Computing on Authenticated Data

Jae Hyun Ahn
Johns Hopkins University
arjuna@cs.jhu.edu

Dan Boneh[*]
Stanford University
dabo@cs.stanford.edu

Jan Camenisch [†]
IBM Zurich Research
jca@zurich.ibm.com

Susan Hohenberger[‡]
Johns Hopkins University
susan@cs.jhu.edu

abhi shelat[§]
University of Virginia
abhi@virginia.edu

Brent Waters[¶]
University of Texas at Austin
bwaters@cs.utexas.edu

February 24, 2011

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.

We carefully formulate the definition of this new primitive, and then provide generic constructions for all univariate and closed predicates, and specific efficient constructions for a broad class of natural predicates such as quoting, weighted sums, averages, and Fourier transforms.

1 Introduction

While there has been considerable progress in computing on encrypted data [24, 44, 42], far less is known about computing on unencrypted signed data. Over the past few years, two independent lines of research touched on this area:

- Quoting/redacting: [43, 29, 1, 35, 27, 13, 12, 14] 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.

[*] Supported by NSF, DARPA, and AFOSR.

[†] 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 CAREER CNS-1053886, DARPA PROCEED, a Microsoft New Faculty Fellowship and a Google Faculty Research Award.

[§] Supported by NSF CAREER CNS-0845811, TC-1018543, DARPA PROCEED, and a Microsoft New Faculty Fellowship.

[¶] Supported by NSF CNS-0915361 and CNS-0952692, the MURI program under AFOSR Grant No: FA9550-08-1-0352, Department of Homeland Security Grant 2006-CS-001-000001-02 (subaward 641), DARPA PROCEED, a Google Faculty Research award, and the Alfred P. Sloan Foundation.

- Arithmetic: [30, 47, 18, 10, 23, 9, 8] 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 [22]. We show in Section 6 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.

In this paper, we put forth a general framework for computing on authenticated data that encompasses both lines of research and much more. We first present clean definitions of security that generalize and strengthen previous results. Then we show two generic solutions for computing signatures on any predicate; however, these generic constructions are not efficient. We next present efficient constructions for three problems: quoting substrings in Section 4, a subset predicate in Section 5, and a weighted average over data in Section 6. 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.

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 Section 6 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 attacker should be unable to produce a signature on a message that is not derivable from the set of signed messages at his possession. For example, suppose Alice generates (m, σ) and gives it to Bob who then

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

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'. For example, a signed average of some data reveals nothing more about the data and a signed quote reveals nothing about the messages from which it was quoted; this includes hiding its length and the position of the quote, and even the origin of the message it was derived from.

Defining context hiding is an interesting and subtle task. In the next section, we give a clean definition that captures a very strong privacy requirement. We discuss earlier attempts at defining privacy following our definition in Section 2.3.

We note that notions such as group or ring signatures [19, 2, 15, 6, 40] 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. For practical purposes 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 *univariate* predicates, namely predicates $P(M, m')$ where M contains a single message (P is false when $|M| > 1$). The first construction uses any standard signature scheme S where the signing algorithm is deterministic. (One can enforce determinism using PRFs [25].) To sign a message $m \in \mathcal{M}$ one simply uses S to sign each message m' such that $P(m, m') = 1$. [2] The signature consists of all these signature components. To verify a signature for m verification checks the signature component corresponding to the message m. To derive a signature m' from m, one simply 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, the signatures in this scheme 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, both of these generic approaches are too expensive for most interesting predicates. We detail these generic approaches and proofs in Appendix B, where we also discuss a generic construction using NIZK.

Our Quoting Construction. We then turn to more efficient constructions, starting with quoting. Since no schemes prior to this work satisfy our strong security requirements, we set out to construct a secure signature for quoting *substrings*[3], which although conceptually simple turns out to be non-trivial to realize securely.

[2] For simplicity here we assume that the predicate P is closed in that if $P(a, b) = P(b, c) = 1$ then $P(a, c) = 1$.

[3] 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”.

We first note that the application of the brute force generic construction to a quoting predicate would result in n^2 components for a signature on a signature of n characters. So any interesting construction must perform more efficiently than this.

A natural approach would be to assign with each character in the message a randomly chosen tag t from a large space. Then a signature of a string of n characters can consist of n standard signatures where the i-th component signs the i-th character and the $(i+1)$-st tag t_{i+1}. In this manner the characters are chained together. The derivation of a substring simply consists of "snipping" off the relevant components and the verification algorithm verifies the individual signatures as well as the chain structure. While such a structure prevents the forgery attacks outlined above, it does not satisfy our context hiding goals. In particular, it is trivial to test (using the tags) if one signature originated from another.

Our main construction provides a secure solution and is significantly more efficient than the brute force construction. 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 ℓ group elements; however the derive procedure cannot be applied again to this short signature.

We prove our construction selectively secure. [4] In addition, we give some potential future directions for achieving adaptive security and removing the use of random oracles.

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 [7] 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 [3, 31] 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. Our construction for weighted sums is presented in Section 6.

Thinking Beyond Beyond simple quoting and summation, one can imagine predicates P that support 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. Not much is known beyond our generic (inefficient) constructions and there is a wealth of deep problems to study.

[4] Following an analog of [16] selective security for signatures requires the attacker to give the forgery message before seeing the verification key.

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

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

2.1 Security: Unforgeability

To define unforgeability, we extend the basic notion of existential unforgeability with respect to adaptive chosen-message attacks [26]. 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 $\mathcal{M}$ described below.

— 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. The challenger returns the handle h to the adversary.
- *SignDerive*$(\vec{h} = (h_1, \ldots, h_k),\ m')$: 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 2.1.

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. However, for all the predicates we consider in this paper this will be quite easy.

Definition 2.3 (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 $\mathbf{Forg}_{\mathcal{A}}$ 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, $\mathbf{Forg}_{\mathcal{A}}$ 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. Thus, our notion is a generalization of unforgeability.

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 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 Appendix A.

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 $(pk, sk) \leftarrow$ **KeyGen**(1^λ) 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:*

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

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' 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 [29, 12, 14, 11] 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 Appendix A, 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.

Note that Definition 2.4 only guarantees privacy when (pk, sk) are generated honestly. In most application scenarios we envision, the context-hiding property is most enjoyed by the original signer, and so we consider it reasonable for the keys to be generated correctly. Future work may consider more adversarial setups, say where the signer wishes derived signatures to leak information about the original messages, but only to some trusted party.

A simpler approach to proving unforgeability. For systems that are strongly context hiding unforgeability follows from a much simpler game than the one used in Section 2.1. In particular, it suffices to just give the adversary the ability to obtain top level signatures signed by *sk*. In Appendix A.2, we define this simpler unforgeability game and prove equivalence to Definition 2.3 using strong context hiding.

2.3 Related Work

Early work on quotable signatures [43, 29, 37, 36, 27, 13, 17, 11] 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. [17] and Brzuska et al. [11] (see also Naccache [38, p. 63] and [9, 8]). 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 [17, 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 [17, 11]), which can have negative privacy implications. Overcoming such document linkage while maintaining unforgeability is a primary technical challenge. This requires moving beyond techniques that use visible nonces to link parts of messages.

Indeed, in most prior constructions, such as [17, 11], 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. The challenge is to "link" different parts of the signature together to prevent these mix and match attacks without employing fixed nonces. We introduce new techniques in this work which link pieces together using randomness that can be re-randomized in controlled ways.

Another line of related work studies "sanitizable" signatures [37, 1, 35, 14, 12] that allow a proxy to compute signatures on related messages, but requires the proxy to have a secret key. In contrast, our proposal is more along the lines of homomorphic encryption, where the proxy computes the new signatures without any additional secret keys.

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 $\mathbf{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ 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 3.1 (CDH [20]) *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 date, while satisfying our strong notions of unforgeability and context hiding. We first describe it in detail and then provide an example of how it works. 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.

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.[5]

[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.

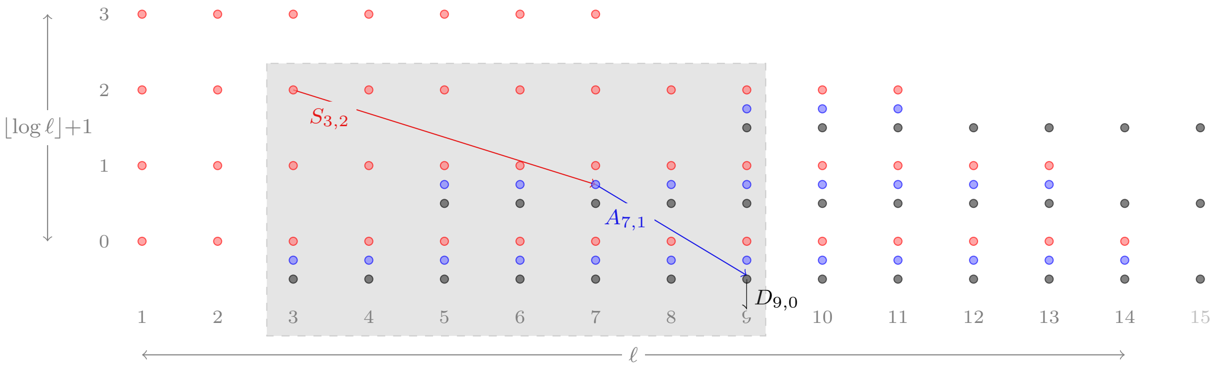

Figure 1: This diagram represents the elements of an original/Type I signature for length $\ell = 14$. We view each dot as the starting point of an arrow and establish simple rules for where these arrows end. To quote a message, follow the arrows over the quote positions. Intuitively, each red/blue arrow makes a power-of-two jump corresponding to the row it is in and allows one to include the message positions it crosses in the quote (i.e., an arrow from column i_c to i'_c includes positions i_c to $i'_c - 1$ in the quote). Each red (resp., blue) dot represents a start (resp., one) arrow starting in (i_c, i_r) and ending in $(i_c + 2^{i_r}, i_r - 1)$. These arrows correspond to the element pairs $S_{i,j}, \widetilde{S_{i,j}}$ (resp., $A_{i,j}, \widetilde{A_{i,j}}$) in our construction. Each black dot represents a zero arrow starting in (i_c, i_r) and ending in $(i_c, i_r - 1)$ and corresponds to the element pair $D_{i,j}, \widetilde{D_{i,j}}$ in our construction. To form a very short Type II signature for the substring $m_{3,6}$ (the substring of m beginning at position 3 of length 6), combine the pairs associated with $S_{3,2}$, $A_{7,1}$ and $D_{9,0}$ as illustrated above. Here, we draw these arrows. A Type II quote always starts with a start arrow (red dot) and then contains one or zero arrows (blue or black dots, respectively) according to the binary representation of the length of the quote (e.g., for $6 = 110_2 = 2^2 + 2^1$, first a (red) start arrow from row 2, then a (blue) one arrow from row 1, and finally a (black) zero arrow in row 0.) To form a Type I signature on the substring $m_{3,6}$, include all dots in the gray box. There are a few black dots at location 15 for termination purposes.

To (roughly) see how this works for a message of length n, visualize n columns with $(\lfloor \lg n \rfloor + 1)$ rows as in Figure 1, where the $(1, 0)$ entry is in the bottom-left. The columns correspond to the characters of the message, so if the message is "abcdefg" then there are seven columns, one under each letter. The rows correspond to the numbers $\lg n$ down to 0. Each location in the matrix (except along row 0) contains one or more out-going arrows. For clarity, we represent these arrows in Figure 1 by colored dots at their point of origination. We'll establish rules for when these arrows exist and where each arrow ends shortly.

A Type II quote will trace a $\lg n$ 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 under 'b' (because $2^2 = 4$ is the largest power of two less than 5) and end in row 0 of 'g'.

The main technical challenge is to make this a $\lg n$-length path rather than a 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 row 0) 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 remaining 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*: same as start arrows; used to include characters after the initial 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 $D_{i,j}, \widetilde{Di,j}$ 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 "abcdefg", we have 7 columns and 3 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 $1 = 2^0$). Moving according to our starting position 'b' and length $5 = 101_2$, we start in row 2 of column 'b', take the start arrow ($S_{2,2}$) to row 2 of column 'f', take the zero arrow ($D_{6,1}$) to row 1 of column 'f', and then take the one arrow ($A_{6,0}$) to row 0 of column 'g'. (Recall that we land one past the last included character.)

For a quote of length ℓ, the elements on this $O(\lg \ell)$ path of arrows form a very short Type II signature. For Type I signatures, we include the elements corresponding to all arrows that make connections within the columns corresponding to the quote. 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.

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 $H : \{0,1\}^* \to \mathbb{G}$ 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 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 $r_{i,j} \in \mathbb{Z}_p$:

$$S_{i,j} = g^{\alpha} g^{-x_{i+2^j,j-1}} H_s(m_{i,2^j})^{r_{i,j}} \quad , \quad \widetilde{S_{i,j}} = g^{r_{i,j}} \qquad \text{[start arrow: start and include power } j\text{]}$$

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

$$A_{i,j} = g^{x_{i,j}} g^{-x_{i+2^j,j-1}} H(m_{i,2^j})^{r'_{i,j}} \quad , \quad \widetilde{A_{i,j}} = g^{r'_{i,j}} \qquad \text{[one arrow: include power } j \text{ and decrease } j\text{]}$$

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

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

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 immediately 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 $t_{i,j} \in \mathbb{Z}_p$:

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

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 $t'_{i,j} \in \mathbb{Z}_p$:

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

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

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

Quote-Type II(pk, σ, m, m') : The quote algorithm takes a Type I signature and produces a Type II signature. As before, 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:

1. Let $I = i - 2^{j+1}$ and $J = j + 1$.
2. Next, compute

$$\begin{aligned} X_{i,j} &= \left(e(g,g)^\alpha \cdot e(H_s(m_{I,2^J}), \widetilde{S_{I,J}})\right) \;/\; e(S_{I,J}, g) \\ &= \left(e(g,g)^\alpha \cdot e(H_s(m_{I,2^J}), g^{r_{I,J}})\right) \;/\; e(g^\alpha g^{-x_{I+2^J,J-1}} H_s(m_{I,2^J})^{r_{I,J}}, g) \\ &= e(g,g)^{x_{I+2^J,J-1}} = e(g,g)^{x_{i,j}} \end{aligned}$$

The verification accepts if and only if all of the following checks 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 4.1 (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.*

Proof of the above theorem appears in Appendix C.

Efficiency Discussion This construction presents the best known balance between time and space complexity. The quotable (Type I) signatures require $O(\ell \lg \ell)$ elements in $\mathbb{G}$ for a message of length ℓ. 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 ℓ 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 ℓ for re-randomization. Similarly, verification requires only $O(\lg \ell)$ pairings.

Removing the Random Oracle and Obtaining Full Security There are a few different options for adapting the above construction to the standard model. 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, we can use the Boneh-Boyen techniques [5] 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, we can remove the random oracle by instantiating H with the Waters hash [45] 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 ℓ is the length of the quote. A direction for achieving full security is to use the recent "Dual System" techniques introduced by Waters [46]. 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 [32] 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.

5 A Construction for Subset Predicates based on ABE

In this section we briefly point out a surprising connection to Attribute Based Encryption (ABE). We show that existing constructions for Ciphertext-Policy ABE [3, 31] naturally lead to context hiding quotable signatures for arbitrary message subsets (as opposed to the substring predicate considered in the previous section). In particular, a message will be a sets of strings (strings can be used to encode elements for different types of sets), and the predicate $P(\vec{m}, m') = 1$ iff $m' \subseteq m_i$ for some $m_i \in \vec{m}$. Observer that this disallows "collusions" between two different signatures where m' is a subset of the union of multiple messages, but not any single one. (Otherwise, this would be trivially realizable from standard signatures schemes.)

Our main tool is an observation of Naor that shows that secret keys in Identity Based Encryption [7] 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 words in a message $m \in \Sigma^*$ as attributes, that is, we treat a message $m = (a_1, \ldots, a_\ell) \in \Sigma^\ell$ as a sequence of attributes $a_1, \ldots, a_\ell$, then we can define the signature on m as a set of ℓ secret keys corresponding to the ℓ 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 words in m by simply removing the secret keys corresponding to words 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 [3, 31] 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 informally.

Theorem 5.1 (informal) *The Ciphertext-Policy ABE systems in [3, 31] translate using Naor's transformation into a signature scheme supporting quoting for arbitrary subsets of a message. Security of the CP-ABE implies unforgeability and context hiding.*

6 An Arithmetic Example: Computing Weighted Averages

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[6]

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

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 $\mathbf{v}_1, \ldots, \mathbf{v}_k$.

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

Signature schemes for this predicate P are presented in [10, 9, 8] while schemes over $\mathbb{Z}$ (rather than $\mathbb{F}_p$) are presented in [23]. 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 [10] 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 $\boldsymbol{NCS}_1$ from [10] is unforgeable and context-hiding in the random oracle model, assuming tags are generated independently at random by the unforgeability challenger when respoding to Sign queries.*

Unforgeability is Theorem 6 in [10], 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.[7] It is easy to show that any homomorphic unique signature must be context hiding and hence $\mathbf{NCS}_1$ is context hiding.

Viewed in this way, the scheme $\mathbf{NCS}_1$ 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 ith 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 σ on the sum $(\sum_{i=1}^n s_i, 1, \ldots, 1)$. Since the schemes are context hiding, the server can publish the sum $\sum_{i=1}^n s_i$ and the signature σ 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 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 Directions

We presented a new vision for computing on authentication data via the notion of slightly homomorphic signatures. We formulated strong notions of unforgeability and privacy that are not captured by prior definitions. Finally, we provide generic constructions for all univariate and closed predicates, and specific efficient constructions for a broad class of natural predicates such as quoting, weighted sums, averages, and Fourier transforms.

While these predicates are an important step, there are a wealth of desirable predicates P that one could imagine supporting complex operations on signed messages, such as speadsheet

[7] Recall that in *unique* signatures [33] 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 σ_1, σ_2 such that $\sigma_1 \neq \sigma_2$ and yet $\mathbf{Verify}(pk, m, \sigma_1) = \mathbf{Verify}(pk, m, \sigma_2) = 1$.

operations, graph algorithms and image processing. Many of these directions appear to offer deep problems requiring new techniques for efficient realization.

References

[1] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In *ESORICS '05*, volume 3679 of LNCS, pages 159–177, 2005.

[2] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, pages 614–629, 2003.

[3] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[4] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[5] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT '04*, volume 3027, pages 223–238, 2004.

[6] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO '04*, volume 3152 of LNCS, pages 45–55, 2004.

[7] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3), 2003.

[8] Dan Boneh and David Freeman. Homomorphic signatures for polynomial functions. In *Proc. of Eurocrypt*, 2011. Cryptology ePrint Archive, Report 2011/018.

[9] Dan Boneh and David Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Proc. of PKC*, 2011. Cryptology ePrint Archive, Report 2010/453.

[10] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In *Public-Key Cryptography — PKC '09*, volume 5443 of *Springer LNCS*, pages 68–87, 2009.

[11] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable signatures for tree-structured data: definitions and constructions. In *Applied Cryptography and Network Security (ACNS) '08*, volume 6123 of LNCS, pages 87–104, 2010.

[12] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In *Public Key Cryptography*, volume 5443 of LNCS, pages 317–336, 2009.

[13] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Santizable signatures: How to partially delegate control for authenticated data. In *BIOSIG 2009*, pages 117–128, 2009.

[14] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In *Public Key Cryptography (PKC) '10*, volume 6056 of LNCS, pages 444–461, 2010.

[15] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO '04*, volume 3152, pages 56–72, 2004.

[16] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[17] Ee-Chien Chang, Chee Liang Lim, and Jia 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*, pages 133–147, 2009.

[18] Denis Charles, K Jain, and K Lauter. Signatures for network coding. *International Journal of Information and Coding Theory*, 1(1):3–14, 2009.

[19] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, volume 547 of LNCS, pages 257–265, 1991.

[20] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[21] Nigel Smart (editor). ECRYPT2 Yearly Report on Algorithms and Keysizes (2008-2009), Revision 1.0, July 27, 2009. Available at `http://www.ecrypt.eu.org/documents/D.SPA.7.pdf`.

[22] Christina Fragouli and Emina Soljanin. *Network Coding Fundamentals.* Now Publishers Inc., Hanover, MA, USA, 2007.

[23] Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In *Public Key Cryptography — PKC '10*, volume 6056 of *Springer LNCS*, pages 142–160, 2010.

[24] Craig Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009.

[25] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479, 1984.

[26] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[27] Stuart Haber, Yasuo Hatano, Yoshinori Honda, William Horne, Kunihiko Miyazaki, Tomas Sander, Satoru Tezoku, and Danfeng Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In *ASIACCS '08*, pages 353–362, 2008.

[28] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In *EUROCRYPT '09*, volume 5479 of LNCS, pages 333–350, 2009.

[29] Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *CT-RSA*, pages 244–262. Springer-Verlag, 2002.

[30] M. Krohn, M. Freedman, and D. Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proc. of IEEE Symposium on Security and Privacy*, pages 226–240, 2004.

[31] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, 2010.

[32] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC '10*, volume 5978 of LNCS, pages 455–479, 2010.

[33] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO*, pages 597–612, 2002.

[34] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[35] Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki 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*, pages 343–354, 2006.

[36] Kunihiko Miyazaki, Mitsuru Iwamura, Tsutomu Matsumoto, Ryoichi Sasaki, Hiroshi Yoshiura, Satoru Tezuka, and Hideki Imai. Digitally signed document sanitizing scheme with disclosure condition control. *IEICE Transactions on Fundamentals*, E88-A(1):239–246, 2005.

[37] Kunihiko Miyazaki, Seiichi Susaki, Mitsuru Iwamura, Tsutomu Matsumoto, Ryoichi Sasaki, and Hiroshi Yoshiura. Digital document sanitizing problem. *IEICE Technical Report*, 103(195(ISEC2003 12-29)):61–67, 2003.

[38] David Naccache. Is theoretical cryptography any good in practice? CHES 2010 invited talk, 2010. available at `www.iacr.org/workshops/ches/ches2010`.

[39] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, February 1978.

[40] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret: Theory and applications of ring signatures. In *Essays in Memory of Shimon Even*, pages 164–186, 2006.

[41] Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transaction on Computer Systems*, 1:38–44, 1983.

[42] N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography — PKC '10*, volume 6056 of *Springer LNCS*, pages 420–443, 2010.

[43] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Context extraction signatures. In *Information Security and Cryptology (ICISC)*, volume 2288 of LNCS, pages 285–304, 2001.

[44] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology — EUROCRYPT '10*, volume 6110 of *Springer LNCS*, pages 24–43, 2010.

[45] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT '05*, volume 3494, pages 320–329, 2005.

[46] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology – CRYPTO '09*, volume 5677, pages 619–636, 2009.

[47] Fang Zhao, Ton Kalker, Muriel Médard, and Keesook Han. Signatures for content distribution with network coding. In *Proc. Intl. Symp. Info. Theory (ISIT)*, 2007.

A A Computational Definition of Context Hiding

Let $(\mathbf{KeyGen}, \mathbf{SignDerive}, \mathbf{Verify})$ 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 $P(m, m') = 1$. The challenger computes the following three values: $\sigma \leftarrow \mathbf{Sign}(sk, m)$, $\sigma_0 \leftarrow \mathbf{Sign}(sk, m')$ and $\sigma_1 \leftarrow \mathbf{SignDerive}(pk, \sigma, m, m')$. The challenger then picks a random $b \in \{0, 1\}$ and returns (σ, σ_b) to the adversary.

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 A.1 (Context Hiding) *For a predicate P and message space $\mathcal{M}$, a P-homomorphic signature scheme* $(\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{SignDerive}, \mathbf{Verify})$ *is context hiding if for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\boldsymbol{CH}}$ is negligible in λ.*

A.1 Relation to Strong Context Hiding

Lemma A.2 *A homomorphic signature scheme that is strongly context hiding is context hiding.*

Proof. (Sketch) Let $\Pi = (\mathbf{KeyGen}, \mathbf{SignDerive}, \mathbf{Verify})$ 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 $(PK, SK) \leftarrow \mathbf{KeyGen}(1^\lambda; z_1)$, 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 Π is strongly context hiding, then $p(\lambda)$ must be exponentially small, and so Π must also be context-hiding.

The adversary A' works as follows: On input the challenge tuple (SK, σ, σ'), 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 (σ, σ'). 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 (σ, σ') 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 $p(\lambda)$. □

A.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 σ.

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 A.3 *A quoteable signature scheme* Π *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* λ.

Lemma A.4 *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 $\bot$ 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 $\bot$ 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, ϵ) with (i, σ, m, ϵ), and return σ. 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, σ, m, i_1), and return σ.

Claim A.5 $\Pr[H_1(\Pi, \mathcal{A}, \lambda, P) = 1] = \Pr[\mathbf{Unforg}(\Pi, \mathcal{A}, \lambda, P) = 1]$.

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 $\bot$ 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, ϵ) with (i, σ, m, ϵ), and return σ. 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, σ, m, i_1), and return σ.

Claim A.6 *$H_{2,0}(\Pi, \mathcal{A}, \lambda, P)$ is identically distributed to $H_1(\Pi, \mathcal{A}, \lambda, P)$.*

By inspection.

Claim A.7 *$H_{2,i}(\Pi, \mathcal{A}, \lambda, P)$ 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. □

B Generic Constructions for Univariate 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 Section 2.1.)

3. For all $m \in \mathcal{M}$, $P(m, m) = 1$.

In this section, we present and discuss generic approaches to 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.

B.1 A Brute Force Construction From Any Signature Scheme

Let (G, S, V) be a signature scheme with a deterministic signing algorithm.[8] One can construct a P-homomorphic signature scheme for any simple predicate P as follows:

KeyGen(1^λ) : 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 σ is the tuple $(S(sk, m), U = \{S(sk, m') \mid m' \in P^0(\{m\})\})$.

SignDerive(pk, σ, 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 σ_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 σ_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, σ) : 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 B.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. 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.

[8] 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.

B.2 An Accumulator-based Construction

Assumption B.2 (RSA [39]) *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 B.3 (Shamir [41]) *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 B.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^λ) : The setup algorithm chooses N as a 20λ-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 2λ-bit prime numbers, e.g., [28], which we treat as a random oracle.[9] 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, σ, 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

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

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

Verify(pk, m, σ) : 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 ℓ-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 ℓ-symbol quote. Thus, this scheme optimizes on space, but may require significant computation.

Theorem B.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.

[9] We choose our modulus and hash output lengths to obtain λ-bit security based on the recent estimates of [21].

Lemma B.6 (Context-Hiding) *The homomorphic signature scheme from §B.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

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

Because $\mathbf{Sign}(sk, m')$ and $\mathbf{SignDerive}(pk, (\sigma, m), m')$ 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 B.7 (Unforgeability) *If the RSA assumption holds, then the Section B.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 [28]. 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 B.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 A.4, 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 ϵ in the **NHU** game; then we can construct an adversary $\mathcal{B}$ that breaks the RSA assumption (with odd prime e^*) with probability ϵ/s minus a negligible amount as follows.

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

Setup $\mathcal{B}$ chooses 2λ-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 jth 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 $\mathbf{primes}(U)$ 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, σ).

Response Eventually, $\mathcal{A}$ outputs a valid message-signature pair (M, σ), 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 $\mathbf{primes}(U)$ 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 B.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, σ) 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, σ) 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 $\mathbf{Adv}_{\mathcal{A}}$[Game x] as the advantage of adversary $\mathcal{A}$ in Game x.

Claim B.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 B.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 2λ-bit primes are chosen at random. By Theorem B.4, we know that there are at least $2^{2\lambda}/(2\lambda)$ 2λ-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 λ. □

Claim B.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 σ can only be a valid forgery if $\mathcal{A}$ guessed the 2λ-bit prime that H_p would respond with on input M. By Theorem B.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 B.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 $1 \leq i^* \leq s$ was chosen at random. Thus, the probability that $\mathcal{A}$ forges on the i^*th query is $1/s$. □

This completes our proof. □

B.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 [4] 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 (non black-box) reductions. In particular, the statements to prove in non-interactive zero-knowledge require transforming the circuits of the signature scheme and the quoting predicate into an instance of Hamiltonian circuit or 3-SAT. 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[10].

C Security Proof for Powers-of-Two Construction

We now provide a proof of Theorem 4.1 in Section 4 by showing the following lemmas.

[10] Using non-interactive CS-proofs [34] 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.

Lemma C.1 (Strong Context-Hiding) *The Section 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, $\mathbf{SignDerive}(pk, \sigma, M', M)$ has an identical distribution to that of $\mathbf{Sign}(sk, M)$, which implies that the two distributions are statistically close.

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

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

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

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 $t' = 1$, $\mathbf{Sign}(SK, M')$ is composed of the following values:

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

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

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

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

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 $\mathbf{SignDerive}(pk, \sigma, M', M)$ has identical distribution as that of $\mathbf{Sign}(sk, M')$.

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

$$\begin{aligned}
B &= g^{\alpha} \cdot H_s(m'_{\beta})^{u} \prod_{j<\beta,\ \ell'_j=1} H(m'_j)^{u_j} \prod_{j'<\beta,\ \ell'_{j'}=0} g^{z_{j'} u_{j'}} \\
\tilde{S} &= g^{u}, \quad Z_j = g^{u_j}
\end{aligned}$$

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

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

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, σ, M, M') has identical distribution as that of **Sign**(sk, M').

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

Lemma C.2 (Unforgeability) *If the CDH assumption holds in $\mathbb{G}$, then the Section 4 quotable signature scheme is selectively unforgeable in the* **Unforg** *game in the random oracle model.*

Proof. (Sketch) We first apply Lemma A.4, 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 ϵ in the selective **NHU** unforgeability game; then we can construct an adversary $\mathcal{B}$ that breaks the CDH assumption with probability ϵ 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 β be the highest bit of ℓ^* 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, $\mathcal{B}$ give 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^β where β is the largest power of two less than ℓ^*, 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:

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

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, 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 β^* is highest bit of ℓ^* 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 ℓ^* 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 ℓ^* 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 jth bit of ℓ^* is 1. If the jth 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 jth bit of ℓ^* is 1. If the jth 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 jth bit of ℓ^* 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 ℓ 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 ℓ and $j = 0$ to $\mathsf{min}(\lfloor \lg(i - 1) - 1 \rfloor, \lfloor \lg(\ell - i + 1) \rfloor)$:

(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

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

(b) Else, if $X_{i,j} = 1$ and $X_{i,j-1} = 0$, then the jth bit of ℓ^* 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

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

(c) Else, if $X_{i,j} = 0$ and $X_{i,j-1} = 1$, then the jth bit of ℓ^* 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

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

(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 $\beta = \lfloor \lg(\ell) \rfloor$. ℓ^*_i denotes i-th bit of ℓ^* when we start counting with zero as the least significant bit, and ℓ_i denotes i-th bit of ℓ.
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 δ 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 γ 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 σ^* on $M^* = (t^*, m^*)$. Recall that $\ell^* = |m^*|$ and $\beta = \lfloor \lg(\ell^*) \rfloor$. Here ℓ^*_i denotes i-th bit of ℓ^* 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:

- γ such that $H_s(m^*_\beta) = g^\gamma$.
- δ_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 σ^*.
  - $S_{1,\beta} = g^\alpha g^{-x_{1+2^\beta,\beta-1}} H_s(m^*_\beta)^{r_c}$, $\widetilde{S_{1,\beta}} = g^{r_{1,\beta}}$

    $\mathcal{B}$ knows γ 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 δ 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 σ^* 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 γ 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 δ_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.

□