

**Lemma 5.7.** (Strong Context Hiding) *If* $\Pi$ *is a CP-ABE scheme that supports key reduction, then the above subset signature scheme is strongly context hiding under Definition 2.4.*

*Proof.* This follows directly from the key reduction property of the CP-ABE scheme. Let $(pk, sk) \leftarrow \mathbf{KeyGen}(1^\lambda)$ be a key pair. A signature scheme (**KeyGen**, **SignDerive**, **Verify**) is strongly context hiding for the simple subset predicate $P$ if for all such triples $((pk, sk), m, m')$ where $P(m, m') = 1$, the following two distributions are statistically close:

$$\big\{\big(sk, \sigma \leftarrow \mathbf{Sign}(sk, m),\ \mathbf{Sign}(sk, m')\big)\big\}_{sk,m,m'}$$
$$\big\{\big(sk, \sigma \leftarrow \mathbf{Sign}(sk, m),\ \mathbf{SignDerive}(pk, \sigma, m, m')\big)\big\}_{sk,m,m'}$$

where the distributions are taken over the random coins of **Sign** and **SignDerive**. If we substitute the signature algorithms for their underlying CP-ABE algorithms, we have the following two distributions:

$$\big\{\big(sk, \sigma \leftarrow \mathbf{KeyGen}_{ABE}(sk, m),\ \mathbf{KeyGen}_{ABE}(sk, m')\big)\big\}_{sk,m,m'}$$
$$\big\{\big(sk, \sigma \leftarrow \mathbf{KeyGen}_{ABE}(sk, m),\ \mathbf{KeyReduce}_{ABE}(pk, \sigma, m, m')\big)\big\}_{sk,m,m'}$$

where the distributions are taken over the random coins of $\mathbf{KeyGen}_{ABE}$ and $\mathbf{KeyReduce}_{ABE}$. The statistical closeness of these distributions is directly guaranteed by the key reduction specification when the predicate is satisfied, i.e., $m' \subseteq m$. □

**Lemma 5.8.** (Unforgeability) *If* $\Pi$ *is a (resp., selectively) secure CP-ABE scheme that supports key reduction, then the above subset signature scheme is (resp., selectively) unforgeable in the* **Unforg** *game.*

*Proof.* We first apply Lemma 2.8, which allows us to only consider adversaries $\mathcal{A}$ that asks queries to *Sign* oracle in the simpler **NHU** game.

We deal with the non-selective case first. Suppose an adversary $\mathcal{A}$ can produce a forgery with probability $\epsilon$ in the **NHU** selective unforgeability game; then we can construct an adversary $\mathcal{B}$ that breaks the selective security of the CP-ABE scheme with key reduction with probability $1/2 + \epsilon/2$. $\mathcal{B}$ begins to run $\mathcal{A}$ and proceeds as follows:

*Setup* When $\mathcal{B}$ receives the public parameters *pk* from its challenger, it passes these on as the signature public key to $\mathcal{A}$.

*Sign* When $\mathcal{A}$ queries **Sign**$(m)$, $\mathcal{B}$ queries its key generation oracle on $m$ and returns the response to $\mathcal{A}$.

*Response* If $\mathcal{A}$ does not output a valid forgery, then $\mathcal{B}$ simply chooses and outputs a random bit. If $\mathcal{A}$ outputs a valid message-signature pair $(m^*, \sigma^*)$, where $m^*$ is not a subset of any message queried to **Sign**, then $\mathcal{B}$ picks two arbitrary messages $m_0, m_1$ in the message space of the CP-ABE scheme. It outputs these to its challenger together with a challenge access structure, which is the AND of all attributes in $m^*$. (Recall that in this signature scheme messages are sets of strings). This challenge access structure is allowed, under the CP-ABE security experiment, because none of the other private keys created by the oracle can satisfy it. Once the challenge ciphertext CT* is returned, $\mathcal{B}$ simply uses the private key $\sigma^*$ to decrypt CT* and to then state which of the two challenge messages was encrypted.

*The Selective Case* Suppose an adversary $\mathcal{A}$ can produce a forgery with probability $\epsilon$ in the **NHU** *selective* unforgeability game; then we can construct an adversary $\mathcal{B}$ that breaks the *selective* security of the CP-ABE scheme with key reduction with probability $\epsilon$ plus a negligible amount. $\mathcal{B}$ behaves as above, except that prior to **Setup** there is a selective disclosure phase where $\mathcal{A}$ first announces the message $m^*$ on which he will forge. $\mathcal{B}$ then announces to its challenger that its challenge access structure will be the AND of all attributes in $m^*$. This information is latter used, as before, in $\mathcal{B}$'s Response phase, where now $\mathcal{A}$ will only output $\sigma^*$.

*Analysis* The following analysis applies to both the selective and non-selective cases. The view of $\mathcal{A}$ when interacting with $\mathcal{B}$ is identical to its view when interacting with a real **NHU** game challenger. Whenever $\mathcal{A}$ correctly produces a forgery, then $\mathcal{B}$ correctly identified the challenge message. Whenever $\mathcal{A}$ fails to produce a forgery, then $\mathcal{B}$ guesses the challenge message with probability $1/2$. Thus, if $\mathcal{A}$ succeeds with probability $\epsilon$, then $\mathcal{B}$ succeeds with probability $\epsilon + (1-\epsilon)/2 = 1/2 + \epsilon/2$. □

## 6. Computing Weighted Averages and Fourier Transforms

So far we only constructed schemes for *univariate* predicates $P$. We now give an example where one computes on multiple signed messages. Let $p$ be a prime, $n$ a positive integer, and $\mathcal{T}$ a set of tags. The message space $\mathcal{M}$ consists of pairs:

$$\mathcal{M} := \mathcal{T} \times \mathbb{F}_p^n$$

Now, define the predicate $P$ as follows: $P(\varepsilon, m) = 1$ for all $m \in \mathcal{M}$ and[12]

[12] Recall, the signature on $\epsilon$ is the output the KeyGen algorithm.

$$P\Big(\big((t_1,\mathbf{v}_1),\ldots,(t_k,\mathbf{v}_k)\big),\ (t,\mathbf{v})\Big)=1 \iff \begin{cases} t=t_1=\cdots=t_k, \text{ and} \\ \mathbf{v}\in \operatorname{span}(\mathbf{v}_1,\ldots,\mathbf{v}_k) \end{cases}$$

Thus, given signatures on vectors $\mathbf{v}_1,\ldots,\mathbf{v}_k$ grouped together by the tag $t$, anyone can create a signature on a linear combination of these vectors. This can be done iteratively so that given signed linear combinations, new signed linear combinations can be created. Unforgeability means that if the adversary obtains signatures on vectors $\mathbf{v}_1,\ldots,\mathbf{v}_k$ for particular tag $t\in\mathcal{T}$ then he cannot create a signature on a vector outside the linear span of $\mathbf{v}_1,\ldots,\mathbf{v}_k$.

Signature schemes for this predicate $P$ are presented in [2,15–18] while schemes over $\mathbb{Z}$ (rather than $\mathbb{F}_p$) are presented in [32]. These schemes were originally designed to secure network coding where context hiding is not needed since there are no privacy requirements for the sender (in fact, the sender is explicitly transmitting all his data to the recipient). The question then is how to construct a system for predicate $P$ above that is both unforgeable and context hiding. Fortunately, we do not need to look very far. The linearly homomorphic signature schemes in [17] can be shown to be context hiding. We state this in the following theorem.

**Theorem 6.1.** *If the CDH assumption holds in group $\mathbb{G}$, then the signature scheme **NCS**$_1$ from [17] is unforgeable and context-hiding in the random oracle model, assuming tags are generated independently at random by the unforgeability challenger when responding to Sign queries.*

Unforgeability is Theorem 6 in [17], which holds only when tags $t_i\in\mathcal{T}$ are generated independently at random by the signer. While context hiding has not been considered before for this scheme, it is not difficult to show that the scheme is context hiding. The scheme is unique in the sense that every vector $\mathbf{v}$ has a unique valid signature.[13] It is easy to show that any homomorphic unique signature must be context hiding and hence **NCS**$_1$ is context hiding.

Viewed in this way, the scheme **NCS**$_1$ gives the ability to carry out authenticated addition on signed data. Consider a server that stores signed data samples $s_1,\ldots,s_n$ in $\mathbb{F}_p$. The signature on sample $s_i$ is actually a signature on the vector $(s_i|\mathbf{e}_i)\in\mathbb{F}_p^{n+1}$, where $\mathbf{e}_i$ is the $i$th unit vector in $\mathbb{F}_p^n$. The server stores $(i,s_i)$ and a signature on $(s_i|\mathbf{e}_i)$. (The vector $\mathbf{e}_i$ need not be stored with the data and can be reconstructed from $i$ when needed). Using **SignDerive**, the server can compute a signature $\sigma$ on the sum $(\sum_{i=1}^n s_i, 1,\ldots,1)$. Since the schemes are context hiding, the server can publish the sum $\sum_{i=1}^n s_i \mod p$ and the signature $\sigma$ on the sum and (provably) reveal no other information on the original data. The "augmentation" $(1,\ldots,1)$ proves that the published message really is the claimed sum of the original samples (the tag $t$ prevents mix-and-match attacks between different data sets). We can similarly publish a sum of any required subset.

More generally, the server can publish an authenticated inner product of the samples $\mathbf{s}:=(s_1,\ldots,s_n)$ with any public vector $\mathbf{c}\in\mathbb{F}_p^n$ without leaking additional information about the samples. This is needed, for example, to publish a weighted average of the

[13] Recall that in *unique* signatures [43] in addition to the regular unforgeability requirement there is an additional uniqueness property: for any honestly generated public key $pk$ and any message $m$ in the message space, there do not exist values $\sigma_1,\sigma_2$ such that $\sigma_1\neq\sigma_2$ and yet $\mathbf{Verify}(pk,m,\sigma_1)=\mathbf{Verify}(pk,m,\sigma_2)=1$.

original data set. Similarly, recall that the Fourier transform of the data $(s_1, \ldots, s_n)$ is a specific linear operator (represented by a specific $n \times n$ matrix) applied to this vector. Therefore, we can publish signed Fourier coefficients of the data. If we only publish a subset of the Fourier coefficients, then by context hiding, we are guaranteed that no other information about $(s_1, \ldots, s_n)$ is revealed.

## 7. Conclusion and Open Problems

In summary, the goal of this work is the study of computing on authenticated data. We presented one unified framework for a variety of distinct concepts in this area, including arithmetic, homomorphic, quotable, redactable, and transitive signatures. The definition we provide tackles unforgeability as well as a strong notion of privacy, where an adversary is unable to distinguish a derived signature from a fresh one even when given the original signature. In this setting, we provide generic constructions for all univariate and closed predicates, as well as specific efficient constructions for predicates such as quoting, subsets, weighted sums, averages, and Fourier transforms.

This work leaves open a host of interesting problems. First, one can imagine predicates that support more complex operations on signed messages. One natural set of examples are spreadsheet operations such as median, standard deviation, and rounding on signed data (satisfying unforgeability and context hiding). Other examples include graph algorithms such as computing a signature on a perfect matching in a signed bipartite graph. Still other examples involve efficiently expanding quoting/redacting to more complex data types, such as (potentially compressed) graphical images.

A first step in this direction may be to improve upon some of the constructions for basic predicates presented herein. For example, as discussed at the end of Sect. 4.2, for quoting/redacting on simple text, it is still unknown how to balance time and space efficiently while achieving full security in the standard model from a simple computational assumption.

## Acknowledgements

We are grateful to the anonymous reviewers of TCC 2012 and the Journal of Cryptology for their helpful comments.

## References

[1] G. Ateniese, D.H. Chou, B. de Medeiros, G. Tsudik, Sanitizable signatures, in *ESORICS '05*. LNCS, vol. 3679 (2005), pp. 159–177

[2] N. Attrapadung, B. Libert, Homomorphic network coding signatures in the standard model, in *Public Key Cryptography—PKC 2011*, vol. 6571 (2011), p. 17

[3] N. Attrapadung, B. Libert, T. Peters, Computing on authenticated data: New privacy definitions and constructions, in *ASIACRYPT* (2012), pp. 367–385

[4] N. Attrapadung, B. Libert, T. Peters, Efficient completely context-hiding quotable and linearly homomorphic signatures, in *Public Key Cryptography* (2013), pp. 386–404

[5] A. Beimel, *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)

[6] M. Bellare, O. Goldreich, S. Goldwasser, Incremental cryptography: the case of hashing and signing, in *CRYPTO '94*. LNCS, vol. 839 (1994), pp. 216–233
[7] M. Bellare, D. Micciancio, B. Warinschi, Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions, in *EUROCRYPT* (2003), pp. 614–629
[8] M. Bellare, G. Neven, Transitive signatures based on factoring and RSA, in *ASIACRYPT '02*. LNCS, vol. 2501 (2002), pp. 397–414
[9] M. Bellare, G. Neven, Transitive signatures: new schemes and proofs. *IEEE Transactions on Information Theory*, **51**:2133–2151 (2005)
[10] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in *IEEE Symposium on Security and Privacy* (2007), pp. 321–334
[11] M. Blum, A. De Santis, S. Micali, G. Persiano, Noninteractive zero-knowledge. *SIAM J. Comput.*, **20**(6):1084–1118 (1991)
[12] D. Boneh, X. Boyen, Efficient selective-ID secure identity-based encryption without random oracles, in *Advances in Cryptology—EUROCRYPT '04*. vol. 3027 (2004), pp. 223–238
[13] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in *CRYPTO '04*. LNCS, vol. 3152 (2004), pp. 45–55
[14] D. Boneh, M.K. Franklin, Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, **32**(3) (2003)
[15] D. Boneh, D. Freeman, Homomorphic signatures for polynomial functions, in *Proc. of Eurocrypt*. Cryptology ePrint Archive, Report 2011/018 (2011)
[16] D. Boneh, D. Freeman, Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures, in *Proc. of PKC*. LNCS, Cryptology ePrint Archive, Report 2010/453. vol. 6571 (2011), pp. 1–16
[17] D. Boneh, D. Freeman, J. Katz, B. Waters, Signing a linear subspace: signature schemes for network coding, in *Public-Key Cryptography—PKC '09*. LNCS, vol. 5443 (Springer, Berlin, 2009), pp. 68–87
[18] D. Boneh, M. Hamburg. Generalized identity based and broadcast encryption schemes, in *ASIACRYPT*. (2008), pp. 455–470
[19] C. Brzuska, H. Busch, O. Dagdelen, M. Fischlin, M. Franz, S. Katzenbeisser, M. Manulis, C. Onete, A. Peter, B. Poettering, D. Schröder, Redactable signatures for tree-structured data: definitions and constructions, in *Applied Cryptography and Network Security (ACNS) '08*. LNCS, vol. 6123 (2010), pp. 87–104
[20] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, F. Volk, Security of sanitizable signatures revisited, in *Public Key Cryptography*. LNCS, vol. 5443 (2009), pp. 317–336
[21] C. Brzuska, M. Fischlin, A. Lehmann, D. Schröder, Santizable signatures: how to partially delegate control for authenticated data, in *BIOSIG 2009* (2009), pp. 117–128
[22] C. Brzuska, M. Fischlin, A. Lehmann, D. Schröder, Unlinkability of sanitizable signatures, in *Public Key Cryptography (PKC) '10*. LNCS, vol. 6056 (2010), pp. 444–461
[23] J. Camenisch, A. Lysyanskaya, Signature schemes and anonymous credentials from bilinear maps, in *Advances in Cryptology—CRYPTO '04*. vol. 3152 (2004), pp. 56–72
[24] R. Canetti, S. Halevi, J. Katz, A forward-secure public-key encryption scheme, in *EUROCRYPT* (2003), pp. 255–271
[25] E. Chang, C.L. Lim, J. Xu, Short redactable signatures using random trees, in *CT-RSA '09: Proceedings of the The Cryptographers' Track at the RSA Conference 2009 on Topics in Cryptology* (2009), pp. 133–147
[26] D. Charles, K.J. K. Lauter, Signatures for network coding. *International Journal of Information and Coding Theory*, **1**(1):3–14 (2009)
[27] M. Chase, M. Kohlweiss, A. Lysyanskaya, S. Meiklejohn, Malleable signatures: complex unary transformations and delegatable anonymous credentials. Cryptology ePrint Archive, Report 2013/179 (2013). http://eprint.iacr.org/. Accessed 17 Mar 2014
[28] D. Chaum, E. van Heyst, Group signatures, in *EUROCRYPT*. LNCS, vol. 547 (1991), pp. 257–265
[29] B. Deiseroth, V. Fehr, M. Fischlin, M. Maasz, N.F. Reimers, R. Stein, Computing on authenticated data for adjustable predicates. Cryptology ePrint Archive, Report 2013/217 (2013). http://eprint.iacr.org/. Accessed 17 Mar 2014

[30] W. Diffie, M. Hellman, New directions in cryptography. *IEEE Transactions on Information Theory*, **22**:644–654 (1976)
[31] C. Fragouli, E. Soljanin, *Network Coding Fundamentals* (Now Publishers Inc., Hanover, MA, 2007)
[32] R. Gennaro, J. Katz, H. Krawczyk, T. Rabin, Secure network coding over the integers, in *Public Key Cryptography—PKC '10*. LNCS, vol. 6056 (Springer, Berlin, 2010), pp. 142–160
[33] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University (2009)
[34] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions (extended abstract), in *FOCS* (1984), pp. 464–479
[35] S. Goldwasser, S. Micali, R.L. Rivest, A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, **17**(2):281–308 (1988)
[36] S. Haber, Y. Hatano, Y. Honda, W. Horne, K. Miyazaki, T. Sander, S. Tezoku, D. Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification, in *ASIACCS '08* (2008), p. 353–362
[37] A. Hevia, D. Micciancio, The provable security of graph-based one-time signatures and extensions to algebraic signature schemes, in *ASIACRYPT '02*. LNCS, vol. 2501 (2002), pp. 379–396
[38] S. Hohenberger, B. Waters, Realizing hash-and-sign signatures under standard assumptions, in *EUROCRYPT '09*. LNCS, vol. 5479 (2009), pp. 333–350
[39] R. Johnson, D. Molnar, D. Song, D. Wagner, Homomorphic signature schemes, in *CT-RSA* (Springer, Berlin, 2002), pp. 244–262
[40] M. Krohn, M. Freedman, D. Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution, in *Proc. of IEEE Symposium on Security and Privacy* (2004), pp. 226–240
[41] A.B. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in *EUROCRYPT* (2010)
[42] A.B. Lewko, B. Waters, New techniques for dual system encryption and fully secure HIBE with short ciphertexts, in *TCC '10*. LNCS, vol. 5978 (2010), pp. 455–479
[43] A. Lysyanskaya, Unique signatures and verifiable random functions from the DH-DDH separation, in *CRYPTO* (2002), pp. 597–612
[44] S. Micali, Computationally sound proofs. *SIAM J. Comput.*, **30**(4):1253–1298 (2000)
[45] S. Micali, R.L. Rivest, Transitive signature schemes, in *CT-RSA '02*. LNCS, vol. 2271 (2002), pp. 236–243
[46] K. Miyazaki, G. Hanaoka, H. Imai, Digitally signed document sanitizing scheme based on bilinear maps, in *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security* (2006), pp. 343–354
[47] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, H. Imai, Digitally signed document sanitizing scheme with disclosure condition control. *IEICE Trans. Fundam.*, **E88-A**(1):239–246 (2005)
[48] K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, Digital document sanitizing problem. *IEICE Technical, Report*, **103**:61–67 (2003)
[49] D. Naccache, Is theoretical cryptography any good in practice? CHES 2010 invited talk (2010). www.iacr.org/workshops/ches/ches2010. Accessed 13 Jun 2012
[50] G. Neven, A simple transitive signature scheme for directed trees. *Theor. Comput. Sci.*, **396**(1–3):277–282 (2008)
[51] R. Rivest, Two signature schemes. Slides from talk given at Cambridge University (2000). http://people.csail.mit.edu/rivest/Rivest-CambridgeTalk.pdf. Accessed 13 Jun 2012
[52] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, **21**(2):120–126 (1978)
[53] R.L. Rivest, A. Shamir, Y. Tauman, How to leak a secret: theory and applications of ring signatures, in *Essays in Memory of Shimon Even* (2006), pp. 164–186
[54] S.F. Shahandashti, M. Salmasizadeh, J. Mohajeri, A provably secure short transitive signature scheme from bilinear group pairs, in *Security and Communication Networks*. LNCS, vol. 3352 (2005), pp. 60–76
[55] A. Shamir, On the generation of cryptographically strong pseudorandom sequences. *ACM Trans Comput Syst*, **1**:38–44 (1983)
[56] N.P. Smart, F. Vercauteren, Fully homomorphic encryption with relatively small key and ciphertext sizes, in *Public Key Cryptography—PKC '10*. LNCS, vol. 6056 (Springer Berlin, 2010), pp. 420–443

[57] N. Smart. ECRYPT2 Yearly Report on Algorithms and Keysizes (2008–2009), Revision 1.0. Edited by Smart (2009). http://people.csail.mit.edu/rivest/Rivest-CambridgeTalk.pdf. Accessed 13 Jun 2012
[58] R. Steinfeld, L. Bull, Y. Zheng, Context extraction signatures, in *Information Security and Cryptology (ICISC)*. LNCS, vol. 2288 (2001), pp. 285–304
[59] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in *Advances in Cryptology—EUROCRYPT '10*. LNCS, vol. 6110 (Springer, Berlin, 2010), pp. 24–43
[60] B. Waters, Efficient identity-based encryption without random oracles, in *Advances in Cryptology—EUROCRYPT '05*. vol. 3494 (2005), pp. 320–329
[61] B. Waters, Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions, in *Advances in Cryptology—CRYPTO '09*. vol. 5677 (2009), pp. 619–636
[62] B. Waters, Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization, in *Public Key Cryptography—PKC '11* (2011), pp. 53–70
[63] L. Wei, S.E. Coull, M.K. Reiter, Bounded vector signatures and their applications, in *ASIACCS '11*. (2011), pp. 277–285
[64] X. Yi, Directed transitive signature scheme, in *CT-RSA '07*. LNCS, vol. 4377 (2007), pp. 129–144
[65] F. Zhao, T. Kalker, M. Médard, K. Han, Signatures for content distribution with network coding, in *Proc. Intl. Symp. Info. Theory (ISIT)* (2007)