Computing on Authenticated Data

Jae Hyun Ahn[1], Dan Boneh[2,⋆], Jan Camenisch[3,⋆⋆], Susan Hohenberger[1,⋆⋆⋆],
Abhi Shelat[4,†], and Brent Waters[5,‡]

[1] Johns Hopkins University
{arjuna,susan}@cs.jhu.edu
[2] Stanford University
dabo@cs.stanford.edu
[3] IBM Research – Zurich
jca@zurich.ibm.com
[4] University of Virginia
abhi@cs.virginia.edu
[5] University of Texas at Austin
bwaters@cs.utexas.edu

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

[⋆] 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 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, a Microsoft Faculty Fellowship and a Google Faculty Research Award.

[†] 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 and CNS-0952692, 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.

R. Cramer (Ed.): TCC 2012, LNCS 7194, pp. 1–20, 2012.

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.

1 Introduction

In tandem with recent progress on computing *any function* on encrypted data, e.g., [27,46,44], this work explores computing on unencrypted signed data. In the past few years, several independent lines of research touched on this area:

- Quoting/redacting: [45,32,2,36,30,17,16,18] 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: [33,50,22,13,26,12,11,48] 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 [25]. 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 [11].
- Transitivity: [41,35,6,31,7,43,49,40] 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 [35]. 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, a subset predicate, and a weighted average over data (which captures weighted sums and Fourier transforms). Our quoting substring construction is novel and significantly more efficient than the generic solutions. It is detailed in Section 4. 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 in Section 5.

1.1 Overview

A general framework. Let $\mathcal{M}$ be some message space and let $2^{\mathcal{M}}$ be its powerset. Consider a predicate $P : 2^{\mathcal{M}} \times \mathcal{M} \to \{0, 1\}$ 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 the full version [1] 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 $\mathbf{v}_1, \ldots, \mathbf{v}_k$.

We emphasize that signature derivation can be iterative. For example, given a message-signature pair (m, σ) from Alice, Bob can publish a derived message-signature pair (m', σ') for an m' where $P(m, m')$ holds. Charlie, using (m', σ'), may further derive a signature σ'' 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

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

attacker should be unable to produce a signature on a message that is not derivable from the set of signed messages at his possession. E.g., suppose Alice generates (m, σ) and gives it to Bob who then publishes a derived signature (m', σ'). Then an attacker given (m', σ') 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 Section 2.3; 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 [23,5,19,9,42] 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 [28].) 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 — effecting 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 the full version [1], 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 ℓ 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. We sketch those constructions in Section 5 and provide further details in [1].

Other Predicates. One can also imagine predicates P 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.

[2] A substring of $x_1 \ldots x_n$ is some $x_i \ldots x_j$ 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 [20], selective security for signatures requires the attacker to give the forgery message before seeing the verification key.

2 Definitions

Definition 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 Π for message space $\mathcal{M}$ and predicate P is a triple of PPT algorithms:

KeyGen(1^λ): 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 σ' 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, σ): Given a public key, message, and purported signature σ, 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 **KeyGen**(1^n) and for all $M \in \mathcal{M}^*$ and $m' \in \mathcal{M}$ we have:

- if $P(M, m') = 1$ then **SignDerive**$(pk, (\mathbf{Sign}(sk, M), M), m') \neq \perp$, and
- for all signature tuples $\{\sigma_m\}_{m \in M}$ such that $\sigma' \leftarrow$ **SignDerive**$(pk, (\{\sigma_m\}_{m \in M}, M), m') \neq \perp$, we have **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 (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 [29]. 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 Π 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 $\mathbf{Sign}(sk, m) \to \sigma$ and places (h, m, σ) 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, σ_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 $\mathbf{SignDerive}(pk, (\{\sigma_m\}_{m\in M}, M), m') \to \sigma'$ and places (h', m', σ') into T, and returns h' to the adversary.
- *Reveal*(h): Returns the signature σ corresponding to handle h, and adds (σ', m') to the set Q.

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

- $\mathbf{Verify}(pk, m', \sigma') = 1$ 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 1.

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

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

A P-homomorphic signature scheme Π 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, $\boldsymbol{Forg}_{\mathcal{A}}$ is negligible in λ.*

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, σ) is a message signature pair and (m', σ') is a message-signature pair derived from it, namely $\sigma' = \textbf{SignDerive}(pk, \sigma, m, m')$. For example, suppose m' is a quote from m. Then given (m', σ') 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 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 the full version [1].

Definition 4 (Strong Context Hiding). *Let $M \subseteq \mathcal{M}^*$ and $m' \in \mathcal{M}$ be messages such that $P(M, m') = 1$. Let $(pk, sk) \leftarrow \textbf{KeyGen}(1^\lambda)$ be a key pair. A signature scheme $(\textbf{KeyGen}, \textbf{SignDerive}, \textbf{Verify})$ is strongly context hiding (for predicate P) if for all such triples $((pk, sk), M, m')$, the following two distributions are statistically close:*

$$\big\{\big(sk, \{\sigma_m\}_{m \in M} \leftarrow \textbf{Sign}(sk, M),\ \ \textbf{Sign}(sk, m')\big)\big\}_{sk, M, m'}$$

$$\big\{\big(sk, \{\sigma_m\}_{m \in M} \leftarrow \textbf{Sign}(sk, M),\ \ \textbf{SignDerive}(pk, (\{\sigma_m\}_{m \in M}, M), m')\big)\big\}_{sk, M, m'}$$

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 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 4 captures a broad range of privacy requirements for derived signatures. Earlier work in this area [32,16,18,15] only considered weaker privacy requirements using more complex definitions. The simplicity and breadth of Definition 4 is one of our key contributions.

Definition 4 uses statistical indistinguishability meaning that even an unbounded adversary cannot distinguish derived signatures from newly created ones. In the full version [1], 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 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 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.

A simpler approach to proving unforgeability. For systems that are strongly context hiding, unforgeability follows from a simpler game than that of Section 2.1. In particular, it suffices to just give the adversary the ability to obtain top level signatures signed by *sk*. In the full version [1], we define this simpler unforgeability game and prove equivalence to Definition 3 using strong context hiding.

2.3 Related Work

Early work on quotable signatures [45,32,38,37,30,17,21,15] 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. [21] and Brzuska et al. [15] (see also Naccache [39, p. 63] and Boneh-Freeman [12,11][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 [21,15,12,11] 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 [21,15,12,11]), 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 [21,15], 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 [38,2,36,18,16] that allow a proxy to compute signatures on related messages, but requires the proxy to have a secret key, and *incremental* signatures [4], 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 [41], where *anyone* can compute on the authenticated data.

[4] As acknowledged in Section 2.2 of Boneh-Freeman [11], our definitional notion is stronger than and predates the "weak context hiding" notion of [11]. Indeed, the fact that [11] 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 [15], 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 Preliminaries: Algebraic Settings

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 1 (CDH [24]). *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 λ:* $\Pr[a, b, \leftarrow \mathbb{Z}_p; z \leftarrow \mathcal{A}(g, g^a, g^b) : z = g^{ab}]$.

4 A Powers-of-2 Construction for Quoting Substrings

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 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 Section 2. In particular, we can view a message M as a pair $(t, m) \in \{0, 1\}, \{0, 1\}^*$. 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.

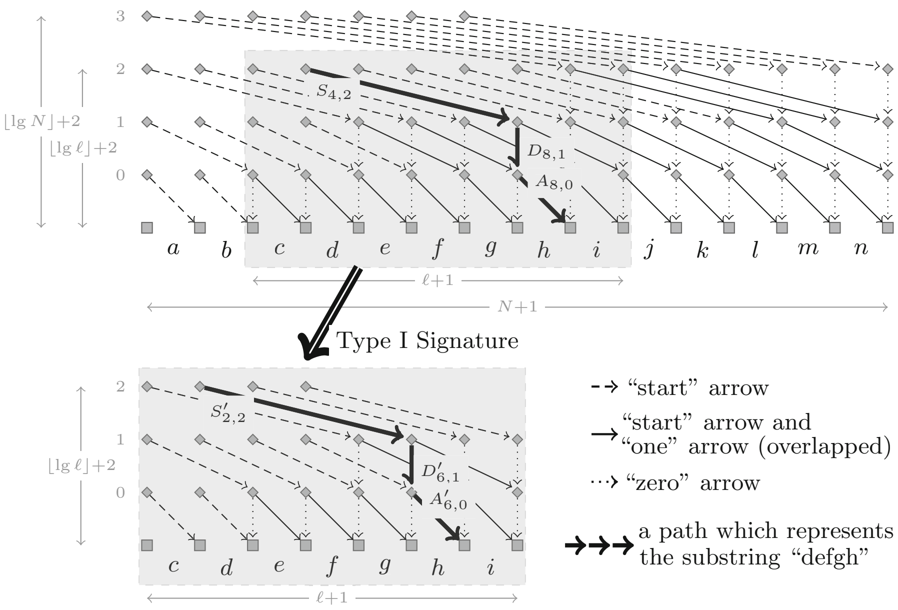

Fig. 1. The top diagram represents a signature on "abcdefghijklmn" with length $N = 14$. Each arrow corresponds to some group elements in the construction. Logically, whenever the elements corresponding to an arrow are included in a quoted signature, the characters underneath this arrow are included 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. Details are below.

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.[5]

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 Figure 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

[5] 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.

correspond to the numbers $\lg n$ down to 0, plus an extra row at the bottom.[6] 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. E.g., 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 Figure 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:

- *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*: operate similarly to start arrows and used to include characters after a start arrow includes the quote prefix. 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 ℓ, 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

[6] 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.

elements corresponding to all arrows that make connections within the columns corresponding to the quote. We illustrate this in Figure 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^λ) : 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 $H : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_s : \{0,1\}^* \rightarrow \mathbb{G}$ 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 Σ.

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 ℓ 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]

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

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

[one arrow: include power j and decrease j]

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

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 $r''_{i,j} \in \mathbb{Z}_p$:

$$[\text{zero arrow: decrease } j]$$
$$D_{i,j} = g^{x_{i,j}} g^{-x_{i,j-1}} g^{z_j r''_{i,j}} \ , \ \widetilde{D_{i,j}} = g^{r''_{i,j}}$$

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, σ, m, m'); if $t' = 0$, output Quote-Type II(PK, σ, m, m'), where these algorithms are defined below.

Quote-Type I(pk, σ, 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 σ 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 σ.) 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 σ.

The quote signature σ' is comprised of the following values:

For $i = 1$ to ℓ' and $j = 0$ to $\lfloor \lg(\ell' - i + 1) \rfloor$, for randomly chosen $t_{i,j} \in \mathbb{Z}_p$:

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

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

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

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

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

Quote-Type II(pk, σ, 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 content-hiding attacks.

Consider the length ℓ' written as a binary string. Let β' 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 jth bit of ℓ' 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 jth bit of ℓ' 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, σ). Otherwise, output Verify-Type II(pk, m, σ), where these algorithms are defined immediately below.

Verify–Type I(pk, m, σ) : Parse σ 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 ℓ and $j = 0$ to $\mathsf{min}(\lfloor \lg(i-1) - 1 \rfloor, \lfloor \lg(\ell - i + 1) \rfloor)$,

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

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

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

In the full version [1], we prove this theorem. We also discuss in detail the efficiency of this construction, how to remove the random oracle, and how to obtain full security.

5 Subsets and Weighted Averages

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. We sketch these constructions here and provide further details in the full version of this paper [1].

Briefly, our subset construction extends the concept of Naor [10] who observed that every IBE scheme can be transformed into a standard signature scheme by applying the IBE KeyGen algorithm as a signing 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 [8,34,47] 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 [31, §6.4], but without context hiding requirements.

Next, we consider a construction for weighted averages, which captures Fourier transforms and weighted sums. This is particularly interesting, because so far we only constructed schemes for *univariate* predicates P. We can 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[7]

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

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 [13,12,11,14,3] while schemes over $\mathbb{Z}$ (rather than $\mathbb{F}_p$) are presented in [26]. 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 observe that under the CDH assumption, the linearly homomorphic signature scheme, $\mathbf{NCS}_1$, due to Boneh, Freeman, Katz and Waters [13]

[7] Recall, the signature on ϵ is the output the KeyGen algorithm.

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.

Acknowledgments. We are grateful to the anonymous reviewers for their helpful comments.

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. Cryptology ePrint Archive, Report 2011/096 (2011), `http://eprint.iacr.org/`
2. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
3. Attrapadung, N., Libert, B.: Homomorphic Network Coding Signatures in the Standard Model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
4. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental Cryptography: The Case of Hashing and Signing. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg (1994)
5. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
6. Bellare, M., Neven, G.: Transitive Signatures Based on Factoring and RSA. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 397–414. Springer, Heidelberg (2002)
7. Bellare, M., Neven, G.: Transitive signatures: New schemes and proofs. IEEE Transactions on Information Theory 51, 2133–2151 (2005)
8. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
9. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
10. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3) (2003)
11. Boneh, D., Freeman, D.M.: Homomorphic Signatures for Polynomial Functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011); Cryptology ePrint Archive, Report 2011/018
12. Boneh, D., Freeman, D.M.: Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011); Cryptology ePrint Archive, Report 2010/453
13. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a Linear Subspace: Signature Schemes for Network Coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
14. Boneh, D., Hamburg, M.: Generalized Identity Based and Broadcast Encryption Schemes. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455–470. Springer, Heidelberg (2008)

15. Brzuska, C., Busch, H., Dagdelen, O., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., Schröder, D.: Redactable Signatures for Tree-Structured Data: Definitions and Constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)
16. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of Sanitizable Signatures Revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
17. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Santizable signatures: How to partially delegate control for authenticated data. In: BIOSIG 2009, pp. 117–128 (2009)
18. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of Sanitizable Signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
19. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
20. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
21. Chang, E.-C., Lim, C.L., Xu, J.: Short Redactable Signatures Using Random Trees. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 133–147. Springer, Heidelberg (2009)
22. Charles, D., Jain, K., Lauter, K.: Signatures for network coding. International Journal of Information and Coding Theory 1(1), 3–14 (2009)
23. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
24. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
25. Fragouli, C., Soljanin, E.: Network Coding Fundamentals. Now Publishers Inc., Hanover (2007)
26. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure Network Coding over the Integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
27. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009)
28. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: FOCS, pp. 464–479 (1984)
29. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal of Computing 17(2), 281–308 (1988)
30. Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: ASIACCS 2008, pp. 353–362 (2008)
31. Hevia, A., Micciancio, D.: The Provable Security of Graph-Based One-Time Signatures and Extensions to Algebraic Signature Schemes. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 379–396. Springer, Heidelberg (2002)
32. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)

33. Krohn, M., Freedman, M., Mazieres, D.: On-the-fly verification of rateless erasure codes for efficient content distribution. In: Proc. of IEEE Symposium on Security and Privacy, pp. 226–240 (2004)
34. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
35. Micali, S., Rivest, R.L.: Transitive Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 236–243. Springer, Heidelberg (2002)
36. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally signed document sanitizing scheme based on bilinear maps. In: ASIACCS 2006: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, pp. 343–354 (2006)
37. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions on Fundamentals E88-A(1), 239–246 (2005)
38. Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H.: Digital document sanitizing problem. IEICE Technical Report, 103(195(ISEC 2003 12-29)), 61–67 (2003)
39. Naccache, D.: Is theoretical cryptography any good in practice? CHES 2010 invited talk (2010), http://www.iacr.org/workshops/ches/ches2010
40. Neven, G.: A simple transitive signature scheme for directed trees. Theoretical Computer Science 396(1-3), 277–282 (2008)
41. Rivest, R.: Two signature schemes. Slides from talk given at Cambridge University (2000), http://people.csail.mit.edu/rivest/Rivest-CambridgeTalk.pdf
42. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret: Theory and Applications of Ring Signatures. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science. LNCS, vol. 3895, pp. 164–186. Springer, Heidelberg (2006)
43. Shahandashti, S.F., Salmasizadeh, M., Mohajeri, J.: A Provably Secure Short Transitive Signature Scheme from Bilinear Group Pairs. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 60–76. Springer, Heidelberg (2005)
44. Smart, N.P., Vercauteren, F.: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
45. Steinfeld, R., Bull, L., Zheng, Y.: Content Extraction Signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
46. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
47. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
48. Wei, L., Coull, S.E., Reiter, M.K.: Bounded vector signatures and their applications. In: ASIACCS 2011, pp. 277–285 (2011)
49. Yi, X.: Directed Transitive Signature Scheme. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 129–144. Springer, Heidelberg (2006)
50. Zhao, F., Kalker, T., Médard, M., Han, K.: Signatures for content distribution with network coding. In: Proc. Intl. Symp. Info. Theory (ISIT) (2007)