Computing on Authenticated Data for Adjustable Predicates

Björn Deiseroth, Victoria Fehr, Marc Fischlin, Manuel Maasz, Nils Fabian Reimers, and Richard Stein

Darmstadt University of Technology, Germany

Abstract. The notion of P-homomorphic signatures, introduced by Ahn et al. (TCC 2012), generalizes various approaches for public computations on authenticated data. For a given predicate P anyone can derive a signature for a message m' from the signatures of a set of messages M, as long as $\mathsf{P}(M, m') = 1$. This definition hence comprises notions and constructions for concrete predicates P such as homomorphic signatures and redactable signatures.

In our work we address the question of how to combine P_i-homomorphic schemes for different predicates $\mathsf{P}_1, \mathsf{P}_2, \ldots$ to create a richer and more flexible class of supported predicates. One approach is to statically combine schemes for predicates into new schemes for logical formulas over the predicates, such as a scheme for AND ($\mathsf{P}_1 \wedge \mathsf{P}_2$). The other approach for more flexibility is to derive schemes which allow the signer to dynamically decide which predicate to use when signing a message, instead of supporting only a single, fixed predicate.

We present two main results. One is to show that one can indeed devise solutions for the static combination for AND, and for dynamically adjustable solutions for choosing the predicate on the fly. Moreover, our constructions are practical and add only a negligible overhead. The other main result is an impossibility result for static combinations. Namely, we prove that, in contrast to the case of AND, many other formulas like the logical OR ($\mathsf{P}_1 \vee \mathsf{P}_2$) and the NOT ($\neg\mathsf{P}$) do not admit generic combinations through so-called canonical constructions. This implies that one cannot rely on general constructions in these cases, but must use other methods instead, like finding new predicate-specific solutions from scratch.

1 Introduction

The notion of P-homomorphic signatures has been put forward by Ahn et al. [1] as a generalization of several concurrent approaches to compute on authenticated data. The predicate P takes as input a set of messages M and determines the admissible messages m' which can be derived from M, and for which a signature can be publicly computed from the signatures for the messages in M. Examples covered by such signatures include homomorphic signatures [14,19,13,6,16,18,3,8,7,15,11] where m' is the sum of all messages in M, transitive signatures [20,5,23,26,25,10] where m' describes a path in a graph given by

M, and redactable signatures [19,24,21,2,17,12,22,9] where m' is a substring of the single message M.

Ahn et al. [1] proposed two general security notions for P-homomorphic signatures. The first one is unforgeability and says that one should not be able to forge signatures for fresh messages which have not been signed before, and which are not publicly derivable. The other notion is called context hiding and provides strong privacy. It says that a derived signature for an admissible message m' and freshly created signatures for m' have statistically close distributions. This guarantees for instance that the original message in case of redactable signatures remains hidden. The context hiding notion has been subsequently refined in [4].

P-Homomorphic Signatures with Adjustable Predicates. While the abstract notion of P-homomorphic signatures is very handy for arguing about the security of solutions, any construction so far, even the ones in [1,4], are for a specific fixed predicate P, such as quoting substrings of a message. What is currently unknown is how to adjust solutions for fixed predicates in the following sense:

- One desirable option may be the possibility to combine a set of given homomorphic schemes for predicates $\mathsf{P}_1, \mathsf{P}_2, \ldots$ into one for a new P-homomorphic signature scheme. Here, P may be a simple combination such as $\mathsf{P}_1 \wedge \mathsf{P}_2$ or $\mathsf{P}_1 \vee \mathsf{P}_2$, or describe even more complex functions. An example are two redactable schemes, one allowing for redaction only at the front of the message (P_1), and the other one enabling redaction only at the end (P_2). Then a $\mathsf{P}_1 \vee \mathsf{P}_2$-homomorphic scheme would be a scheme for quoting substrings, by first pruning at the front and then truncating in another step at the end. Note that the problem here is to present a general transformation which supports a rich set of combinations from, say, basic predicates $\mathsf{P}_1, \mathsf{P}_2, \ldots$, instead of having to build schemes for P from scratch.
- Another desirable feature, which is not offered by the previous ability to combine predicates, is that signer can decide "on the fly" for each signature which predicate P the signature should support. Here, the set of admissible predicates is only bound by the universe $\mathcal{P}$ of predicates for which such signature schemes have been devised yet. This would allow to make the set of admissible message derivates depend on the message itself, e.g., supporting selective redaction for different messages.

We call general constructions with the first property *statically adjustable* because the combined predicate P is fixed at the time of key generation. The latter schemes are called *dynamically adjustable.* Both approaches have their merits and display their full power only in combination. One can first derive (statically) adjustable schemes for a larger universe $\mathcal{P}$, and then use this universe for the dynamically adjustable scheme.

Constructing Schemes with Statically Adjustable Predicates. We first investigate simple static combinations such as $\mathsf{P}_1 \wedge \mathsf{P}_2$, $\mathsf{P}_1 \vee \mathsf{P}_2$, and $\neg\mathsf{P}$. Having solutions for these cases would immediately allow arbitrarily complex combinations of predicates. Our first result is to confirm for the logical AND that the "componentwise" solution works: sign each message with the schemes for predicates $\mathsf{P}_1, \mathsf{P}_2$ individually, and derive signatures by applying the corresponding algorithms for each component.

Our main result is to show that the logical OR, $\mathsf{P}_1 \vee \mathsf{P}_2$, in general does not admit *canonical* constructions. Such canonical constructions can combine given signatures of the individual schemes into one for the $\mathsf{P}_1 \vee \mathsf{P}_2$ predicate, and can vice versa split any signature for the OR into parts for the individual schemes. Our AND construction is of this type. Our negative result for the OR holds for (almost) arbitrary predicates $\mathsf{P}_1, \mathsf{P}_2$, essentially only excluding trivial examples like $\mathsf{P}_1 \vee \mathsf{P}_1$. Note that we cannot hope to show a similar result for *non*-canonical solutions, as for some cases we know constructions from scratch for $\mathsf{P}_1 \vee \mathsf{P}_2$ (e.g., for quotable substrings).

We actually present a more general result, saying that one cannot find canonical constructions for any predicate combination $f(\mathsf{P}_1, \mathsf{P}_2, \dots)$ if one is able to efficiently find a derivable message m' under $f(\mathsf{P}_1, \mathsf{P}_2, \dots)$ and from a message set M, such that m' is not derivable under one of the predicates individually. This excludes the AND case, because any derivable message m' in $\mathsf{P}_1 \wedge \mathsf{P}_2$ must be also valid according to both in P_1 and P_2. Yet, this notion includes the OR case if m' can be derived under one predicate, and therefore the OR, but not under the other predicate. It also covers the NOT case straightforwardly, because if m' is derivable under $f(\mathsf{P}_1) = \neg\mathsf{P}_1$, then it is clearly not derivable under P_1. The impossibility result holds even if the canonical construction depends on f and the predicates. Put differently, it seems that the only general and non-trivial solutions for statically adjustable predicates are the ones for logical ANDs.

Constructing Schemes with Dynamically Adjustable Predicates. Does the negative result for statically adjustable parameters also rule out solutions for the dynamic case? Not necessarily, because in this case we assume that the signer adaptively chooses the predicate P from the universe $\mathcal{P}$ for which constructions are already known. Indeed we show that the "certify-then-sign" construction provides a solution in this case: use a regular signature scheme to certify a public key for the P-homomorphic scheme for the chosen predicate $\mathsf{P} \in \mathcal{P}$ and sign the message under the secret key for P. Some care must be taken, though, because in order to preserve context hiding the key pair for the P-homomorphic scheme must remain fixed throughout the life time.

2 Preliminaries

We recall the definition and security notions of P-homomorphic signatures, as given in [1,4], and adopt them slightly for our adjustable setting.

2.1 Adjustable $\mathcal{P}$-homomorphic Signature Schemes

We assume a fixed but public universe $\mathcal{P}$ of predicates $\mathsf{P}_1, \mathsf{P}_2, \dots$, each predicate associated with a publicly known P_i-homomorphic signature scheme. A predicate $\mathsf{P}_i : 2^{\mathcal{M}} \times \mathcal{M} \to \{0,1\}$ indicates whether a set of messages M allows to derive another message m' from the message space $\mathcal{M}$ or not. We give the signer and the verifier the predicate P in question as additional input. In case of a single fixed predicate P, as for the statically adjustable setting, where the universe $\mathcal{P}$ is a singleton, this is an invariant for the scheme and could be ignored by both algorithms. In fact, in this case the notion basically coincides with the definition of a P-homomorphic scheme, the only difference being the predicate given to the signers and verifier as additional input. In this sense the definition of schemes with statically adjustable predicates is a rehash of the notion of P-homomorphic signatures. We stress that we do not suggest to change the terminology for P-homomorphic schemes. The reader should bear in mind, however, that schemes with statically adjustable predicates in this paper implicitly assume a *construction* from selected P-homomorphic schemes underneath. In light of this it matches the dynamic counterpart where predicates are chosen adaptively for each signature.

We simplify the notation below, and write $\mathsf{Verify}(pk, M, \Sigma, \mathsf{P})$ as shorthand for $\bigwedge_{m \in M} \mathsf{Verify}(pk, m, \sigma_m, \mathsf{P})$ with $\Sigma = \{\sigma_m\}_{m \in M}$. Similarly, we sometimes write $\Sigma \leftarrow \mathsf{Sign}(sk, M, \mathsf{P})$ for $\Sigma = \{\mathsf{Sign}(sk, m, \mathsf{P}) \mid m \in M\}$.

Definition 1 (Adjustable $\mathcal{P}$-homomorphic Signature Scheme). *A* (statically or dynamically) adjustable $\mathcal{P}$-homomorphic signature scheme *is a tuple of PPT algorithms* ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignDer}, \mathsf{Verify}$) *such that:*

- $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *maps the security parameter* $\lambda \in \mathbb{N}$*, given in unary, to a key pair.*
- $\sigma \leftarrow \mathsf{Sign}(sk, m, \mathsf{P})$ *on input the secret key sk, a message* $m \in \mathcal{M}$*, and a predicate* $\mathsf{P} \in \mathcal{P}$ *returns a signature* σ *to* m *and* P*.*
- $\sigma' \leftarrow \mathsf{SignDer}(pk, M, \Sigma, m', \mathsf{P})$ *takes as input the public key pk, a set of messages* $M \subseteq \mathcal{M}$ *along with signatures* $\Sigma = \{\sigma_m\}_{m \in M}$*, a message* $m' \in \mathcal{M}$*, and the predicate* $\mathsf{P} \in \mathcal{P}$ *to be applied, and outputs a signature* σ' *(or a special symbol* $\perp$ *indicating failure).*
- $b \leftarrow \mathsf{Verify}(pk, m, \sigma, \mathsf{P})$*, given the public key pk, a signature* σ*, a message* $m \in \mathcal{M}$*, and a predicate* $\mathsf{P} \in \mathcal{P}$*, returns* 1 *if the signature is valid for the given message, and* 0 *if not.*

We assume the usual correctness condition, namely, that for any $\lambda \in \mathbb{N}$*, any* $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, any* $(m, M, m') \in \mathcal{M} \times 2^{\mathcal{M}} \times \mathcal{M}$ *and any* $\mathsf{P} \in \mathcal{P}$ *we have:*

- *if* $\sigma \leftarrow \mathsf{Sign}(sk, m, \mathsf{P})$*, then* $\mathsf{Verify}(pk, m, \sigma, \mathsf{P}) = 1$ *with probability* 1*; and*
- *for any* $\Sigma = \{\sigma_m\}_{m \in M}$*, if* $\mathsf{Verify}(pk, M, \Sigma, \mathsf{P}) = 1$ *and* $\mathsf{P}(M, m') = 1$*, then for any* $\sigma' \leftarrow \mathsf{SignDer}(pk, M, \Sigma, m', \mathsf{P})$ *we have* $\mathsf{Verify}(pk, m', \sigma', \mathsf{P}) = 1$ *with probability* 1*.*

2.2 Unforgeability

For any predicate P and set M of messages it is convenient to consider the set of messages which can be derived (recursively) from M through P. Hence, similar to [1], we define $\mathsf{P}(M) = \{m' \in \mathcal{M} \mid \mathsf{P}(M, m') = 1\}$ for any $M \subseteq \mathcal{M}$, as well as $\mathsf{P}^0(M) = M$ and $\mathsf{P}^i(M) = \mathsf{P}(\mathsf{P}^{i-1}(M))$ for $i > 0$. Let $\mathsf{P}^*(M) = \bigcup_{i\in\mathbb{N}_0} \mathsf{P}^i(M)$. We sometimes switch between the set $\mathsf{P}^*(M)$ and its predicate analogue, with $\mathsf{P}^*(M, m') = 1$ iff $m' \in \mathsf{P}^*(M)$. Unless mentioned differently, we assume that any predicate can be evaluated efficiently.

We also presume, without further mentioning it, that predicates are *monotone*, that is, $\mathsf{P}(M') \subseteq \mathsf{P}(M)$ if $M' \subseteq M$. It follows inductively that $\mathsf{P}^*(M') \subseteq \mathsf{P}^*(M)$ in this case as well. This is necessary to ensure that, below in the unforgeability game, the set of messages for which a signature can be trivially derived from known signatures for M, does not shrink by asking for more signatures.[1] An alternative is to consider below all subsets $M' \subseteq M$ and declare that any message which is in $\mathsf{P}(M')$ to be a message for which a signature is trivial to derive from the signatures for messages in M'.

We again consider both the static and the dynamic case simultaneously, with the understanding that the predicate is fixed in the static case via $\mathcal{P} = \{\mathsf{P}\}$.

Definition 2 (Unforgeability). *A (statically or dynamically) adjustable $\mathcal{P}$-homomorphic signature scheme (KeyGen, Sign, $\mathsf{SignDer}$, Verify) is called* unforgeable*, if any PPT adversary $\mathcal{A}$ has a negligible advantage in the following game:*

1. *The challenger $\mathcal{C}$ generates the key pair $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives pk to the adversary $\mathcal{A}$. The challenger initializes two empty sets T and Q.*
2. *$\mathcal{A}$ interleaves adaptively the following queries:*
   - *Signing queries: $\mathcal{A}$ chooses a message $m \in \mathcal{M}$ and a predicate $\mathsf{P} \in \mathcal{P}$, upon which $\mathcal{C}$ returns a unique handle h to $\mathcal{A}$, runs $\sigma \leftarrow \mathsf{Sign}(sk, m, \mathsf{P})$, and stores $(h, m, \sigma, \mathsf{P})$ in T.*
   - *Derivation queries: $\mathcal{A}$ chooses a set of handles $\boldsymbol{h} = \{h_i\}_i$, a message $m' \in \mathcal{M}$ and a predicate P. The challenger $\mathcal{C}$ retrieves the tuples $(h_i, m_i, \sigma_i, \mathsf{P}_i)$ from T and returns $\perp$ if one of these tuples does not exist, $\mathsf{P}_i \neq \mathsf{P}$ for some i, or $\mathsf{P}(M, m') = 0$. Otherwise, the challenger returns a unique handle h' to $\mathcal{A}$, runs $\sigma' \leftarrow \mathsf{SignDer}(pk, M, \{\sigma_m\}_{m\in M}, m', \mathsf{P})$ for $M = \{m_i\}_i$ and stores $(h', m', \sigma', \mathsf{P})$ in T.*
   - *Reveal queries: If $\mathcal{A}$ chooses a handle h then $\mathcal{C}$ returns $\perp$ if there does not exist a tuple of the form $(h, m, \sigma, \mathsf{P})$ in T. Otherwise, it returns σ to $\mathcal{A}$ and adds (m, σ, P) to the set Q.*
3. *$\mathcal{A}$ outputs a pair (m, σ, P) and wins if the following conditions hold:*
   - *$\mathsf{Verify}(pk, m, \sigma, \mathsf{P}) = 1$, and*
   - *$m \notin \mathsf{P}^*(M_\mathsf{P})$, where $M_\mathsf{P} = \{m \in \mathcal{M} \mid (m, *, \mathsf{P}) \in Q\}$, the set of messages in the query set Q for the same predicate P.*

Note that the condition on $m \notin \mathsf{P}^*(M_\mathsf{P})$ can be relaxed by considering the set M of messages which have been signed under *some* predicate (and not only those which have been signed under the same predicate P as in the forgery attempt). In the static case both cases coincide, of course.

[1] Interestingly, this is not stipulated explicitly in previous works [1,4]. Still, the predicates for the constructions there satisfy this property. It is, nonetheless, generally required for a reasonable definition in order to avoid trivial examples of schemes which are formally unforgeable, but intuitively insecure.

2.3 Context Hiding

The original definition of Ahn et al. [1] requires a strong privacy requirement, basically saying that a derived signature (from previously signed messages M), and a fresh signature for the new message m' are statistically close. It follows that a derived signature does not leak any information about the starting messages M, and thus implies other common privacy notions for, say, redactable signature schemes [9]. Still, the notion has been strengthened in [4] to adaptive context hiding and complete context hiding, basically saying that derived signatures (for messages with any valid signatures) and fresh signatures are close. The generalization to valid signatures as input, instead of only signed messages, allows to cover previously excluded cases like rerandomizable signatures.

While the notion of adaptive context hiding is game-based, the notion of complete context hiding is defined through statistically close distributions of signatures. It is convenient for us here to present the latter definition also through a game, but considering *unbounded* adversaries (as opposed to *efficient* adversaries for adaptive context hiding). Otherwise the notions are identical. Our game-based definition of complete context hiding can be seen easily to be equivalent to the distributional approach in [4].

Definition 3 ((Complete and Adaptive) Context Hiding). *A (statically or dynamically) adjustable $\mathcal{P}$-homomorphic signature scheme ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignDer}, \mathsf{Verify}$) is called* completely (resp. adaptively) context hiding*, if any unbounded (resp. PPT) adversary $\mathcal{A}$ has a negligible advantage in the following game:*

1. *The challenger $\mathcal{C}$ generates the key pair $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives (sk, pk) to the adversary $\mathcal{A}$.*
2. *The adversary selects a set M of messages, and set $\{\sigma_m\}_{m\in M}$ of signatures, a predicate $\mathsf{P} \in \mathcal{P}$, and a message m' and hands it to the challenger. If $\mathsf{P}(M, m') = 0$ or if $\mathsf{Verify}(pk, M, \{\sigma_m\}_{m\in M}, \mathsf{P}) = 0$ then the challenger immediately returns $\perp$. Else it picks a random bit $b \leftarrow \{0, 1\}$ and computes a derived siganture $\sigma' \leftarrow \mathsf{SignDer}(pk, M, \{\sigma_m\}_{m\in M}, m', \mathsf{P})$ if $b = 0$, and a fresh signature $\sigma' \leftarrow \mathsf{Sign}(sk, m', \mathsf{P})$ in case $b = 1$. It returns σ' to the adversary.*
3. *Eventually the adversary outputs a bit $b^* \in \{0, 1\}$ and wins if $b^* = b$. The advantage of $\mathcal{A}$ is defined to be $\mathbf{Adv}(\mathcal{A}) = \left|\mathrm{Prob}[b^* = b] - \frac{1}{2}\right|$.*

Some remarks are in place. First note that the adversary can ask the challenger only once. A standard hybrid argument shows that this remains true for multiple (polynomially many) queries for which the challenger re-uses the same bit b. For both cases, the static and the dynamic one, the advantage grows by a factor proportional to the number of queries.

Secondly, note that in the dynamically adjustable case we do not aim to hide the predicate P which has been used to compute the signature. In a stronger requirement one could demand that the actual predicate remains hidden, either among all predicates from the universe, or among the predicates for which the public derivation algorithm would succeed. The former would require a super-polynomial set $\mathcal{P}$ (else the privacy attacker could probe the derivation algorithms for all predicates). The latter would mean a trade-off between privacy, usability, and the signer's intention for restricting the class of admissible public operations: if the signature would hide the corresponding predicate among multiple possibilities, then signatures for a different predicate than the original choice may be derivable. This would imply that the signer loses some control about the (in)ability to derive further signatures. Hence, we do not pursue such stronger requirements here.

3 Statically Adjustable Computations

In this section we investigate statically adjustable constructions for the basic operations AND, OR, and NOT. As explained in the introduction, we can give a general solution for AND, but cannot hope to give (general) transformations for the other two cases.

Below we consider combinations for arbitrary functions f over a fixed[2] number q of predicates $\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_q$. We assume that such a function $f(\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_q)$ over the predicates itself constitutes a predicate and defines a set of derivable messages from M in a straightforward way, by evaluating the predicates for (M, m') and plugging the results into the formula. If viewed as sets, our basic examples for OR, AND, and NOT can then be written as $f_\vee(\mathsf{P}_1, \mathsf{P}_2)(M) = \mathsf{P}_1(M) \cup \mathsf{P}_2(M)$, and $f_\wedge(\mathsf{P}_1, \mathsf{P}_2)(M) = \mathsf{P}_1(M) \cap \mathsf{P}_2(M)$, as well as $f_\neg(\mathsf{P}_1)(M) = \mathcal{M} \setminus \mathsf{P}_1(M)$.

Note that one could more generally also define $f(\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_q)$ for divisible message sets $M = (M_1, M_2, \ldots, M_q)$ by evaluating $f(M, m')$ as a logical formula over $\mathsf{P}_1(M_1, m'), \ldots, \mathsf{P}_q(M_q, m')$, i.e., assigning only the i-th part M_i of M to the i-th predicate, instead of using the same set M for all predicates. This can be captured in our notion with a single M by having the predicates P_i first project M onto M_i and then evaluating the actual predicate on (M_i, m'). For sake of readability we use the simpler notion with identical M.

We also assume that the message spaces $\mathcal{M}_i$ of all schemes are identical. This can always be achieved by setting $\mathcal{M} = \bigcap_{i=1}^{q} \mathcal{M}_i$. Note that, if message spaces are not identical this in principle allows to distinguish, say, in case of OR which predicate can be used to create a signature for some message. Since this would violate the idea of privacy immediately, we restrict ourselves to the case of identical message spaces.

3.1 Statically Adjustable Computations for AND

We first confirm that the solution to sign each message component-wise under a set of public keys yields a secure solution for the AND. Instead of considering only two predicates we allow to combine any fixed number q of predicates.

[2] Note that, in general, the number of combined predicates is specific for the scheme and must not depend on the security parameter, i.e., the design of the scheme does not change with the security parameter. In this sense the number q of predicates is constant in the security parameter.

Construction 1 (AND-Construction). *Let ($\mathsf{KeyGen}_i, \mathsf{Sign}_i, \mathsf{SignDer}_i, \mathsf{Verify}_i$) be P_i-homomorphic signature schemes for predicates $\mathsf{P}_1, \ldots, \mathsf{P}_q$. Then the following scheme ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignDer}, \mathsf{Verify}$) is a P-homomorphic signature scheme for $\mathsf{P} = \mathsf{P}_1 \wedge \ldots \wedge \mathsf{P}_q$:*

- *$\mathsf{KeyGen}(1^\lambda)$ runs $(sk_i, pk_i) \leftarrow \mathsf{KeyGen}_i(1^\lambda)$ for all $i = 1, 2, \ldots, q$, and outputs $sk = (sk_1, \ldots, sk_q)$ and $pk = (pk_1, \ldots, pk_q)$.*
- *$\mathsf{Sign}(sk, m, \mathsf{P})$ computes $\sigma_i \leftarrow \mathsf{Sign}_i(sk_i, m, \mathsf{P}_i)$ for all i and returns $\sigma = (\sigma_1, \ldots, \sigma_q)$.*
- *$\mathsf{SignDer}(pk, M, \Sigma, m', \mathsf{P})$ first checks that $\mathsf{P}_i(M, m') = 1$ for all i, and then creates $\sigma'_i \leftarrow \mathsf{SignDer}_i(pk_i, M, \Sigma_i, m', \mathsf{P}_i)$ where Σ_i is the set of projections on the i-th component for each signature tuple in $\Sigma = \{\sigma_m\}_{m \in M}$. It returns $\sigma' = (\sigma'_1, \ldots, \sigma'_q)$.*
- *$\mathsf{Verify}(pk, M, \Sigma, \mathsf{P})$ returns 1 if and only if $\mathsf{Verify}_i(pk_i, M, \Sigma_i, \mathsf{P}_i) = 1$ for all i (where again Σ_i is the set of projections on the i-th component for each signature in Σ).*

Correctness follows easily from the correctness of the underlying P_i-homomorphic schemes.

Proposition 1. *For any constant q and any unforgeable and completely (resp. adaptively) context-hiding P_i-homomorphic schemes, Construction 1 (AND-Construction) is unforgeable and completely (resp. adaptively) context-hiding.*

For concrete parameters our proof shows that the advantage of breaking unforgeability resp. context hiding for the AND scheme is bounded by the sum of the advantages for the corresponding property over all P_i-homomorphic schemes.

Proof. We first show unforgeability, then context hiding.

Unforgeability Assume that there exists a successful adversary $\mathcal{A}$ against unforgeability (Definition 2) for the P-homomorphic signature scheme where $\mathsf{P} = \mathsf{P}_1 \wedge \ldots \wedge \mathsf{P}_q$. For each $i \in \{1, 2, \ldots, q\}$, we first construct an adversary $\mathcal{A}_i$ against the unforgeability of the underlying P_i-homomorphic signature schemes:

- $\mathcal{A}_i$ initially receives pk_i from the challenger $\mathcal{C}_i$ for the game against the P_i-homomorphic signature schemes.
- $\mathcal{A}_i$ creates an initially empty table T' and runs $(sk_j, pk_j) \leftarrow \mathsf{KeyGen}_j(1^\lambda)$ for all $j = 1, 2, \ldots, q$, $j \neq i$ to create the other keys.
- $\mathcal{A}_i$ invokes adversary $\mathcal{A}$ against the AND-scheme on $pk = (pk_1, \ldots, pk_q)$.
- For every signing query (m, P) from $\mathcal{A}$, adversary $\mathcal{A}_i$ creates a signing query for message m and the predicate P_i for its challenger and gets the handle h, then computes $\sigma_j \leftarrow \mathsf{Sign}_j(sk_j, m, \mathsf{P}_j)$ for all $j \neq i$, and stores $(j, h, m, \sigma_j, \mathsf{P}_j)$ in T'.
- For every derivation query $(\{h\}, m', \mathsf{P})$ of $\mathcal{A}$, adversary $\mathcal{A}_i$ passes a derivation query for the corresponding handles $(\{h\}, m', \mathsf{P}_i)$ to its challenger to receive a handle h'. If $h' \neq \perp$ adversary $\mathcal{A}_i$ looks up all entries $(j, h, m, \sigma_m, \mathsf{P}_j)$ for

$j \neq i$ in T' for the queried handles in $\{h\}$ to form $M = \{m\}$, internally checks $\mathsf{P}_j(M, m') = 1$, and computes $\sigma'_j \leftarrow \mathsf{SignDer}_j(pk_j, M, \{\sigma_m\}_{m \in M}, m', \mathsf{P}_j)$. If no error occurs it returns h' to $\mathcal{A}$ and stores $(j, h', m', \sigma'_j, \mathsf{P}_j)$ in T' for all $j \neq i$; else it returns $\perp$.
– For every reveal request $\mathcal{A}_i$ runs a reveal request for the corresponding handle h, combines the reply σ_i with the values σ_j from entries $(j, h, m, \sigma_j, \mathsf{P}_j)$ in T' to σ and sends it to $\mathcal{A}$; in case of an error it simply returns $\perp$.
– When $\mathcal{A}$ eventually outputs a tuple (m, σ, P), then $\mathcal{A}_i$ outputs the tuple $(m, \sigma_i, \mathsf{P}_i)$ for the i-th component σ_i in σ.

Note that for each i adversary $\mathcal{A}_i$ perfectly simulates an attack of $\mathcal{A}$ on the P-homomorphic scheme with the help of its challenger, such that $\mathcal{A}$ would output a successful forgery with the same probability in the simulation as in the original attack. By construction, we also have that the message set M_{P} of queries $(m, *, \mathsf{P})$ in $\mathcal{A}$'s queries in the simulation is identical to the set M_{P_i} for queries $(m, *, \mathsf{P}_i)$ of $\mathcal{A}_i$ to its challenger for each i. Hence, from $m \notin \mathsf{P}^*(M_{\mathsf{P}})$ it follows that $m \notin \mathsf{P}_i^*(M_{\mathsf{P}_i})$ for some $i \in \{1, 2 \ldots, q\}$. Furthermore, since verification succeeds for all components, it also holds that $\mathsf{Verify}_i(pk_i, m, \sigma, \mathsf{P}_i) = 1$ for this i.

In other words, any successful forgery yields a successful forgery against (at least) one of the underlying schemes. It follows that the probability of breaking unforgeability for the AND scheme is bounded from above by the sum of the probabilities to break each underlying scheme.

Context Hiding. Assume next that there exists a successful adversary $\mathcal{A}$ against context hiding (Definition 3) for our P-homomorphic signature scheme with $\mathsf{P} = \mathsf{P}_1 \wedge \ldots \wedge \mathsf{P}_q$. As in the case of unforgeability we construct, for each $i \in \{1, 2, \ldots, q\}$, an adversary $\mathcal{A}_i$ against context hiding of the i-th scheme. The advantage of $\mathcal{A}$ will be bounded from above by the sum over all advantages of the $\mathcal{A}_i$'s via a standard hybrid argument. Furthermore, each $\mathcal{A}_i$ will be efficient if $\mathcal{A}$ is, such that the claim remains true for adaptive context hiding.

Adversary $\mathcal{A}_i$ receives a pair (sk_i, pk_i) from its challenger and creates the other key pairs (sk_j, pk_j) for $j \neq i$ by running $\mathsf{KeyGen}_j(1^\lambda)$. It hands $sk = (sk_1, \ldots, sk_q)$ and $pk = (pk_1, \ldots, pk_q)$ to adversary $\mathcal{A}$ and waits for the adversary to create a challenge request M, Σ, m'. For each signature σ_m in Σ adversary $\mathcal{A}_i$ extracts the i-th component and thereby forms the set Σ_i. It passes M, m', and Σ_i to its own challenger to receive a signature σ'_i (or an error message). It creates the signatures σ'_j for $j < i$ by running the signing algorithm on m'; for $j > i$ it runs the signature derivation algorithm on M, m', Σ_j to create the remaining signatures σ'_j. In all cases it checks the validity of the predicates and signatures. If there is an error it returns $\perp$ to the adversary $\mathcal{A}$, and $(\sigma'_1, \ldots, \sigma'_q)$ otherwise. If $\mathcal{A}$ eventually outputs a bit b^* then $\mathcal{A}_i$, too, outputs this bit and stops.

For the analysis note that $\mathcal{A}_1$, given that its challenger uses $b = 0$, describes the case that all signatures are derived via $\mathsf{SignDer}$. It follows that the probability of $\mathcal{A}$ correctly outputting 0 for derived signatures in the attack (and thus in the perfect simulation through $\mathcal{A}_1$) is exactly the probability that $\mathcal{A}_1$ returns 0, given $b = 0$ in its challenge. Analogously, given $b = 1$ adversary $\mathcal{A}_q$ only creates fresh signatures via Sign in all components, hence given $b = 1$ the probability that $\mathcal{A}_q$ returns 0 is exactly the same that $\mathcal{A}$ outputs 0 in the case that all signatures are fresh. A standard hybrid argument now yields: $\mathbf{Adv}(\mathcal{A}) = \sum_{i=1}^{q} \mathbf{Adv}(\mathcal{A}_i)$. This proves context hiding. □

3.2 Statically Adjustable Computations for OR and NOT

Our impossibility result holds for canonical constructions which combine P_i-homomorphic schemes in a general way, ruling out specific constructions which ignore the underlying schemes and builds a new scheme from scratch. We require four algorithms, one for synthesizing public keys of the individual schemes into one for the combined scheme (PKComb), one for splitting keys ($\mathsf{PKSplit}$), one for combining signatures ($\mathsf{SigComb}$), and one to divide signatures for the combined scheme into signatures for the individual schemes ($\mathsf{SigSplit}$). The latter is usually necessary to reduce the security to the security of the individual schemes.

For sake of readability we follow the statistical indistinguishability approach also used for (complete) context hiding, and require that the distributions of the algorithms above for combining and splitting keys and signatures have identical distributions as if running the actual algorithms of the combined scheme directly. As our proof below shows our impossibility result can be extended to cover computationally indistinguishable distributions.

Definition 4 (Canonical Construction). *Let f be a functional predicate over predicates $\mathsf{P}_1, \ldots, \mathsf{P}_q$ for a fixed number q of predicates. A statically adjustable $f(\mathsf{P}_1, \ldots, \mathsf{P}_q)$-homomorphic signature scheme ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignDer}, \mathsf{Verify}$) is a canonical construction out of P_i-homomorphic signature schemes ($\mathsf{KeyGen}_i, \mathsf{Sign}_i, \mathsf{SignDer}_i, \mathsf{Verify}_i$) if there exist PPT algorithms ($\mathsf{PKComb}, \mathsf{PKSplit}, \mathsf{SigComb}, \mathsf{SigSplit}$) such that:*

Identical distribution of combined keys: *The following random variables are identically distributed:*
- *Let $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and output pk;*
- *Let $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}_i(1^\lambda)$ for all i, $pk \leftarrow \mathsf{PKComb}(pk_1, \ldots, pk_q)$, output pk,*

Identical distribution of split keys: *The following random variables are identically distributed:*
- *Let $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and output $(pk_1, \ldots, pk_q) \leftarrow \mathsf{PKSplit}(pk)$;*
- *Let $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}_i(1^\lambda)$ for all i, output $(pk_1, \ldots, pk_q)$,*

Identical distribution of combined signatures: *For any PPT algorithm $\mathcal{F}$ the following pairs of random variables are identically distributed:*
- *Run $M \leftarrow \mathcal{F}(1^\lambda)$. Compute $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and output $\Sigma \leftarrow \mathsf{Sign}(sk, M, f(\mathsf{P}_1, \ldots, \mathsf{P}_q))$;*
- *Run $M \leftarrow \mathcal{F}(1^\lambda)$. For all i, compute $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}_i(1^\lambda)$ along with $\Sigma_i \leftarrow \mathsf{Sign}_i(sk_i, M, \mathsf{P}_i)$. Synthesize the public key via $pk \leftarrow \mathsf{PKComb}(pk_1, \ldots, pk_q)$ and output $\Sigma \leftarrow \mathsf{SigComb}(pk, pk_1, \ldots, pk_q, \Sigma_1, \ldots, \Sigma_q, M)$.*

Splitting Signatures: *For any PPT algorithm $\mathcal{F}'$ we have that for $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}_i(1^\lambda)$ for all i, $pk \leftarrow \mathsf{PKComb}(pk_1, \dots, pk_q)$, $(M, m') \leftarrow \mathcal{F}'(1^\lambda)$ where $m' \in f(\mathsf{P}_1, \dots, \mathsf{P}_q)(M)$, $\Sigma_i \leftarrow \mathsf{Sign}_i(sk_i, M, \mathsf{P}_i)$, $\Sigma \leftarrow \mathsf{SigComb}(pk_1, \dots, pk_q, \Sigma_1, \dots, \Sigma_q, M)$, $\sigma' \leftarrow \mathsf{SignDer}(pk, M, \Sigma, m', f(P_1, \dots, P_q))$, the probability that $(\sigma_1', \dots, \sigma_q') \leftarrow \mathsf{SigSplit}(pk, pk_1, \dots, pk_q, m', \sigma')$ does not contain some valid component and thus $\mathsf{Verify}_i(pk_i, m', \sigma_i') = 0$ for all i, is negligible.*

In other words, SigSplit returns at least one valid signature for one of the underlying predicates with sufficiently high probability. Our AND-construction is canonical in the above sense: PKComb and SigComb both concatenate their inputs (and PKSplit divides the concatenated keys again), and SigSplit simply returns the signature itself. Note that the definition allows PKComb, PKSplit, SigComb, and SigSplit to depend on the given predicates P_i; the construction only follows a canonical pattern.

In what follows, we need to exclude trivial examples like $\mathsf{P}_1 \vee \mathsf{P}_2 = \mathsf{P}_1 \vee \mathsf{P}_1$. Hence, for the OR we assume below the existence of a message m' which can be derived from a set of messages M under one predicate, but not the other predicate. This clearly prevents $\mathsf{P}_1 = \mathsf{P}_2$. More generally, and to include for instance also the NOT case, we assume that m' can be derived under $f(\mathsf{P}_1, \mathsf{P}_2, \dots)$ but not under one of the predicates; the excluded predicate P_i can be arbitrary, but the output distribution of m' does not depend on this choice. The latter is necessary to ensure that m' does not contain any information about the predicate's index i. Furthermore, we assume that such pairs (M, m') are efficiently computable. We discuss an illuminating example after the definition.

Definition 5 (Efficiently Distinguishable Predicates). *Let f be a functional predicate over predicates $\mathsf{P}_1, \dots, \mathsf{P}_q$. Consider a statically adjustable $f(\mathsf{P}_1, \dots, \mathsf{P}_q)$-homomorphic signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignDer}, \mathsf{Verify})$. Then the predicates are called* efficiently distinguishable with respect to f, *if there exists a PPT algorithm $\mathcal{F}$ such that for any $i \in \{1, 2, \dots, q\}$ and for any $(M, m') \leftarrow \mathcal{F}(1^\lambda, i)$, we have $m' \in \big(f(\mathsf{P}_1, \dots, \mathsf{P}_q)(M) \setminus \mathsf{P}_i^*(M)\big)$. Moreover, for any $i, j \in \{1, 2, \dots, q\}$ the distribution of m' (over the coin tosses of $\mathcal{F}$) in the output of $\mathcal{F}(1^\lambda, i)$ resp. $\mathcal{F}(1^\lambda, j)$ is identical.*

Let us demonstrate the property for the introductory example of two redactable signature schemes (with message space $\mathcal{M} = \{0,1\}^*$), one allowing to drop message bits only at the front (predicate P_1), and the other one only at the end (P_2). Consider the OR predicate $\mathsf{P}_1 \vee \mathsf{P}_2$ describing a scheme for quotable substrings. Then $\mathcal{F}$ can simply pick $m' = 0^\lambda$ and for $i = 1$ output $M = \{0^\lambda 1\}$, and for $i = 2$ it returns $M = \{10^\lambda\}$ instead. Clearly, for $i = 1$ one can derive m' from M via P_2 and therefore for the OR, but not via P_1, because the '1' at the end cannot be redacted through P_1. The same argument holds vice versa for $i = 2$, and the (trivial) distributions on m' are identical for both $i = 1$ and $i = 2$. Hence, this examples has efficiently distinguishable predicates.

The case of NOT is even simpler. Algorithm $\mathcal{F}$ simply needs to find some M and some m' which lies in $(\neg\mathsf{P}(M)) \setminus \mathsf{P}^*(M) = \mathcal{M} \setminus \mathsf{P}^*(M)$, i.e., if m' is not derivable according to $\mathsf{P}^*(M)$. Finally note that constructions based only on AND cannot be distinguishable, since $(\mathsf{P}_1(M) \cap \mathsf{P}_2(M)) \setminus \mathsf{P}_i^*(M) = \emptyset$ for any i.

Theorem 1. *Let f be a functional predicate over predicates $P_1, \ldots, P_q$ for a fixed number q of predicates. Assume further that the predicates are efficiently distinguishable with respect to f. Then there is no adaptively context-hiding, statically-adjustable $f(P_1, \ldots, P_q)$-homomorphic signature scheme which is a canonical construction out of unforgeable P_i-homomorphic signature schemes.*

The proof idea is as follows. Essentially we show how to forge a signature for one of the underlying schemes. For this we use the distinguishability of the predicates to create a set of messages M and a message m' which is derivable by $f(\mathsf{P}_1, \ldots, \mathsf{P}_q)(M)$ but does not lie in $\mathsf{P}_i^*(M)$ for some i. Then we ask for signatures for the messages in M, and derive a signature for m' via the public operation SignDer for the combined scheme and for $f(\mathsf{P}_1, \ldots, \mathsf{P}_q)$. Splitting up the signature into its components via SigSplit we obtain (with sufficiently large probability) a valid signature for m' under the i-th scheme. But since $m' \notin \mathsf{P}_i^*(M)$ we thus create a valid forgery, contradicting the security of the underlying scheme. In the course of the proof we use the context hiding property to show that the "skewed" choice of M, m' (with $m' \notin \mathsf{P}_i^*(M)$) does not bias the success probability of SigSplit for returning a valid signature component for the i-th scheme significantly. The formal proof appears in the full version.

We stress that the impossibility result holds for the computational notion of *adaptive* context hiding (with efficient distinguishers), which even strengthens our result. As mentioned before, a slightly more involved argument allows to extend the result also to algorithms PKComb, PKSplit, SigComb whose output is only computationally indistinguishable from the one of the original algorithms (instead of being identical). This requires some additional steps to prove that gradually replacing the algorithms does not change the behavior of SigSplit in the above proof significantly.

4 Dynamically Adjustable Computations

In the dynamic case we assume a polynomial universe $\mathcal{P}$ of predicates such that there exists a P_i-homomorphic scheme for each $\mathsf{P}_i \in \mathcal{P}$. We furthermore assume that given (a description of) P_i one can efficiently recover the corresponding scheme, e.g., if the universe consists only of a fixed number of predicates. Vice versa, we assume that P_i is identifiable from the scheme's public key pk_i. This in particular implies that the public keys for predicates must be unique. For simplicity we assume an ordering on predicates in $\mathcal{P}$ and often identify the predicate P_i and the scheme with its number i according to this order. We simply call sets $\mathcal{P}$ as above *efficient*.

In the construction we need to assume that for a given predicate identifier i there is a fixed yet (pseudo)random key pair $(sk_i, pk_i) \leftarrow \mathsf{KeyGen}_i(1^\lambda)$, generated according to the key generation algorithm for the scheme for predicate P_i. This key pair remains identical for all signature requests for P_i. For a polynomial universe $\mathcal{P}$ this can be in principle implemented by generating the keys (sk_i, pk_i) when creating the scheme's keys (sk, pk), and storing them in sk. In practice this may indeed be admissible for a small number of predicates, a more applicable approach may be to generate the keys on the fly via a pseudorandom function. Namely, store a key κ of a pseudorandom function in sk, and to create the key pair for predicate P_i, recover the (pseudo)random output $\omega_i = \mathsf{PRF}(\kappa, \mathsf{P}_i)$ and re-run $\mathsf{KeyGen}_i(1^\lambda; \omega_i)$ for ω_i to derive the same pair (sk_i, pk_i) as before. For unforgeability it can be formally shown via standard techniques that this solution is (quasi) as secure as generating fresh key pairs and maintaining a table to look up previous keys; for context hiding, however, one requires an additional assumption on the security of the underlying scheme to preserve privacy, as discussed in the full version.

Similarly, the public keys pk_i and their (fixed) certificates $cert_i$ may be published at once, or may be attached to each signature upon creation. Below we adopt the latter solution as it rather complies with our notion of (stateless) $\mathcal{P}$-homomorphic signatures. Hence, below we assume for simplicity that the efficient universe $\mathcal{P}$ stores all pairs (sk_i, pk_i) with once-created certificates $cert_i$ at the beginning in sk. For certification we use a regular signature scheme which we can subsume as a special case under P-homomorphic schemes, without considering a SignDer algorithm nor context hiding. If we define $\mathsf{P}(M) = M$ for this scheme, unforgeability for this "homomorphic" scheme corresponds to the common notion of unforgeability for regular schemes.

Construction 2 (Certify-Then-Sign Construction). *Let $\mathcal{P}$ be an efficient set of predicates $\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_q$. Let $(\mathsf{KeyGen}_0, \mathsf{Sign}_0, \mathsf{Verify}_0)$ be a regular signature scheme. Define the following dynamically adjustable $\mathcal{P}$-homomorphic signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignDer}, \mathsf{Verify})$:*

- *$\mathsf{KeyGen}(1^\lambda)$ generates $(sk_0, pk_0) \leftarrow \mathsf{KeyGen}_0(1^\lambda)$, generates key pairs $(sk_i, pk_i) \leftarrow \mathsf{KeyGen}_i(1^\lambda)$ for all predicates P_i, and certificates $cert_i \leftarrow \mathsf{Sign}_0(sk_0, pk_i)$ for all i. It returns $sk = (sk_0, \{(sk_i, pk_i, cert_i)\}_i)$ and $pk = pk_0$.*
- *$\mathsf{Sign}(sk, m, \mathsf{P}_i)$ looks up $(sk_i, pk_i, cert_i)$ for P_i in sk and computes $\sigma_i \leftarrow \mathsf{Sign}_i(sk, m)$ and returns $\sigma = (\sigma_i, pk_i, cert_i)$.*
- *$\mathsf{SignDer}(pk, M, \Sigma, m', \mathsf{P}')$ checks that all signatures carry the same pk_i and $cert_i$ for predicate P_i, that $\mathsf{P}' = \mathsf{P}_i$, that $\mathsf{P}_i(M, m') = 1$, that $\mathsf{Verify}_i(pk_i, M, \Sigma) = 1$, and, if all checks succeed, computes $\sigma'_i \leftarrow \mathsf{SignDer}_i(pk_i, M, \Sigma, m')$ and returns $\sigma' = (\sigma'_i, pk_i, cert_i)$.*
- *$\mathsf{Verify}(pk, m, \sigma, \mathsf{P})$ checks that P corresponds to the public key in $(\sigma_i, pk_i, cert_i)$, that $\mathsf{Verify}_0(pk, pk_i, cert_i) = 1$, and that $\mathsf{Verify}_i(pk_i, m, \sigma_i) = 1$. Only if all checks succeed, it returns 1.*

It is straightforward to verify that the above construction is correct in the sense that genuine (fresh and derived) signatures are accepted by Verify. This follows from the correctness properties of the regular scheme and of the P_i-homomorphic ones.

Proposition 2. *Assume that the signature scheme $(\mathsf{KeyGen}_0, \mathsf{Sign}_0, \mathsf{Verify}_0)$ and all P_i-homomorphic schemes are unforgeable according to Definition 2. Then the*

Certify-then-Sign Construction 2 is also unforgeable for the efficient universe $\mathcal{P} = \{P_1, \ldots, P_q\}$ for the fixed number q of predicates.

In terms of concrete security, the success probability of any adversary against the construction is (for similar running time) bounded from above by the probability of forging certificates, plus q times the maximal advantage against any of the schemes from $\mathcal{P}$.

Proof. Assume that there exists a successful forger $\mathcal{A}$. Then this adversary is able to forge with non-negligible probability a signature $\sigma^* = (\sigma, pk, cert)$ for a message m such that, in particular, $\mathsf{Verify}_0(pk_0, pk, cert) = 1$. Note that if the probability that $\mathcal{A}$ succeeds and that pk does not match any of the keys pk_i created by the signer for the predicates P_i, was non-negligible, then this would straightforwardly contradict the unforgeability of the certification scheme. Namely, construct an algorithm $\mathcal{A}_0$ against the certification scheme which, on input pk_0, creates the polynomial number of key pairs $(sk_i, pk_i) \leftarrow \mathsf{KeyGen}_i(1^n)$ and asks for signatures $cert_i$ for all pk_i from the signing oracle, and then emulates the attack of $\mathcal{A}$ with the help of the secret keys. If $\mathcal{A}$ eventually outputs $\sigma^* = (\sigma, pk, cert)$, then $\mathcal{A}_0$ returns $pk, cert$ as the forgery attempt.

If the probability that $\mathcal{A}$ would succeed for a fresh pk with non-negligible probability as defined above, then our efficient algorithm $\mathcal{A}_0$, which perfectly simulates the actual attack, would then successfully forge a signature $cert$ for a new "message" pk with non-negligible probability. Since this would contradict the unforgeability of the certification scheme, we can assume that this case happens with negligible probability only. It follows that $\mathcal{A}$ must succeed with non-negligible probability for a key $pk = pk_i$ for some (unique) i, such that $\mathsf{Verify}_i(pk_i, m, \sigma, \mathsf{P}_i) = 1$, and the message is not trivially derivable under the corresponding predicate P_i from the signing queries for P_i.

Note that the specific choice pk_i may depend on the adversary's randomness. However, there must exist at least one predicate P_i (among the q schemes) such that $\mathcal{A}$ succeeds for this key fixed pk_i with non-negligible probability. We can now derive an adversary $\mathcal{A}_i$ successfully forging signatures for this P_i-homomorphic scheme. Adversary $\mathcal{A}_i$ receives from the challenger the public key pk_i and gets access to a Sign_i-oracle. It generates (sk_0, pk_0) and all other key pairs (sk_j, pk_j) and signs all of them, including pk_i. The adversary $\mathcal{A}_i$ then runs $\mathcal{A}$ on pk_0, supplying all signatures requests for $\mathsf{P}_j \neq \mathsf{P}_i$ with the help of the secret keys, and using the external signing oracle for P_i. If $\mathcal{A}$ finally returns m and $(\sigma, pk, cert)$ then $\mathcal{A}_i$ returns m and σ_i.

Note that, if $\mathcal{A}$ has a non-negligible success probability for forging under the key pk_i, then $\mathcal{A}_i$ has the same success probability. This follows as the signature verifies under pk_i, and if the message m is not derivable from $\mathcal{A}$'s queries for P_i, then this is also true for $\mathcal{A}_i$. This, however, would contradict the unforgeability assumption about the P_i-homomorphic scheme. $\square$

Proposition 3. *Assume that all P_i-homomorphic schemes are completely (resp. adaptively) context-hiding according to Definition 3. Then the Certify-*

then-Sign Construction 2 is also completely (resp. adaptively) context-hiding for an efficient universe $\mathcal{P} = \{P_1, P_2, \ldots, P_q\}$ of a fixed number q of predicates.

The proof is similarly to the unforgeability case and omitted for space reasons.

Acknowledgments. We thank the anonymous reviewers for comments. Marc Fischlin was supported by a Heisenberg grant Fi 940/3-1 of the German Research Foundation (DFG).

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 1–20. Springer, Heidelberg (2012)
2. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
3. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
4. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012)
5. Bellare, M., Neven, G.: Transitive signatures based on factoring and RSA. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 397–414. Springer, Heidelberg (2002)
6. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
7. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
8. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
9. Brzuska, C., et al.: Redactable signatures for tree-structured data: Definitions and constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)
10. Camacho, P., Hevia, A.: Short transitive signatures for directed trees. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 35–50. Springer, Heidelberg (2012)
11. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (2012)
12. Chang, E.-C., Lim, C.L., Xu, J.: Short redactable signatures using random trees. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 133–147. Springer, Heidelberg (2009)

13. Charles, D., Jain, K., Lauter, K.: Signatures for network coding. Int. J. Inf. Coding Theory 1(1), 3–14 (2009)
14. Desmedt, Y.: Computer security by redefining what a computer is. In: Proceedings of the 1992-1993 Workshop on New Security Paradigms, NSPW 1992-1993, pp. 1992–1993. ACM (1993)
15. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)
16. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
17. Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: Abe, M., Gligor, V. (eds.) ASIACCS 2008, pp. 353–362. ACM Press (March 2008)
18. Johnson, R., Walsh, L., Lamb, M.: Homomorphic signatures for digital photographs. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 141–157. Springer, Heidelberg (2012)
19. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
20. Micali, S., Rivest, R.L.: Transitive signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 236–243. Springer, Heidelberg (2002)
21. Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H.: Digital documents sanitizing problem. Technical Report ISEC2003-20. IEICE (2003)
22. Nojima, R., Tamura, J., Kadobayashi, Y., Kikuchi, H.: A storage efficient redactable signature in the standard model. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 326–337. Springer, Heidelberg (2009)
23. Shahandashti, S.F., Salmasizadeh, M., Mohajeri, J.: A provably secure short transitive signature scheme from bilinear group pairs. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 60–76. Springer, Heidelberg (2005)
24. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
25. Wang, L., Cao, Z., Zheng, S., Huang, X., Yang, Y.: Transitive signatures from braid groups. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 183–196. Springer, Heidelberg (2007)
26. Yi, X.: Directed transitive signature scheme. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 129–144. Springer, Heidelberg (2006)