2014 IEEE 27th Computer Security Foundations Symposium

Malleable Signatures: New Definitions and Delegatable Anonymous Credentials

Melissa Chase
Microsoft Research
Email: melissac@microsoft.com

Markulf Kohlweiss
Microsoft Research
Email: markulf@microsoft.com

Anna Lysyanskaya
Brown University
Email: anna@cs.brown.edu

Sarah Meiklejohn
UC San Diego
Email: smeiklej@cs.ucsd.edu

***Abstract*—A signature scheme is malleable if, on input a message and a signature, it is possible to efficiently compute a signature on a related message, for a transformation that is *allowed* with respect to this signature scheme. In this paper, we first provide new definitions for malleable signatures that allow us to capture a broader range of transformations than was previously possible. We then give a generic construction based on malleable zero-knowledge proofs that allows us to construct malleable signatures for a wide range of transformation classes, with security properties that are stronger than those that have been achieved previously. Finally, we construct delegatable anonymous credentials from signatures that are malleable with respect to an appropriate class of transformations (that we show our malleable signature supports). The resulting instantiation satisfies a stronger security notion than previous schemes while also scaling linearly with the number of delegations.**

***Index Terms*—signatures; malleability; anonymous credentials; delegation; definitions;**

I. Introduction

A signature scheme is malleable — alternatively, homomorphic — if, given a signature σ on a message m, one can efficiently derive a signature σ' on a message $m' = T(m)$ for an *allowed* transformation T. As an example, m' might be any excerpt from m, and indeed for appropriate classes of allowed transformations, malleable signatures provide a generalization of a variety of existing primitives, such as quotable and redactable signatures [1], [2]. Formal definitions for malleable signatures were first given by Ahn et al. [3] and later refined by Attrapadung et al. [4], and consider security with respect to two properties: unforgeability, in which an adversary can derive signatures on new messages using only allowed transformations of previously signed messages, and *context hiding*, in which signatures on derived messages should appear identical to fresh signatures issued by the signer; this latter property implicitly requires that the size of the signature depend only on the message, so they cannot grow as they are transformed, unless the transformed messages change as well.

More formally, the definition of Ahn et al. requires that an adversary cannot generate a signature on a message m' unless there exists some previously signed message m and an allowed transformation T such that $T(m) = m'$. While this definition captures a wide variety of allowed transformation classes, such as the ones for quotable and redactable signatures, there are other classes of transformations that it does not and cannot capture. Consider an example where Alice and Bob want to jointly generate an ElGamal encryption key (e.g., for threshold decryption). We could attempt to do this as follows: Alice chooses a random exponent a and sends $A = g^a$ to Bob, who chooses a random exponent b and publishes the joint key $pk = A^b$. As a result, a and b are shares of the corresponding secret key. If Bob is malicious, however, he can ignore A and publish $pk = g^b$; i.e., a public key for which he knows the entire secret key. One natural way to prevent this is to have Alice also sign A using a scheme that is malleable only with respect to exponentiation, and then require that pk is accompanied by a valid signature. Then, if Bob can produce such a valid signature, he must have computed it honestly.

Unfortunately, previous definitions of malleable signatures do not capture this intuition, and indeed this construction will not work with such definitions. In fact, *there is no way to meaningfully capture a signature malleable with respect to exponentiation under the previous definitions.* For our application, there always exists — for any g and pk — an exponent b such that $g^b = pk$, but what matters is whether or not the adversary *knows* this transformation. One of the main goals of our work is therefore to generalize previous definitions in a way that naturally allows for this type of transformation class.

As a more sophisticated application of malleable signatures, we consider an anonymous credential system [5], [6], [7]. In such a system, a user Alice is known in different contexts by different unlinkable pseudonyms, yet she can demonstrate possession of a credential issued to one pseudonym nym to a verifier who knows her by another pseudonym nym'. Let $\mathcal{T}$ be the set of transformations that, on input some pseudonym nym, output another pseudonym nym' of the same user, i.e. for every such pair $\mathsf{nym}, \mathsf{nym}'$, there exists some $T \in \mathcal{T}$ such that $\mathsf{nym}' = T(\mathsf{nym})$. Then a signature scheme that is malleable with respect to $\mathcal{T}$ gives us an anonymous credential system: a credential is an authority's signature σ on nym; malleability makes it possible for Alice to transform it into a signature σ' on nym'; context hiding ensures that σ' cannot be linked to the original pseudonym nym; and finally, unforgeability ensures that Alice cannot compute σ' unless she received a signature from the authority on one of her pseudonyms.

Somewhat surprisingly, not only can malleable signatures yield anonymous credentials, but, as we show in this paper, they can also yield *delegatable* anonymous credentials (DACs). A DAC system [8], [9] allows users to delegate their anonymous credentials; when presenting their credentials (and

DOI 10.1109/CSF.2014.22

also when obtaining and delegating them), the various participants need not reveal any persistent identifiers — or in fact anything — about themselves. DACs are much more privacy-friendly than the traditional anonymous credential model, which assumes that the verifying party knows the public key of the credential issuer(s), as who issued Alice's credentials reveals a lot of information about Alice. For example, the identity of the local DMV that issued her driver's license might reveal her zip code, and if her date of birth and gender are also leaked, this is often enough to uniquely identify her [10], [11], meaning verifiers that require multiple credentials might easily learn Alice's identity. Since DACs protect the identity of every link on Alice's certification chain, they make it impossible to infer any information about Alice based on who issued her credentials.

In order to construct a DAC from malleable signatures, we essentially follow the same outline as for the (non-delegatable) anonymous credential, but consider a different class of transformations. In the non-delegatable scenario, the transformation took as input Alice's nym and outputs Alice's nym'. In the DAC scenario, when Alice is delegating her credential to Bob, she uses a transformation T that takes as input her pseudonym nym_A and the length ℓ of her certification chain, and outputs Bob's pseudonym nym_B and the length $\ell + 1$ of his new certification chain; to be allowed, T's description must include Alice's secret key, so that only Alice can perform this transformation. Intuitively, this construction yields a DAC scheme, yet its security crucially relies on a transformation being allowed not by virtue of its input-output behavior, but by virtue of what its description contains. As a result, previous definitions of security for malleable and homomorphic signatures are not strong enough to be useful for this application: DACs require not only that an allowed transformation exist, but that the party applying it "know" its description. Additionally, to ensure Alice's privacy even in the face of adversarial root authorities, context hiding must hold even for adversarially-generated public keys for the signature scheme. This flavor of context hiding was not captured by previous definitions.

a) Our contributions: In this paper, we overcome the definitional obstacles explained above by first proposing, in Section III, new definitions for malleable signatures. Our definition of context hiding, extending that of Attrapadung et al., allows for adversarially-generated keys and signatures. Our unforgeability definition requires that the transformation T and the original message m that was signed by the signing oracle be extractable from (m', σ'). To ensure that these definitions are not overly strong, we relate them to the relevant definitions of Ahn et al. and Attrapadung et al. and observe that, for many classes of transformations, the definitions of unforgeability are equivalent (whereas working with adversarially-generated keys makes our definition of context hiding strictly stronger than their computational definitions).

With these new definitions in hand, we provide in Section IV a general construction of context-hiding malleable signatures for a large range of unary transformation classes. Our construction relies generically on non-interactive zero-knowledge (NIZK) proofs that provide controlled malleability; such proofs were recently defined and realized by Chase et al. [12]. Aside from its usefulness in our construction of DACs, our signature construction enjoys other nice properties. Although it is not the first construction of signatures from zero-knowledge proofs — the Fiat-Shamir heuristic [13] is an example of this approach, as are the signatures of knowledge due to Chase and Lysyanskaya [8] and the construction using PRFs due to Bellare and Goldwasser [14] — ours is the first such construction to achieve malleability. As for malleability, previous work gives *ad-hoc* constructions of malleable signatures for various classes of allowed transformations (such as redactable [15], [16], [1], [17], [2], sanitizable [18], [19], [20], quotable [21], and transitive signatures [22], [23]), but ours is the first *general* efficient construction of malleable signatures. The only previous work that gave a general approach to homomorphic signatures was by Ahn et al. [3], who gave (among other contributions, such as an efficient construction of a quotable signature) an *inefficient* general construction for which a malleable signature on m is essentially a set of non-malleable signatures on each message in the set $\{m' \mid m' = T(m) \wedge T \in \mathcal{T}\}$.

Finally, we follow the intuition developed above and construct, in Section V, a delegatable anonymous credentials scheme generically from a malleable signature and a commitment scheme. Our new definitions for malleable signatures also conveniently allow us to provide a new definition, presented in Section II-C, for credential unforgeability; our definition is both more powerful and considerably simpler than existing definitions. In addition to satisfying this new definition, our construction provides several desirable functional features (non-interactive delegation and the ability to delegate polynomially many times). Our construction is generic and can be instantiated either using succinct non-interactive arguments of knowledge (SNARGs) and homomorphic encryption [24], or using Groth-Sahai proofs [12]. The latter relies only on standard assumption (e.g., Decision Linear [25]).[1]

b) Related work on malleable signatures: Here we distinguish between work on unary and n-ary transformations. As mentioned above, some specific types of unary homomorphic signatures have been studied over the last decade or so, such as redactable and quotable signatures in which, given a signature on a document m, one can derive signatures on a redacted version of m in which some parts are blacked out, or signatures on quotations from m. These can be viewed as special motivating cases of context-hiding malleable signatures. A somewhat related type of signature is an incremental signature scheme [27], in which a signature on a document can be efficiently updated when the document is updated. Recent work on computing on authenticated data [3], [4] gives a general definitional framework for the problem (which we draw on in our definitions) and some general (but inefficient, as discussed

[1]The details of such a construction, see [26], are outside the scope of this work, and a precise performance analysis is still subject to future work.

above) constructions for unary transformations, as well as some efficient and elegant provably secure constructions for specific unary transformation classes, such as quoting and subsets.

Subsequent to our work, three other related papers have appeared. Boyle, Goldwasser, and Ivan [28] define *functional signatures*, where the signer can sign a function rather than a message; this has the effect of signing all messages in the range of the function, so the resulting signature can be transformed into a signature on any such message. Backes, Meiser, and Schröder [29] introduce *delegatable functional signatures* (DFS), in which they explicitly model the multi-user setting: each signature includes the public key of the party allowed to transform that signature, and a function describing how it can be transformed (and whether further transformations can be applied by other parties). The paper also offers a separation from one-way functions (which our notion inherits) and a construction based on trapdoor permutations. Neither of these papers use extraction-based definitions, and as such they are more related to the work of Ahn et al. Nevertheless, either functionality could be captured by our definitions using an appropriate choice of transformation classes (and in the case of DFS, commitments/hard relations as in our construction of DACs). In the third result, Bellare and Fuchsbauer [30] propose *policy-based signatures* (PBS) as a generalization of (among others) attribute-based signatures and group signatures, in which an authority first issues a signing key encoding a policy, and then this key can be used to sign any message satisfying this policy. Interestingly, they use a similar simulation and extraction approach in their privacy and unforgeability definitions, and two-tiered malleable signatures (i.e., signatures that allow at most one transformation to be applied) imply policy-based signatures. They also give a simple notion of delegation, in which many users can add restrictions to this policy; again, this could be constructed from a malleable signature for an appropriate transformation class. See Section VI-C for discussion of the relation to DFS and PBS.

Finally, research on n-ary transformations was initiated by work on transitive signatures [22], [23], and by Rivest in a series of talks; the first paper to address this subject more generally and consider several binary transformations was Johnson et al. [15]. A more recent line of work explored homomorphic signatures under linear and polynomial functions [31], [32], [33], [34], [35], [36], [37], [38]; these papers focus on transforming n message-signature pairs into a message-signature pair in which the message is a linear or polynomial function of the input messages, under somewhat weaker notions of security and privacy. This is somewhat incomparable to our work, as we are interested in more general transformations.

c) Related work on delegatable anonymous credentials: The first construction of delegatable anonymous credentials, by Chase and Lysyanskaya [8], allowed a constant number of delegations. Belenkiy et al. [9] gave the first DAC system that allowed for a polynomial number of delegations using Groth-Sahai proofs [39]; their construction, however, was somewhat ad-hoc and relied on ad-hoc assumptions. Finally, Fuchsbauer [40] gave a construction that built on the construction of Belenkiy et al. and allows for non-interactive issuing and delegation of credentials, also based on ad-hoc assumptions.

Our construction obtains many of the nicest features of each of these previous constructions. We support non-interactive issuing and delegation of credentials, our credentials scale linearly with the number of times they are delegated. Finally, we realize a simulation-extractable notion of delegatable anonymous credentials that is simpler and stronger than any of the previous definitions.

II. Preliminaries and Notation

As our construction of a malleable signature uses malleable proofs, we first discuss the definitions for such proofs here. We next recall existing definitions for delegatable anonymous credentials, and propose our new definition for credential unforgeability.

A. Standard definitions for zero-knowledge proofs of knowledge

Definition II.1. [12] *A set of algorithms* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ *constitute a non-interactive (NI) proof system for an efficient relation* R *with associated language* L_R *if completeness and soundness below are satisfied. A NI proof system is extractable if, in addition, the extractability property below is satisfied. A NI proof system is witness-indistinguishable (NIWI) if the witness-indistinguishability property below is satisfied. An NI proof system is zero-knowledge (NIZK) if the zero-knowledge property is satisfied. A NIZK proof system that is also extractable constitutes a non-interactive zero-knowledge proof of knowledge (NIZKPoK) system. A NIWI proof system that is also extractable constitutes a non-interactive witness-indistinguishable proof of knowledge (NIWIPoK) system.*

1) *Completeness [47]. For all* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$ *and* $(x, w) \in R$, $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ *for all proofs* $\pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, x, w)$.
2) *Soundness [47]. For all PPT* $\mathcal{A}$, *and for* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$, *the probability that* $\mathcal{A}(\mathsf{crs})$ *outputs* (x, π) *such that* $x \notin L$ *but* $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ *is negligible.* Perfect *soundness is achieved when this probability is 0.*
3) *Extractability [48]. There exists a PPT extractor* $E = (E_1, E_2)$ *such that* $E_1(1^k)$ *outputs* (crs_e, τ_e), *and* $E_2(\mathsf{crs}_e, \tau_e, x, \pi)$ *outputs a value* w *such that (1) any PPT* $\mathcal{A}$ *given* σ *cannot distinguish between the honest CRS and one output by* E_1*; i.e.,*

$$Pr[\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k) : \mathcal{A}(\mathsf{crs}) = 1] \approx Pr[(\mathsf{crs}_e, \tau_e) \xleftarrow{\$} E_1(1^k) : \mathcal{A}(\mathsf{crs}_e) = 1],$$

and (2) for all PPT $\mathcal{A}$, *the probability that* $\mathcal{A}$ *outputs* (x, π) *such that* $\mathcal{V}(\mathsf{crs}_e, x, \pi) = 1$ *but* $R(x, E_2(\mathsf{crs}_e, \tau_e, x, \pi)) = 0$ *is negligible; i.e., there exists a negligible*

function $\nu(\cdot)$ such that

$$Pr[(\mathsf{crs}_e, \tau_e) \xleftarrow{\$} E_1(1^k); \\ (x, \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}_e) : \\ \mathcal{V}(\mathsf{crs}_e, x, \pi) = 1 \wedge (x, E_2(\mathsf{crs}_e, \tau_e, x, \pi)) \notin R] < \nu(k).$$

Perfect *extractability is achieved if this probability is 0, and crs_e is distributed identically to* crs.

4) *Witness indistinguishability [49]. For all (x, w_1, w_2) such that $(x, w_1), (x, w_2) \in R$, any PPT $\mathcal{A}$ cannot distinguish between proofs for w_1 and proofs for w_2; i.e.,*

$$Pr[\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k); \\ (x, w_1, w_2) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}); \\ \pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, x, w_0) : \\ \mathcal{A}(\pi) = 1 \wedge (x, w_0), (x, w_1) \in R] \\ \approx Pr[\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k); \\ (x, w_1, w_2) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}); \\ \pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, x, w_1) \\ : \mathcal{A}(\pi) = 1 \wedge (x, w_0), (x, w_1) \in R].$$

Perfect *witness indistinguishability is achieved when these two distributions are identical.*

5) *Zero knowledge [49]. There exists a polynomial-time simulator algorithm $S = (S_1, S_2)$ such that $S_1(1^k)$ outputs (crs_s, τ_s), and $S_2(\mathsf{crs}_s, \tau_s, x)$ outputs a value π_s such that for all $(x, w) \in R$, a PPT adversary $\mathcal{A}$ cannot distinguish between proofs produced by the prover and simulator; i.e., for all PPT adversaries $\mathcal{A}$,*

$$Pr[\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k) : \mathcal{A}^{P(\mathsf{crs}, \cdot, \cdot)}(\mathsf{crs}) = 1] \\ \approx Pr[(\mathsf{crs}_s, \tau_s) \xleftarrow{\$} S_1(1^k) : \mathcal{A}^{S(\mathsf{crs}_s, \tau_s, \cdot, \cdot)}(\mathsf{crs}_s) = 1],$$

where, on input (x, w), P outputs $\perp$ if $(x, w) \notin R$ and $\pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, x, w)$ otherwise, and S also outputs $\perp$ if $(x, w) \notin R$, and returns $\pi \xleftarrow{\$} S_2(\mathsf{crs}_s, \tau_s, x)$ otherwise. Perfect *zero knowledge is achieved if for all $(x, w) \in R$, these distributions are identical.*

B. Definitions for malleable proofs

Let $R(\cdot, \cdot)$ be a relation such that the corresponding language $L_R := \{x \mid \exists w \text{ such that } (x, w) \in R\}$ is in NP. As defined for malleable proofs [12], the relation is *closed* with respect to a transformation $T = (T_{\text{inst}}, T_{\text{wit}})$ if for every $(x, w) \in R$, $(T_{\text{inst}}(x), T_{\text{wit}}(w)) \in R$. The formal definition of a malleable proof extends the definition of a non-interactive proof $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ by adding an additional algorithm ZKEval, designed to transform proofs. More formally, ZKEval, on input the CRS crs, a transformation T, an instance x and a proof π such that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$, outputs a proof π' for $x' := T_{\text{inst}}(x)$ such that $\mathcal{V}(\mathsf{crs}, x', \pi') = 1$. The proof system is then *malleable* with respect to a set of transformations $\mathcal{T}$ if for every $T \in \mathcal{T}$, ZKEval can be computed efficiently.

In addition to defining this basic notion of malleability, Chase et al. also defined how to meaningfully *control* the malleability of a proof system by extending the strong notion of simulation-sound extractability [45], [44] to deal with malleability; this means requiring that, for a set of transformations $\mathcal{T}$, if an adversary can produce a proof π for an instance x then the extractor should be able to extract from π either a witness w or a transformation $T \in \mathcal{T}$ and previous instance x' such that $x = T_{\text{inst}}(x')$ (the definition of simulation-sound extractability required only this first condition). More formally, this is defined as follows:

Definition II.2. [12] *Let $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ be a NIZKPoK system for an efficient relation R, with a simulator (S_1, S_2) and an extractor (E_1, E_2). Let $\mathcal{T}$ be a set of unary transformations for the relation R such that membership in $\mathcal{T}$ is efficiently testable. Let SE_1 be an algorithm that, on input 1^k, outputs $(\mathsf{crs}, \tau_s, \tau_e)$ such that (crs, τ_s) is distributed identically to the output of S_1. Let $\mathcal{A}$ be given, let $Q := Q_{inst} \times Q_{proof}$ be a table used to store the instances queried to S_2 and the proofs given in response, and consider the following game:*

- *Step 1.* $(\mathsf{crs}, \tau_s, \tau_e) \xleftarrow{\$} SE_1(1^k)$.
- *Step 2.* $(x, \pi) \xleftarrow{\$} \mathcal{A}^{S_2(\mathsf{crs}, \tau_s, \cdot)}(\mathsf{crs}, \tau_e)$.
- *Step 3.* $(w, x', T) \leftarrow E_2(\mathsf{crs}, \tau_e, x, \pi)$.
- *Step 4.* $b \leftarrow (w \neq \perp \wedge (x, w) \notin R) \vee ((x', T) \neq (\perp, \perp) \wedge (x' \notin Q_{inst} \vee x \neq T_{inst}(x') \vee T \notin \mathcal{T})) \vee (w, x', T) = (\perp, \perp, \perp))$.

The NIZKPoK satisfies controlled-malleable simulation-sound extractability *(CM-SSE, for short) with respect to $\mathcal{T}$ if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that the probability (over the choices of SE_1, $\mathcal{A}$, and S_2) that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ and $(x, \pi) \notin Q$ but $b = 1$ is at most $\nu(k)$.*

Definition II.3. [12] *For a non-interactive zero-knowledge proof system $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ with an associated simulation (S_1, S_2), an efficient relation R malleable with respect to $\mathcal{T}$, an adversary $\mathcal{A}$, and a bit b, let $p_b^{\mathcal{A}}(k)$ be the probability of the event that $b' = 0$ in the following game:*

- *Step 1.* $(\mathsf{crs}_s, \tau_s) \xleftarrow{\$} S_1(1^k)$.
- *Step 2.* $(\mathsf{state}, x_1, \pi_1, \ldots, x_q, \pi_q, T) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}_s, \tau_s)$.
- *Step 3. If $\mathcal{V}(\mathsf{crs}_s, x_i, \pi_i) = 0$ for some i, $1 \leq i \leq q$, or $T \notin \mathcal{T}$, abort and output $\perp$. Otherwise, form*

$$\pi \xleftarrow{\$} \begin{cases} S_2(\mathsf{crs}_s, T_{inst}(x_1, \ldots, x_q)) & \text{if } b = 0 \\ \mathsf{ZKEval}(\mathsf{crs}_s, T, \{x_i, \pi_i\}_{i=1}^q) & \text{if } b = 1. \end{cases}$$

- *Step 4.* $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \pi)$.

Then the proof system is strongly derivation private *if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.*

C. Delegatable anonymous credentials

At a high level, delegatable anonymous credentials (DAC) allow credentials to be both delegated and issued within the context of a system in which users are *pseudonymous*; i.e., they can use a different pseudonym for each of the different people with whom they interact. As such, algorithms are required for generating each of these pseudonyms, as well as issuing and delegating credentials, and proving (in an anonymous way) the possession of a credential.

In order to conceptually identify users with pseudonyms, we require that a nym output by NymGen is a commitment to the underlying sk with opening open, with verification algorithm NymVerify.

A delegatable anonymous credentials scheme consists of 8 algorithms (Setup, KeyGen, NymGen, NymVerify, Issue, CredProve, CredVerify, Delegate) that behave as follows:

- $\mathsf{Setup}(1^k)$: Generate public parameters pp for the system.
- $\mathsf{KeyGen}(pp)$: Generate public and secret keypair (pk, sk); the secret key sk represents a user's "true identity."
- $\mathsf{NymGen}(pp, sk)$: Compute a pseudonym nym for the user corresponding to sk together with a value open that can be used for verification.
- $\mathsf{NymVerify}(pp, \mathsf{nym}, sk, \mathsf{open})$: Check that a given pseudonym nym belongs to the user corresponding to sk. (In practice, this algorithm will never be run; instead, a user might form a proof of knowledge of sk, open corresponding to nym such that this holds.)
- $\mathsf{Issue}(pp, sk_0, pk_0, \mathsf{nym}_r)$: Issue a credential, rooted at the authority owning pk_0, to owner of nym_r.
- $\mathsf{CredProve}(pp, sk, \mathsf{nym}, \mathsf{open}, \mathsf{nym}', \mathsf{open}', \mathcal{C})$: Generate a proof π of possession of credential $\mathcal{C}$ that has been delegated to some nym, where the owner of nym also owns a pseudonym nym'.
- $\mathsf{CredVerify}(pp, pk_0, \mathsf{nym}, \ell, \pi)$: Verify that the pseudonym nym is in possession of a level-ℓ credential, rooted at pk_0.
- $\mathsf{Delegate}(pp, sk_{old}, \mathsf{nym}_{old}, \mathsf{open}_{old}, \mathsf{nym}_{new}, \mathcal{C})$: Delegate the credential $\mathcal{C}$, currently delegated to the pseudonym nym_{old}, to the pseudonym nym_{new}.

The main security requirements are anonymity and unforgeability. Briefly, anonymity requires that pseudonyms hide their owners' secret keys, and that a proof of possession of a credential does not reveal the pseudonym to which the credential was initially issued by the root authority, or the pseudonyms to which the credential was delegated (or any other information besides the final pseudonym, the original issuer, and the number of times the credential has been delegated). Unforgeability requires that one cannot prove possession or delegate a credential without knowing a secret key corresponding to some pseudonym to which a credential has been issued.

Our definition of anonymity is fairly similar to the definition given by Belenkiy et al.; the main modifications are a non-interactive Issue protocol and simulated parameters that are distributed identically to those output by Setup. Essentially, we require that there exist a simulator $(\mathsf{SimSetup}, \mathsf{SimCred}, \mathsf{SimProve})$ such that (1) SimSetup produces parameters and the simulation trapdoor, (2) SimCred produces credentials indistinguishable from those produced by Issue and Delegate (but without knowledge of the root secret key or a previous credential), and (3) SimProve produces proofs indistinguishable from those produced by CredProve (but without knowledge of a credential). For a formal definition of our anonymity property, as well as a definition of correctness, see our report [26].

Our definition of unforgeability, on the other hand, is a departure from that of Belenkiy et al. It is conceptually similar to the definition of simulation-sound extractability for non-interactive zero knowledge, in that we require that unforgeability hold in the presence of a simulator that grants and proves possession of credentials at any level for any pseudonym of the adversary's choice without access to the root authority's secret key. (In contrast, Belenkiy et al. use different parameters for unforgeability and anonymity; in their construction under the simulation parameters extraction is impossible.)

Formally, we define an *augmented setup algorithm* SimExtSetup that produces (pp, τ_s, τ_e) such that (pp, τ_s) is distributed identically to the output of SimSetup. We then have the following definition.

Definition II.4 (Unforgeability)**.** *A delegatable anonymous credentials scheme* (Setup, KeyGen, NymGen, NymVerify, Issue, CredProve, CredVerify, Delegate) *with simulator* (SimSetup, SimCred, SimProve) *is* unforgeable *if there exists a pair* (SimExtSetup, Extract) *(where* SimExtSetup *augments* SimSetup*) such that (1)* nym *is a hiding, binding commitment when* pp *are chosen by* SimExtSetup*, even when* τ_s *and* τ_e *are given; and (2) it is hard to form a proof of a credential for a pseudonym* nym *at level* ℓ *without knowing the secret key corresponding to* nym *as well as the secret key corresponding to some* nym_1 *to which a credential at level* $\ell' \leq \ell$ *has been issued; formally, for any adversary* $\mathcal{A}$*, consider the following game, wherein* $C(\cdot,\cdot) = \mathsf{SimCred}(pp, \tau_s, vk_0, \cdot, \cdot)$*,* $P(\cdot,\cdot) = \mathsf{SimProve}(pp, \tau_s, vk_0, \cdot, \cdot)$*, and* Extract *share state:*

- *Step 1.* $(pp, \tau_s, \tau_e) \xleftarrow{\$} \mathsf{SimExtSetup}(1^k)$*;* $(vk_0, sk_0) \xleftarrow{\$} \mathsf{KeyGen}(pp)$.
- *Step 2.* $((\mathsf{nym}, \ell), \pi) \xleftarrow{\$} \mathcal{A}^{C(\cdot,\cdot), P(\cdot,\cdot)}(pp, vk_0, \tau_e)$.
- *Step 3.* $(\{(\mathsf{nym}_i, sk_i, \mathsf{open}_i)\}_{i=1}^k, \ell') \leftarrow \mathsf{Extract}(pp, \tau_e, (vk_0, \mathsf{nym}, \ell), \pi)$*, where* $\mathsf{nym}_k = \mathsf{nym}$ *and* $\ell' + k - 2 = \ell$.[2]

Then for all PPT algorithms $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (over the choices of* SimExtSetup*,* SimCred*,* SimProve*, and* $\mathcal{A}$*) that* $\mathsf{CredVerify}(pp, vk_0, \mathsf{nym}, \ell, \pi) = 1$ *and* (nym, ℓ) *was not queried to* SimProve *but either*

1) $\mathcal{A}$ *created a new credential; i.e.,* (nym_1, ℓ') *was not queried to* SimCred *or* $\ell < \ell'$*,*

[2]The decrement -2 results from the original recipient of the credential and the final owner of π being counted twice in $\ell' + k$. The latter is counted twice as he can change his pseudonym from nym_{k-1} to nym.

2) *$\mathcal{A}$ delegated through pseudonyms it did not own; i.e.,* $\mathsf{NymVerify}(pp, \mathsf{nym}_j, sk_j, \mathsf{open}_j) = 0$ *for some* j, $1 \leq j \leq k$, *or* $k \neq 2 + \ell - \ell'$,
3) *$\mathcal{A}$ proved possession for a credential it did not own; i.e.,* $sk_{k-1} \neq sk_k$,

is at most $\nu(k)$.

III. Defining Malleable Signatures

Formally, a malleable signature scheme consists of four algorithms: KeyGen, Sign, Verify, and SigEval. The first three comprise a standard signature; the additional algorithm, SigEval, on input the verification key vk, messages $\vec{m} = (m_1, \ldots, m_n)$, signatures $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, and a transformation T on messages, outputs a signature σ' on the message $T(\vec{m})$. (Here and in all of our definitions, we consider the most general case wherein the transformation may combine many messages. Our construction in Section IV, however, supports only unary transformations; i.e., those operating on a single message.)

Definition III.1 (Malleability)**.** *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* malleable *with respect to a set of transformations* $\mathcal{T}$ *closed under composition if there exists an efficient algorithm* SigEval *that on input* $(vk, T, \vec{m}, \vec{\sigma})$*, where* $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$*,* $\mathsf{Verify}(vk, \sigma_i, m_i) = 1$ *for all* i*, and* $T \in \mathcal{T}$*, outputs a valid signature* σ' *for the message* $m := T(\vec{m})$*; i.e., a signature* σ' *such that* $\mathsf{Verify}(vk, \sigma', m) = 1$.

Our definition has one key notational difference with the previous definitions of Ahn et al. [3] and Attrapadung et al. [4]: whereas their definitions were given with respect to a predicate on input and output messages, we found it more natural to instead consider the set of allowed *transformations*. By using transformations, we inherently capture the dual requirements that the result of an operation be efficiently computable, and that an adversary *know* the transformation that was applied; these requirements could also potentially be captured using predicates (e.g., by using an optional witness input to the predicate, as is briefly mentioned in [3]), but in a more roundabout way.

A. Simulation-based definitions for malleable signatures

We begin with the idea of a *simulatable* signature scheme, introduced by Abe et al. [41], [42], [43], in which there are two indistinguishable ways of producing a signature: using the signing key and the standard signing algorithm, or using a global trapdoor and a simulated signing algorithm. Using simulatable signatures allows us to easily consider the notion of context hiding with respect to adversarially-chosen keys, as well as a simpler notion for signature unforgeability. (Ahn et al. use statistical context hiding to achieve a simpler version of their unforgeability definition; since extraction and statistical hiding are contradictory, we cannot.) Simulatability also lines up nicely with the anonymity requirements for credentials, in which there should exist a simulator that can simulate credentials.

Before we present our definition of simulatability — which is somewhat modified from the original; in particular they required only that simulated signatures verify, whereas we require them to be indistinguishable from standard signatures — we must expand the standard notion of a signature scheme to consider signature schemes in which the key generation process is split up into two parts: a trusted algorithm Gen for generating "universal" parameters crs (we can think of these as the setting; e.g., the description of a group), and an algorithm KeyGen that, given these parameters, generates a keypair (vk, sk) specific to a given signer.

Definition III.2 (Simulatability)**.** *A signature scheme* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* simulatable *if there exists an additional PPT algorithm* KeyCheck *that, on input* crs*,* vk*, and* sk*, outputs whether or not* (vk, sk) *is in the range of* $\mathsf{KeyGen}(\mathsf{crs})$*, and a PPT simulator* $(\mathsf{SimGen}, \mathsf{SimSign})$ *such that the CRS in* $(\mathsf{crs}, \tau_s) \xleftarrow{\$} \mathsf{SimGen}(1^k)$ *is indistinguishable from* $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}(1^k)$ *and signatures produced by* SimSign *are indistinguishable from honest signatures; i.e., for all PPT* $\mathcal{A}$*,*

$$Pr[\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}(1^k) : \mathcal{A}^{S(\mathsf{crs},\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1]$$
$$\approx Pr[(\mathsf{crs}, \tau_s) \xleftarrow{\$} \mathsf{SimGen}(1^k) : \mathcal{A}^{S'(\mathsf{crs},\tau_s,\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1],$$

where, on input (vk, sk, m)*,* S *outputs* $\perp$ *if* $\mathsf{KeyCheck}(\mathsf{crs}, vk, sk) = 0$ *and* $\mathsf{Sign}(\mathsf{crs}, sk, m)$ *otherwise, and* S' *outputs* $\perp$ *if* $\mathsf{KeyCheck}(\mathsf{crs}, vk, sk) = 0$ *and* $\mathsf{SimSign}(\mathsf{crs}, \tau_s, vk, m)$ *otherwise.*

Although simulatability might seem to be a fairly strong property, we show in Section VI that any non-simulatable signature scheme in the standard model can be easily transformed into a simulatable signature scheme in the CRS model (i.e., using split Gen and KeyGen algorithms) by adding a proof of knowledge to the public key. Our signature construction in Section IV achieves simulatability directly by using cm-NIZKs.

B. Simulation context hiding

We next present a definition of context hiding that requires transformed signatures to be indistinguishable from freshly simulated signatures on the transformed messages; if regular signatures were used instead of simulated signatures, this would be quite similar to the standard notion of context hiding. As mentioned above, however, incorporating simulatability allows us to easily build in the notion of adversarially-generated keys, which is essential for our credentials application where the issuer need not be trusted for anonymity.

Definition III.3 (Simulation context hiding)**.** *For a simulatable signature* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigEval})$ *with an associated simulator* $(\mathsf{SimGen}, \mathsf{SimSign})$*, malleable with respect to a class of transformations* $\mathcal{T}$*, and an adversary* $\mathcal{A}$ *and a bit* b*, let* $p_b^{\mathcal{A}}(k)$ *be the probability of the event that* $b' = 0$ *in the following game:*

- *Step 1.* $(\mathsf{crs}, \tau_s) \xleftarrow{\$} \mathsf{SimGen}(1^k)$.
- *Step 2.* $(\mathsf{state}, vk, \vec{m}, \vec{\sigma}, T) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \tau_s)$.

- *Step 3. If* $\mathsf{Verify}(\mathsf{crs}, vk, \sigma_i, m_i) = 0$ *for some* i *or* $T \notin \mathcal{T}$*, abort and output* $\perp$*. Otherwise, either form* $\sigma \xleftarrow{\$} \mathsf{SimSign}(\mathsf{crs}, \tau_s, vk, T(\vec{m}))$ *if* $b = 0$*, or form* $\sigma \xleftarrow{\$} \mathsf{SigEval}(\mathsf{crs}, vk, T, \vec{m}, \vec{\sigma})$ *if* $b = 1$.
- *Step 4.* $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \sigma)$.

Then the signature scheme satisfies simulation context hiding *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.

Unsurprisingly, allowing for adversarially-generated keys yields a strictly stronger definition than any of the existing ones for computational context hiding (see Section VI). Finally, while Ahn et al. and Attrapadung et al. both provide statistical variants on context hiding, we cannot hope to simultaneously achieve statistical hiding and an unforgeability definition based on a meaningful notion of extraction.

C. Simulation unforgeability

The main point at which our unforgeability definition diverges significantly from previous definitions is in considering how to check whether a message and signature are in fact a forgery, or whether they were instead obtained using a valid transformation from the signatures issued by the signer. As discussed in the introduction, previous definitions consider the adversary successful only if he produces a signature on a message m for which *there does not exist* a previously signed message m' and an allowed transformation T such that $m = T(m')$. On the other hand, we want to be able to capture the requirement that the adversary "know" a valid transformation from a previously signed message; thus the adversary should be successful if he can produce a signature on m without *knowing* an appropriate m' and T. This allows us to meaningfully capture transformations like exponentiation, and is also crucial in our credential application, since credentials inherently require that signatures can't be transformed without knowledge of some appropriate secret information.

We formalize this notion by requiring an extractor that — given the produced message m and signature, as well as the set of signed messages — produces a previously signed message m' and the transformation (if one exists) that was used to obtain m; then the adversary wins if the extractor fails to produce a valid pair m', T. We note that this is also convenient in that it makes the winning conditions of the unforgeability game efficiently checkable. For simple classes of transformations it may be easy to determine whether there exists a valid transformation given the set of signed messages, but for classes of transformations that are exponentially (or even infinitely) large, it is not clear that this can be done in an efficient manner. With our new definition, on the other hand, determining whether the adversary won is as simple as checking if the extracted transformation T is valid, if the extracted m' we previously signed, and if $T(m') = m$.

By using simulatability, as defined above, we are able to replace all honestly generated and transformed signatures with signatures generated by a simulator; this means that we can simply give the adversary access to an oracle that generates simulated signatures, which has the advantage that we end up with a much cleaner definition than the main one of Ahn et al. (they also provide a similar simplification using the notion of statistical context hiding). The result is a definition similar to that of simulation-sound extractability [44], [45], in which we simulate and extract at the same time.

To formalize this game, we require an amplified setup, $\mathsf{SimExtGen}$, that outputs a tuple $(\mathsf{crs}, \tau_s, \tau_e)$; we then require the (crs, τ_s) part of this to be distributed identically to the output of SimGen, and we give τ_e as input to SigExt. To cover the case of simple — but potentially n-ary — transformations, where not all the information about a transformation can be encoded in the signature, the extractor is also given access to the query history Q.

Definition III.4 (Simulation unforgeability)**.** *For a simulatable signature* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigEval})$ *with an associated PPT simulator/extractor* $(\mathsf{SimExtGen}, \mathsf{SimSign}, \mathsf{SigExt})$ *that is malleable with respect to a class of transformations* $\mathcal{T}$*, an adversary* $\mathcal{A}$*, and a table* $Q = Q_m \times Q_\sigma$ *that contains messages queried to* $\mathsf{SimSign}$ *and their responses, consider the following game:*

- *Step 1.* $(\mathsf{crs}, \tau_s, \tau_e) \xleftarrow{\$} \mathsf{SimExtGen}(1^k)$*;* $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{crs})$.
- *Step 2.* $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{SimSign}(\mathsf{crs}, \tau_s, vk, \cdot)}(\mathsf{crs}, vk, \tau_e)$.
- *Step 3.* $(\vec{m}', T) \leftarrow \mathsf{SigExt}(\mathsf{crs}, vk, \tau_e, m^*, \sigma^*, Q)$.

Then the signature scheme satisfies simulation unforgeability *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (over the choices of* KeyGen*,* $\mathsf{SimSign}$*, and* $\mathcal{A}$*) that* $\mathsf{Verify}(vk, \sigma^*, m^*) = 1$ *and* $(m^*, \sigma^*) \notin Q$ *but either (1)* $\vec{m}' \not\subseteq Q_m$*, (2)* $m^* \neq T(\vec{m}')$*, or (3)* $T \notin \mathcal{T}$ *is at most* $\nu(k)$*.*

Having argued that our definition is significantly stronger than previous definitions, one might be concerned that our definition might be overly restrictive, in the sense that it might rule out previous constructions or many interesting classes of transformations. As we show in Section VI, however, when considering many transformation classes, including essentially all those for which constructions are known, our definition of unforgeability is equivalent to that of Ahn et al. (with respect to simulatable signatures which, as mentioned above, can be easily and generically obtained from non-simulatable signatures). Thus, although our definition does rule out certain classes of transformations (i.e., those where the transformation can neither be extracted nor efficiently derived given the query list and some limited amount of extra information), to date this does not seem to be a significant limitation on the schemes we can construct. On the other hand, as we will see, it allows us to meaningfully capture a wide variety of additional transformations, resulting in new applications.

IV. Malleable Signatures from cm-NIZKs

In this section, we provide a generic construction of malleable signatures from cm-NIZKs. As discussed earlier, this

allows us to capture a broad class of allowable unary transformations. After presenting our construction of delegatable anonymous credentials in the next section, we also see in Section V-C how to instantiate the signature concretely for a particular class of transformations.

Intuitively, our construction is extremely simple: to sign a message, just prove knowledge of the signing key! While this might seem to produce signatures that are independent of the message being signed, we show that by including the message in the instance, we can bind the message and signature together (as was also done, e.g., by Chase and Lysyanskaya [8]); furthermore, defining transformations on signatures is quite straightforward as well, since signatures in our construction are just malleable proofs. Formally, we use a hard relation $\mathcal{R}_{pk}$ with generator $\mathcal{G}$ and a cm-NIZK $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$, malleable with respect to some class of transformations $\mathcal{T}_{\text{nizk}}$, for the relation R such that $((pk, m), sk) \in R$ if and only if $(pk, sk) \in \mathcal{R}_{pk}$. We then construct a simulatable signature, malleable with respect to a class of transformations $\mathcal{T}_{\text{sig}}$, as follows:

- $\mathsf{Gen}(1^k)$: Output $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$.
- $\mathsf{KeyGen}(\mathsf{crs})$: Compute $(pk', sk') \xleftarrow{\$} \mathcal{G}(1^k)$; output $vk := pk'$ and $sk := (pk', sk')$.
- $\mathsf{Sign}(\mathsf{crs}, sk, m)$: Parse $sk = (pk', sk')$; output $\sigma = \pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, (pk', m), sk')$.
- $\mathsf{Verify}(\mathsf{crs}, vk, \sigma, m)$: Output $\mathcal{V}(\mathsf{crs}, (vk, m), \sigma)$.
- $\mathsf{SigEval}(\mathsf{crs}, vk, T, m, \sigma)$: Set T_{inst} to be such that $\forall m, vk,\ T_{\text{inst}}(vk, m) = (vk, T(m))$, and $T_{\text{wit}} = \text{id}$; i.e., such that $\forall sk',\ T_{\text{wit}}(sk') = sk'$. Return $\sigma' \xleftarrow{\$} \mathsf{ZKEval}(\mathsf{crs}, (T_{\text{inst}}, T_{\text{wit}}), (vk, m), \sigma)$.

Looking at the definition of SigEval, we can see that if we wished to construct a signature malleable with respect to a specific class of transformations $\mathcal{T}_{\text{sig}}$, we would require a proof malleable with respect to the class $\mathcal{T}_{\text{nizk}}$ consisting of all transformations of the form $(T_{\text{inst}} = (\text{id}, T), T_{\text{wit}} = \text{id})$ for $T \in \mathcal{T}_{\text{sig}}$.

Theorem IV.1. *If* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ *is zero knowledge, then* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigEval})$ *is simulatable, as defined in Definition III.2.*

Theorem IV.2. *If* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ *is strongly derivation private with respect to* $\mathcal{T}_{nizk}$*, as defined in Definition II.3, then* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigEval})$ *satisfies simulation context hiding with respected to* $\mathcal{T}_{sig}$*, as defined in Definition III.3.*

Proof: To show this, we take an adversary $\mathcal{A}$ that can break simulation context hiding with some non-negligible advantage ϵ and use it to construct an adversary $\mathcal{B}$ that breaks strong derivation privacy with the same advantage.

To start, $\mathcal{B}$ will get as input a pair (crs, τ_s), which it then immediately forwards to $\mathcal{A}$. When $\mathcal{A}$ outputs its challenge (pk', sk', m, σ, T), $\mathcal{B}$ first checks that $\mathsf{Verify}((\mathsf{crs}, pk'), \sigma, m) = 1$, $\mathsf{KeyCheck}(\mathsf{crs}, pk', sk') = 1$ and $T \in \mathcal{T}$; it aborts and outputs $\perp$ if any of these checks fails. Otherwise, it sets $T_{\text{inst}} := T$ and $T_{\text{wit}} := \text{id}$ and outputs $(x := (pk', m), \sigma, (T_{\text{inst}}, T_{\text{wit}}))$ as its own challenge to get back a proof π', which it again forwards directly (as σ') to $\mathcal{A}$. Finally, when $\mathcal{A}$ outputs its guess bit b', $\mathcal{B}$ outputs the same bit.

To see that interactions with $\mathcal{B}$ are distributed identically to those that $\mathcal{A}$ expects, we observe that the pair (crs, τ_s) given to $\mathcal{A}$ is honestly computed. In addition, if $\mathsf{Verify}((\mathsf{crs}, pk'), \sigma, m) = 1$ then, by definition, $\mathcal{V}(\mathsf{crs}, (pk', m), \sigma) = 1$; furthermore, if $T \in \mathcal{T}$ then, again by definition, $(T_{\text{inst}}, T_{\text{wit}}) \in \mathcal{T}_{\text{nizk}}$, meaning that if $\mathcal{B}$ interprets $\mathcal{A}$'s challenge tuple as valid, its own challenge tuple will be interpreted as valid as well. As for the response, if $b = 0$ then $\mathcal{B}$ gets back $S_2(\mathsf{crs}, \tau_s, (pk', m)) = \mathsf{SimSign}(vk = (\mathsf{crs}, pk'), \tau_s, m)$, which is exactly what $\mathcal{A}$ expects. Furthermore, if $b = 1$, then $\mathcal{B}$ gets back $\mathsf{ZKEval}(\mathsf{crs}, (T_{\text{inst}}, T_{\text{wit}}), (pk', m), \sigma) = \mathsf{SigEval}((\mathsf{crs}, pk'), T, m, \sigma)$, which is again exactly what $\mathcal{A}$ was expecting. As $\mathcal{A}$ therefore has the same advantage interacting with $\mathcal{B}$ as it does normally, and furthermore $\mathcal{B}$ succeeds in guessing b whenever $\mathcal{A}$ does, $\mathcal{B}$ succeeds with non-negligible advantage ϵ. ■

Theorem IV.3. *If* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ *is CM-SSE with respect to the class of transformations* $\mathcal{T}_{nizk}$ *and* $\mathcal{R}_{pk}$ *is a hard relation, then* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigEval})$ *is simulation unforgeable with respect to transformation class* $\mathcal{T}_{sig}$*, as defined in Definition III.4.*

Proof: To prove this, we first define the algorithms SimExtGen and SigExt, using the algorithms SE_1 and E_2 that, by CM-SSE, we know exist for the proof. SimExtGen simply outputs $(\mathsf{crs}, \tau_s, \tau_e) \xleftarrow{\$} SE_1(1^k)$; because the (crs, τ_s) output by SE_1 are, by definition, distributed identically to those output by S_1, and furthermore SimGen runs S_1, SimExtGen satisfies the constraint that its own (crs, τ_s) must be distributed identically to that of SimGen. Now, SigExt, given (vk, τ_e, m, σ), runs $E_2(\mathsf{crs}, \tau_e, (vk, m), \sigma)$ to get back a tuple $(w, x', (T_{\text{inst}}, T_{\text{wit}}))$; if $w = \perp$, it then parses $x' = (pk', m')$ and $T_{\text{inst}} = T$ and outputs (m', T), otherwise it outputs $(\perp, \perp)$. We also use the same simulator SimSign as in the proof of Theorem IV.1.

We now observe that, if there exists an adversary $\mathcal{A}$ that can break unforgeability with some non-negligible probability ϵ, it must be the case that one of two events occurs with some non-negligible probability: in the first, Event_1, for $(w, x', T) \leftarrow E_2(\mathsf{crs}, (pk', m^*), \sigma^*)$, $w \neq \perp$ and $(x, w) \in R$. In the second, Event_2, we consider all winning cases in the CM-SSE game; i.e., $(x, w) \notin R$, x' was not queried to the S_2 oracle, $x \neq T_{\text{inst}}(x')$, or $T \notin T$. We define e_1 to be the probability that Event_1 occurs, and e_2 to be the probability that Event_2 occurs. To see that, in the event that $\mathcal{A}$ wins the game, either e_1 or e_2 must be non-negligible (i.e., either Event_1 or Event_2 must have taken place), we observe that for $\mathcal{A}$ to have won, it must be the case that, for $(m', T) := \mathsf{SigExt}(vk, sk, m, \sigma)$, either m' wasn't queried to SimSign, $T \in \mathcal{T}$ and $m^* \neq T(m')$,

or $T \notin \mathcal{T}$, or $(m', T) = (\perp, \perp)$. Based on the definition of SigExt, this first case happens in the event that $x' = (pk', m')$ wasn't queried to S_2, which is part of Event_2. The second case happens in the event that $(x', T) \neq (\perp, \perp)$ and $x \neq T(x')$, which is also part of Event_2. The third case, on the other hand, might happen in the event that $(x', T) \neq (\perp, \perp)$ and $T \notin \mathcal{T}$, or in the event that $(w, x', T) = (\perp, \perp, \perp)$; these are once again part of Event_2. Finally, we get $(m', T) = (\perp, \perp)$ in the event that $w \neq \perp$; in this case, if $(x, w) \in R$ then we are in Event_1, and if $(x, w) \notin R$ then we are in Event_2. As each winning case for $\mathcal{A}$ therefore implies that either Event_1 or Event_2 has taken place, it must be the case that either e_1 or e_2 is non-negligible.

To first use Event_1 to break the hardness of the relation, $\mathcal{B}$ receives as input a public key pk' for R_{pk}. It then generates $(\mathsf{crs}, \tau_s, \tau_e) \xleftarrow{\$} SE_1(1^k)$ and gives crs, τ_e and pk' to $\mathcal{A}$. When $\mathcal{A}$ queries its $\mathsf{SimSign}$ oracle, $\mathcal{B}$ uses its knowledge of τ_s to execute the code honestly; Finally, when $\mathcal{A}$ outputs its pair (m^*, σ^*), $\mathcal{B}$ computes $(w, x', T) := E_2(\mathsf{crs}, \tau_e, (pk', m^*), \sigma^*)$. If Event_1 has occurred, then by definition $w \neq \perp$ and $((pk', m^*), w) \in R$; by definition of the relation R, this means that $(pk', w) \in R_{pk}$, and thus $\mathcal{B}$ can output w to win its game. As $\mathcal{B}$ behaves honestly and interactions with $\mathcal{B}$ are thus distributed identically to those that $\mathcal{A}$ expects, and further $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does and Event_1 occurs, $\mathcal{B}$ succeeds with overall probability $e_1\epsilon$.

To use Event_2 to break CM-SSE, $\mathcal{C}$ receives as input the pair (crs, τ_e); it then generates $(pk', sk') \xleftarrow{\$} \mathcal{G}(1^k)$ and gives crs, τ_e and pk' to $\mathcal{A}$. When $\mathcal{A}$ queries its $\mathsf{SimSign}$ oracle on input m, $\mathcal{C}$ queries its own S_2 oracle on input (pk', m) and returns the resulting proof back to $\mathcal{A}$. Now, when $\mathcal{A}$ outputs its pair (m^*, σ^*), $\mathcal{C}$ computes $(w, x', T) := E_2(\mathsf{crs}, \tau_e, (pk', m^*), \sigma^*)$. If Event_2 has occurred, then by definition one of the winning cases for the CM-SSE game holds, and thus $\mathcal{C}$ can output $(x := (pk', m^*), \sigma^*)$ to win its game. As $\mathcal{C}$ generates vk honestly and returns values of the form $S_2(\mathsf{crs}, \tau_s, (pk', m)) = \mathsf{SimSign}(vk = (\mathsf{crs}, pk'), \tau_s, m)$, interactions with it are distributed identically to those that $\mathcal{A}$ expects; furthermore, $\mathcal{C}$ succeeds whenever $\mathcal{A}$ does and Event_2 occurs, so $\mathcal{C}$ succeeds with overall probability $e_2\epsilon$. As ϵ is assumed to be non-negligible, and by our discussion above so is either e_1 or e_2, the success probability of either $\mathcal{B}$ or $\mathcal{C}$ is therefore non-negligible as well. ■

V. Delegatable Anonymous Credentials from Malleable Signatures

Recall the desired functionality of a credential system: there are various users, each in possession of some secret key, who can use different pseudonyms to represent themselves to other participants; to form a pseudonym, a user Alice computes a commitment to her secret key sk_A. A *credential authority* (CA) publishes a public key pk, and can issue a level-1 credential to some user Bob, known to it by the pseudonym B_1. Bob might now wish to do one of two things with this credential: he can either prove possession of this credential to another user, who might know him under a different pseudonym B_2, or he can *delegate* this credential to another user, Carol, whom he knows under the pseudonym C_1. To delegate, Bob can give to Carol a level-2 credential belonging to her pseudonym C_1, that is "rooted" at the authority pk; Bob should only be able to perform this operation if he is the rightful owner of the pseudonym B_1 to which the credential was issued. In order to fit this framework, credentials must therefore reveal three pieces of information: the root authority, denoted pk_0, the recipient of the credential, denoted nym_r, and the level of the credential, denoted ℓ.

A. Allowable transformations

First, let us describe all of the components in the above landscape in terms of commitments, messages, signatures, and transformations. We already saw that pseudonyms are represented as commitments; this leaves delegation and issuing of credentials, which are represented using transformations, and credentials and proofs of possession, which are both represented using messages and signatures.

The messages that are signed are of the form $m = (\mathsf{nym}, \ell, \mathsf{flag})$, where nym is the owner of the credential, ℓ is its level, and flag represents whether it is a credential or a proof of possession. Intuitively, to create an initial credential, the root authority signs the message $m = (\mathsf{nym}, 1, \mathsf{cred})$, using a signing key pair (vk, sk), to obtain a signature σ. The credential is the tuple $(vk, \mathsf{nym}, 1, \sigma)$, and anyone can verify that this is a valid credential by running the signature verification algorithm $\mathsf{Verify}(vk, \sigma, (\mathsf{nym}, 1, \mathsf{cred}))$. To delegate, the user corresponding to nym can maul the signature σ — using $\mathsf{SigEval}$ — to specify a new recipient nym_{new} and increase the level to 2. To prove possession of the credential under a different pseudonym nym', the user can maul the signature to change nym to nym' and cred to proof.

In order to delegate a credential belonging to nym to another pseudonym nym_{new}, one must know (sk, open) that corresponds to nym. Thus, the description of a valid transformation that takes as input the message $m = (\mathsf{nym}, \ell, \mathsf{cred})$ and outputs $T(m) = (\mathsf{nym}', \ell + 1, \mathsf{cred})$ must include (sk, open). In order to prove possession of a credential under a different pseudonym nym', one needs to know $(sk, \mathsf{open}, \mathsf{open}')$ such that (sk, open) correspond to nym and (sk, open') correspond to nym'. Thus, the description of a valid transformation that takes as input the message $m = (\mathsf{nym}, \ell, \mathsf{cred})$ and outputs $T(m) = (\mathsf{nym}', \ell, \mathsf{proof})$ must include $(sk, \mathsf{open}, \mathsf{open}')$ such that nym corresponds to (sk, open) and nym' corresponds to (sk, open').

To delegate a level-ℓ credential the user with pseudonym nym thus applies a transformation $\langle T \rangle = ((\mathsf{nym}, sk, \mathsf{open}), \mathsf{nym}_{new}, \mathsf{cred})$ such that $T(\mathsf{nym}, \ell, \mathsf{cred})$ outputs $(\mathsf{nym}_{new}, \ell + 1, \mathsf{cred})$ and on any other input T outputs $\perp$. To generate a proof the user applies a transformation $\langle T \rangle = ((\mathsf{nym}, sk, \mathsf{open}), \mathsf{nym}_{new}, \mathsf{proof})$ such that $T(\mathsf{nym}, \ell, \mathsf{cred})$ outputs $(\mathsf{nym}', \ell, \mathsf{proof})$ and on any other input T outputs $\perp$, where cred means the transformation

corresponds to delegation and proof means the transformation corresponds to proving possession.

As mentioned in Definition III.1, we require that the transformation class be closed under composition, so we also consider transformations that take a level-ℓ credential and output a level-$(\ell+k)$ credential; this means that a description is of the form $\langle T \rangle = (\{(\mathsf{nym}_j, sk_j, \mathsf{open}_j)\}_{j=1}^{k}, \mathsf{nym}_{new}, \mathsf{flag}')$, and $T(\mathsf{nym}, \ell, \mathsf{flag})$

$$:= \begin{cases} (\mathsf{nym}_{new}, \ell + k, \mathsf{cred}) \\ \qquad \text{if } \mathsf{nym}_1 = \mathsf{nym} \text{ and } \mathsf{flag} = \mathsf{flag}' = \mathsf{cred} \\ (\mathsf{nym}_{new}, \ell + (k-2), \mathsf{proof}) \\ \qquad \text{if } \mathsf{nym}_1 = \mathsf{nym}, \mathsf{flag} = \mathsf{cred}, \text{ and } \mathsf{flag}' = \mathsf{proof} \\ \bot \\ \qquad \text{otherwise.} \end{cases}$$

Thus, the set of allowable transformations $\mathcal{T}_{\text{dac}}$ consists of transformations whose description is $\langle T \rangle = (\{(\mathsf{nym}_j, sk_j, \mathsf{open}_j)\}_{j=1}^{k}, \mathsf{nym}_{new}, \mathsf{flag}')$, whose input/output behavior is as above, and such that

1) A user needs a pseudonym to which a credential was issued before the user can delegate or prove possession of it: $k > 0$.
2) A user can only delegate a credential he owns; i.e., he must know the opening of the pseudonym to which it was issued: for commitment parameters pp', $\mathsf{nym}_j = \mathsf{Com}(pp', sk_j; \mathsf{open}_j)$ for all $1 \le j \le k$.
3) If this is a proof of possession, meaning $\mathsf{flag}' = \mathsf{proof}$, then $k \ge 2$ and $\langle T \rangle$ must include the opening of nym_{new}, so we require $\mathsf{nym}_k = \mathsf{nym}_{new}$. Additionally, the owner of nym_{new} must be the same as the owner of the pseudonym to which the credential was delegated, so we require $sk_k = sk_{k-1}$.

In terms of credential size, we can see that messages scale logarithmically with the number of levels (as they need to represent the integer ℓ), while the size of the description of a transformation scales linearly, as does the size of our credentials. Since credentials should (as part of their functionality) explicitly reveal how many times they have been delegated, some dependence seems inevitable. As evident for transferable e-cash [46], we conjecture that the strong extraction requirements of our definitions require delegatable credentials to grow linearly in the number of delegations.

B. Our construction

Our construction follows the intuition developed above: to form a pseudonym, a user forms a commitment to a signing secret key; to issue a credential, the root authority signs the recipient's pseudonym and the intended level of the credential; and to delegate and prove possession of a credential, a user mauls the credential (i.e., signature) using one of the allowable transformations defined above. Formally, we use a simulatable signature $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigEval})$, malleable with respect to $\mathcal{T}_{\text{dac}}$, and a commitment scheme $(\mathsf{ComSetup}, \mathsf{Com})$ as follows:

- $\mathsf{Setup}(1^k)$: Compute $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}(1^k)$ and $pp' \xleftarrow{\$} \mathsf{ComSetup}(1^k)$. Output $pp := (\mathsf{crs}, pp')$.
- $\mathsf{KeyGen}(pp)$: Output $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$.
- $\mathsf{NymGen}(pp, sk)$: Pick a random opening open, compute $\mathsf{nym} := \mathsf{Com}(pp', sk; \mathsf{open})$, and output $(\mathsf{nym}, \mathsf{open})$.
- $\mathsf{NymVerify}(pp, \mathsf{nym}, sk, \mathsf{open})$:
  Check that $\mathsf{nym} = \mathsf{Com}(pp', sk; \mathsf{open})$;
  output 1 if this holds and 0 otherwise.
- $\mathsf{Issue}(pp, sk_0, vk_0, \mathsf{nym}_r)$: Compute $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{crs}, sk_0, (\mathsf{nym}_r, 1, \mathsf{cred}))$ and output $\mathcal{C} := (vk_0, 1, \mathsf{nym}_r, \sigma)$.
- $\mathsf{CredProve}(pp, sk, \mathsf{nym}, \mathsf{open}, \mathsf{nym}', \mathsf{open}', \mathcal{C})$:
  Parse $\mathcal{C} = (vk_0, \ell, \mathsf{nym}, \sigma)$ and abort if $\mathsf{Verify}(\mathsf{crs}, vk_0, \sigma, (\mathsf{nym}, \ell, \mathsf{cred})) = 0$.
  Otherwise, set $\langle T \rangle := (((\mathsf{nym}, sk, \mathsf{open}), (\mathsf{nym}', sk, \mathsf{open}')), \mathsf{nym}', \mathsf{proof})$.
  Compute $\sigma' \xleftarrow{\$} \mathsf{SigEval}(\mathsf{crs}, vk_0, T, (\mathsf{nym}, \ell, \mathsf{cred}), \sigma)$, and output $\pi := (\mathsf{nym}', \sigma')$.
- $\mathsf{CredVerify}(pp, vk_0, \mathsf{nym}, \ell, \pi)$: Parse $\pi = (\mathsf{nym}', \sigma')$; output 0 if $\mathsf{nym}' \neq \mathsf{nym}$.
  Otherwise, output $\mathsf{Verify}(\mathsf{crs}, vk_0, \sigma', (\mathsf{nym}, \ell, \mathsf{proof}))$.
- $\mathsf{Delegate}(pp, sk_{old}, \mathsf{nym}_{old}, \mathsf{open}_{old}, \mathsf{nym}_{new}, \mathcal{C})$: Parse $\mathcal{C} = (vk_0, \ell, \mathsf{nym}_{old}, \sigma)$ and abort if $\mathsf{Verify}(\mathsf{crs}, vk_0, \sigma, (\mathsf{nym}_{old}, \ell, \mathsf{cred})) = 0$.
  Otherwise, set $\langle T \rangle := ((\mathsf{nym}_{old}, sk_{old}, \mathsf{open}_{old}), \mathsf{nym}_{new}, \mathsf{cred})$.
  Compute $\sigma' \xleftarrow{\$} \mathsf{SigEval}(\mathsf{crs}, vk_0, T, (\mathsf{nym}_{old}, \ell, \mathsf{cred}), \sigma)$, and output $\mathcal{C}' := (vk_0, \ell + 1, \mathsf{nym}_{new}, \sigma')$.

Theorem V.1. *If the commitment scheme is computationally hiding and perfectly binding, and the signature is simulatable, malleable, simulation unforgeable, and simulation context hiding with respect to $\mathcal{T}_{dac}$, then the above construction describes a secure delegatable anonymous credentials scheme, as defined in Section II-C.*

In the next section, we see how to instantiate the malleable signature, using cm-NIZKs, to achieve the required malleability and security properties.

C. Instantiating our construction

Two constructions of cm-NIZKs exist within the literature, both due to Chase et al.: their original construction [12] based on Groth-Sahai proofs [39], and a more recent construction [24] based on succinct non-interactive arguments of knowledge (SNARGs) and homomorphic encryption. While the latter construction is less efficient, showing that it supports the class of transformations $\mathcal{T}_{\text{dac}}$ is relatively straightforward: all we need to show is that the language and class of transformations are what is called *t-tiered*, meaning that (1) every instance x in the language can be labeled with an integer $i = \mathsf{tier}(x)$, and (2) every transformation $T \in \mathcal{T}_{\text{dac}}$ is such that $\mathsf{tier}(T(x)) > \mathsf{tier}(x)$ for all $x \in L$ such that $\mathsf{tier}(x) < t$, and $T(x) = \bot$ if $\mathsf{tier}(x) = t$.

To see that our language is t-tiered, we recall that the relation R in Section IV was defined as $(x = (pk, m), w = sk) \in R \Leftrightarrow (pk, sk) \in \mathcal{R}_{pk}$ for a hard relation $\mathcal{R}_{pk}$. For the credentials, $m = (\mathsf{nym}_r, \ell, \mathsf{flag})$, so we define $\mathsf{tier}(x) := \ell$ if $\mathsf{flag} = \mathsf{cred}$ and $\mathsf{tier}(x) := \ell + 1$ if $\mathsf{flag} = \mathsf{proof}$.

To see that $\mathcal{T}_{\text{dac}}$ is also t-tiered, we observe that $\langle T \rangle = (\{\mathsf{nym}_i, sk_i, \mathsf{open}_i\}_{i=1}^{k}, \mathsf{nym}_{new}, \mathsf{flag}')$, where it is required that $k > 0$. Looking at the two cases for how transformations behave, for the first we see that $T(\mathsf{nym}_r, \ell, \mathsf{cred}) = (\mathsf{nym}_{new}, \ell + k, \mathsf{cred})$; then $\mathsf{tier}(x) = \ell$, $\mathsf{tier}(T(x)) = \ell + k$ for $k > 0$, and thus $\mathsf{tier}(T(x)) > \mathsf{tier}(x)$ as desired. In the second case, $T(\mathsf{nym}_r, \ell, \mathsf{cred}) = (\mathsf{nym}_{new}, \ell + k - 2, \mathsf{proof})$; here it is additionally required that $k \geq 2$, so $\mathsf{tier}(T(x)) = \ell + k - 2 + 1 > \ell = \mathsf{tier}(x)$ and the condition is still satisfied. To finally satisfy the requirement that $T(x) = \perp$ if $\mathsf{tier}(x) = t$, we could require that credentials can be delegated at most $t - 1$ times (the last tier t would then be reserved for proving possession).

While the result of Chase et al. [24] therefore assures us that we can construct a cm-NIZK supporting $\mathcal{T}_{\text{dac}}$. One might also wish to instantiate our cm-NIZK using their first, more efficient, construction. In order to do this, one would have to show that our relation and class of transformations are what is called *CM-friendly*; this essentially means that all of the objects (instances, witnesses, and transformations) can be represented as elements of a bilinear group and hence the system is compatible with Groth-Sahai proofs. In unpublished work [26] we show that this is indeed the case.

VI. Relations between Malleable Signature Definitions

In this section we relate our simulation-based notions of unforgeability and context hiding (Definitions III.4 and III.3, respectively) to definitions for homomorphic signatures as defined by Ahn et al. [3] and Attrapadung et al. [4], as well as to the recently introduced notions of policy-based signatures [30] and delegatable functional signatures [29].

A. Previous definitions

To begin our comparison, we must first recall the specific definitions of Ahn et al. and Attrapadung et al. to which we are comparing our own. We first recall the two main unforgeability definitions of Ahn et al.; we refer to the first as existential unforgeability, and the second as NHU unforgeability.

Definition VI.1 (Existential unforgeability)**. [3]** *For a signature* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigDerive})$ *malleable with respect to a predicate P, a table $Q = Q_m \times Q_\sigma$, and an adversary $\mathcal{A}$, consider the following game:*

- *Step 1.* $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$*;* $S, Q \leftarrow \emptyset$*.*
- *Step 2.* $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign},\mathsf{SigDerive},\mathsf{Reveal}}(vk)$*, where these oracles behave as described in Fig.1*

Then the signature scheme satisfies existential unforgeability *if for all such PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that the probability (over the choices of* KeyGen*,* Sign*,* $\mathsf{SigDerive}$*, $\mathcal{A}$, and the handles) that* $\mathsf{Verify}(vk, \sigma^*, m^*) = 1$*, and $m^* \notin P^*(Q_m)$ is at most $\nu(k)$.*[3]

[3]Here $P^*(M)$ denotes the set of messages derivable from M by repeated derivation, where a message m' is *derivable* from the set M if $P(M, m') = 1$.

The definition of NHU unforgeability is similar to that of existential unforgeability, with the exception that the Sign oracle is the only one provided.

Definition VI.2 (NHU unforgeability)**. [3]** *For a signature* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigDerive})$ *malleable with respect to a predicate P, a table $Q = Q_m \times Q_\sigma$, and an adversary $\mathcal{A}$, consider the following game:*

- *Step 1.* $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$*;* $Q \leftarrow \emptyset$*.*
- *Step 2.* $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\cdot)}(vk)$*, where* Sign *behaves as follows:*

$$\frac{\mathsf{Sign}_{sk}(m)}{\sigma \leftarrow \mathsf{Sign}(sk, m)}$$
$$\textit{add } (m, \sigma) \textit{ to } Q$$
$$\textit{return } \sigma$$

Then the signature scheme satisfies NHU unforgeability *if for all such PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that the probability (over the choices of* KeyGen*,* Sign*, and $\mathcal{A}$) that* $\mathsf{Verify}(vk, \sigma^*, m^*) = 1$ *and $m^* \notin P^*(Q_m)$ is at most $\nu(k)$.*

For context hiding, we compare simulation context hiding (as defined in Definition III.3) to the notion of *adaptive* context hiding given by Attrapadung et al., which is in turn inspired by the computational notion of context hiding given by Ahn et al.

Definition VI.3 (Adaptive context hiding)**. [4]** *For a signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigDerive})$ *malleable with respect to a predicate P, an adversary $\mathcal{A}$, and a bit b, let $p_b^{\mathcal{A}}(k)$ be the probability of the event that $b' = 0$ in the following game:*

- *Step 1.* $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$*.*
- *Step 2.* $(\mathsf{state}, \vec{m}, \vec{\sigma}, m') \xleftarrow{\$} \mathcal{A}(vk, sk)$*.*
- *Step 3. If* $\mathsf{Verify}(vk, \sigma_i, m_i) = 0$ *for some i or $P(\vec{m}, m') = 0$, abort and output $\perp$. Otherwise, form $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m')$ if $b = 0$, and $\sigma \xleftarrow{\$} \mathsf{SigDerive}(vk, \vec{m}, \vec{\sigma}, m')$ if $b = 1$.*
- *Step 4.* $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \sigma)$*.*

Then the signature scheme satisfies adaptive context hiding *if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.*

Before we move on to consider our new definitions, we relate the two notions of unforgeability using the notion of adaptive context hiding. As we show, when adaptive context hiding holds, NHU unforgeability implies existential unforgeability; as existential unforgeability trivially implies NHU unforgeability, this means the two notions are equivalent. For the rest of this section, we therefore focus mainly on the notion of NHU unforgeability.

Theorem VI.4. *If a P-homomorphic signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SigDerive})$ *is NHU unforgeable and adaptive context hiding, and if $m \in P^*(M)$ is efficiently decidable, then it is also existentially unforgeable.*

$\underline{\mathsf{Sign}_{sk}(m)}$
$h \xleftarrow{\$} \mathcal{H}$
add $(h, m, \mathsf{Sign}(sk, m))$ to S
return h

$\underline{\mathsf{SigDerive}_{pk}(\{h_i\}_i, m'))}$
$(h_i, m_i, \sigma_i) \leftarrow S \ \forall i$
if $P(\vec{m}, m') = 1$, pick $h' \xleftarrow{\$} \mathcal{H}$
add $(h', m', \mathsf{SigDerive}(vk, \vec{m}, \vec{\sigma}, m'))$ to S
return h'

$\underline{\mathsf{Reveal}(h)}$
$(h, m, \sigma) \leftarrow S$
add (m, σ) to Q
return σ

Fig. 1: Oracles for existential unforgeability

The proof of this theorem can be found in our report [26].

B. Relating our definitions to prior work

In Section III, we briefly outlined our reasons for choosing to work in the language of transformations rather than that of predicates. To meaningful relate our security definitions to the predicate-based definitions just presented, however, we first need a way to formally relate transformations and predicates. We do this as follows:

Definition VI.5. *We say that a transformation class $\mathcal{T}$* implements *a predicate P if for all $M \subset \mathcal{M}$ and $m^* \in \mathcal{M}$, $P(M, m^*) = 1$ if and only if there exist $m_1, \ldots, m_n \in \mathcal{M}$ and $T \in \mathcal{T}$ such that $T(m_1, \ldots, m_n) = m^*$.*

Note that we can always construct a transformation class $\mathcal{T}$ that implements a given predicate as follows: for each pair $(\vec{m}, m^*)$ such that $P(\vec{m}, m^*) = 1$, we consider $\mathcal{T}_P$ that contains an extreme partial function; i.e., $\mathcal{T}_P = \{\vec{m} \mapsto m^* \mid P(\vec{m}, m^*) = 1\}$. Sometimes, however, there is a more natural transformation class $\mathcal{T}$ for implementing a given predicate.[4]

The other major syntactic difference is that our simulation-based definitions consider signatures schemes in the CRS model, whereas previous definitions considered signatures in the standard model (i.e., without the trusted setup component Gen defined in Section III-A). This difference is also easily addressed: for any CRS model signature $\Sigma = (\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, a corresponding signature Σ' in the standard model can be defined as $(\mathsf{KeyGen}', \mathsf{Sign}, \mathsf{Verify})$, where KeyGen' runs $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}(1^k)$ and $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{crs})$, and outputs $(vk' := (\mathsf{crs}, vk), sk)$.

With these notational differences out of the way, we now proceed as follows: first, we show that for simulatable signatures, our notions of simulation unforgeability and simulation context hiding imply the notions of NHU unforgeability and adaptive context hiding respectively. Next, we show that existing constructions can be easily made simulatable, without changing their security guarantees, by adding a proof of knowledge and shifting to the CRS model. Finally, we show that for certain classes of transformations (that, to the best of our knowledge, includes all classes of transformations for which constructions exist), our notion of simulation unforgeability is implied by, and thus equivalent to, NHU unforgeability (again, with respect to simulatable signatures, as otherwise our definitions are not well defined).

1) Simulatability-based definitions imply previous definitions: The proof of the following two theorem can be found in in our report [26].

Theorem VI.6. *Let $\mathcal{T}$ be a transformation class that implements the predicate P. Let Σ be a signature in the CRS model that is malleable with respect to $\mathcal{T}$, and let Σ' be the corresponding standard model signature. If (1) Σ is simulatable, and (2) $m \in P^*(M)$ is efficiently decidable, then Σ' is NHU-unforgeable with respect to P if Σ is simulation unforgeable with respect to $\mathcal{T}$.*

Theorem VI.7. *Let $\mathcal{T}$ be a transformation class that implements a predicate P. Let Σ be a signature in the CRS model that is malleable with respect to $\mathcal{T}$, and let Σ' be the corresponding standard model signature. If Σ is simulatable and simulation context hiding with respect to $\mathcal{T}$, then Σ' is adaptive context hiding with respect to P.*

By combining these two theorems with Theorem VI.4, which relates NHU unforgeability back to existential unforgeability, we obtain the following corollary:

Corollary VI.8. *Let $\mathcal{T}$ be a transformation class that implements the predicate P. Let Σ be a signature in the CRS model that is malleable with respect to $\mathcal{T}$, and let Σ' be the corresponding standard model signature. Then if (1) Σ is simulatable, (2) Σ' is adaptive context hiding (alternatively, Σ is simulation context hiding), (3) P is efficiently decidable, and (4) Σ is simulation unforgeable, then Σ' is existentially unforgeable.*

2) Adding simulatability to existing constructions: After demonstrating that our definitions imply previous definitions for simulatable signatures, we now demonstrate that building simulatability into existing constructions is not too difficult. In fact, for any standard model signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ we consider a CRS model signature scheme $\Sigma^* = (\mathsf{Gen}^*, \mathsf{KeyGen}^*, \mathsf{Sign}, \mathsf{Verify})$, in which Gen^* outputs the crs for a non-interactive zero-knowledge proof of knowledge (NIZKPoK) for the relation $R := \{(vk, (sk, r)) \mid \mathsf{KeyGen}(1^k; r) = (vk, sk)\}$, and $\mathsf{KeyGen}^*(\mathsf{crs})$ runs $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k; r)$ and $\pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, vk, (sk, r))$, and outputs $(vk^* := (vk, \pi), sk)$. We denote this CRS model signature as the *extended* CRS model signature.

Theorem VI.9. *Let Σ be a signature scheme in the standard model, let Σ^* be the extended CRS model signature using*

[4] Ahn et al. suggest that in some cases one could pass an additional input w into the predicate. This could also be done here if w is efficiently computable from T.

$(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$*, and let* Σ' *be the standard model scheme corresponding to* Σ^**. Then, if* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ *is a zero-knowledge proof of knowledge,*

1) Σ^* *is simulatable;*
2) *if* Σ *is NHU unforgeable, then* Σ' *is NHU unforgeable;*
3) *if* Σ *is adaptive context hiding, then* Σ' *is adaptive context hiding.*

The proof of this theorem can be found in in our report [26].

We remark that Section 6.1 of [42] describes another mechanism for making signature schemes simulatable using OR proofs. We did, however, not investigate this mechanism with respect to malleability.

3) When existing constructions meet our definitions: For context hiding, our notion of simulation context hiding seems clearly stronger than existing definitions, as it guarantees privacy even in case of adversarially-generated verification keys. What is unclear, however, is whether existing malleable signature schemes (when adapted to be simulatable) can meet our stronger notion of simulation unforgeability. Perhaps unsurprisingly, this depends largely on the class of transformations they require.

As it turns out, for most existing schemes the relevant class of transformations is such that it is easy to determine, given a set of "base" messages and a transformed message, the transformation that was applied. More formally, we have the following definition:

Definition VI.10 (Trivial extractability)**.** *We say a transformation class* $\mathcal{T}$ *is* trivially extractable *if there exists a PPT algorithm* $\mathsf{TrivExt}$ *such that for all polynomial-sized* $M \subset \mathcal{M}$ *and* $m^* \in \{m \mid \exists T \in \mathcal{T}, \vec{m} \in M : m = T(\vec{m})\}$*, the algorithm* $\mathsf{TrivExt}(M, m^*)$ *produces* $\vec{m} = (m_1, \ldots, m_n) \in M$ *and transformation* T *such that* $m^* = T(\vec{m})$*.*

As an example, we consider the linear homomorphic signature construction given by Attrapadung et al. Phrased in the language of transformations, their construction allows for linear combinations of (similarly tagged) messages; i.e., for any vector $\vec{\beta} = (\beta_1, \ldots, \beta_n) \in \mathbb{Z}^n$, $T_{\vec{\beta}}((\tau_1, \vec{y}_1), \ldots, (\tau_n, \vec{y}_n)) = (\tau_1, \sum_i \beta_i \vec{y}_i)$ if $\tau_1 = \ldots = \tau_n$ and $\perp$ otherwise. To see how this transformation class is trivially extractable, we construct the $\mathsf{TrivExt}$ algorithm as follows: on input M and $m^* = (\tau, \vec{y})$, it first lets M_τ define the set of messages in M with tag τ. Second, it identifies a subset of M_τ that forms a basis for the subspace spanned by the vectors in M_τ. Third, it finds coefficients to represent $\vec{y}$ as a linear combination of those basis vectors. Fourth and last, it outputs the basis vectors as $(m_1, \ldots, m_n)$, and the linear combination defined by the coefficients as the transformation T. This algorithm is furthermore efficient, as each step involves basis linear algebra.

In a similar manner, one can show that subset signatures, quotable signatures, and transitive signatures are all trivially extractable as well; briefly, for each of these constructions $\mathsf{TrivExt}$ respectively corresponds to finding a superset, searching for a substring, and finding a path in a graph, all of which have simple efficient algorithms.

We argue that for such transformation classes, NHU unforgeability and simulation unforgeability are equivalent.

Theorem VI.11. *Let* $\mathcal{T}$ *be a transformation class that implements the predicate* P*. Let* Σ *be a signature in the CRS model that is malleable with respect to* $\mathcal{T}$*, and let* Σ' *be the corresponding standard model signature. If* Σ *is simulatable and* $\mathcal{T}$ *is trivially extractable, then* Σ' *is NHU unforgeable with respect to* P *if and only if* Σ *is simulation unforgeable with respect to* $\mathcal{T}$*.*

The proof of this theorem can be found in in our report [26].

Once again, we can combine Theorems VI.9 and VI.11 to get a corollary that shows that, for any of the standard types of transformation considered in previous work (i.e., any trivially extractable transformations), we can construct simulation-unforgeable signature schemes based on ones that are NHU unforgeable.

Corollary VI.12. *If* Σ *is NHU unforgeable for a predicate* P *implemented by a trivially extractable transformation class* T*, then the extended CRS model signature* Σ^* *is simulation unforgeable.*

C. Other schemes implied by our definitions

In this section, we briefly sketch how malleable signatures can be used to imply both policy-based signatures (as introduced by Bellare and Fuchsbauer [30]) and delegatable functional signatures (as introduced by Backes et al. [29]).

1) Policy-based signatures: In a policy-based signature (PBS) scheme a signer is restricted to sign messages m conforming to some authority-specified policy p. The main requirements are that signatures are unforgeability and hide the policy used for their generation. PBS offers value both on the practical side, as they allow a corporation to control what messages its employees can sign under the corporate key, as well as on the theoretical side as they capture others forms of signatures as special cases.

Let $\mathcal{T}$ be a transformation class that implements the policy checker predicate PC of a PBS scheme. A transformation $T_{(p,m,w)} \in \mathcal{T}$, with $T_{p,m,w}(p) = m$ is represented by values $\langle T \rangle = (p, m, w)$ for which $\mathsf{PC}((p, m), w) = 1$. The value w is an additional witness used in the definition of the predicate.

The key insights for realizing policy-based signatures using malleable signatures are that the signing key for a policy corresponds to a signature on the policy under msk, and that generating a signature under that policy corresponds to deriving a signature on m from a signature on p.

The construction is quite straightforward, with the only interesting aspect being that the policy p is added into the policy key skp to make the full transformation description available in $\mathsf{PBS.Sign}$.

- $\mathsf{PBS.Setup}(1^k)$. Run $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}(1^k)$ and $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{crs})$. Return $(pp = (\mathsf{crs}, vk), msk = sk)$.
- $\mathsf{PBS.KeyGen}(pp, msk, p))$. Return $skp = (\mathsf{Sign}(\mathsf{crs}, sk, p), p)$.

- $\mathsf{PBS.Sign}(pp, skp, m, w)$.
  Return $\sigma \xleftarrow{\$} \mathsf{SigEval}(\mathsf{crs}, vk, T_{(p,m,w)}, m, skp)$.
- $\mathsf{PBS.Verify}(pp, m, \sigma)$. Return $\mathsf{Verify}(\mathsf{crs}, vk, m, \sigma)$.
- $\mathsf{PBS.SimSetup}(1^k)$. Run $\mathsf{crs}, \tau_s, \tau_e \xleftarrow{\$} \mathsf{SimExtGen}(1^k)$ and $vk, sk \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{crs})$.
  Return $(pp = (\mathsf{crs}, vk), msk = sk, tr = (\mathsf{crs}, vk, \tau_s, \tau_e))$.
- $\mathsf{PBS.SimKeyGen}(tr, p)$.
  Return $skp = (\mathsf{SimSign}(\mathsf{crs}, \tau_s, vk, p), p)$.
- $\mathsf{PBS.SimSign}(tr, m)$.
  Return $\sigma \xleftarrow{\$} \mathsf{SimSign}(\mathsf{crs}, \tau_s, vk, m)$.
- $\mathsf{PBS.Extr}(tr, m, \sigma)$.
  Return $(p, w) \leftarrow \mathsf{SigExt}(\mathsf{crs}, vk, \tau_e, m, \sigma, \emptyset)$.

To prove that malleable signatures for $\mathcal{T}$ imply policy-based signatures, we need to show that our construction is both *simulatable* and *extractable*.

Proof: (Sketch.) To prove *simulatability* we start from the $\mathbf{Exp}^{SIM}_{\mathsf{PBS}}$ experiment and go through a series of game transformations. We first change the generation of pp_1 to use $\mathsf{PBS.SimSetup}$ to obtain an additional trapdoor tr_1. We also change the generation of psk_1 to use $\mathsf{PBS.SimKeyGen}(tr_1, p)$. The difference in the success probability is bounded by the success probability of a reduction breaking *simulatability* of the underlying malleable signature scheme. Next, we change the generation of σ_1 to use $\mathsf{PBS.SimSign}$ instead of $\mathsf{PBS.Sign}$. The difference in the success probability is bounded by the success probability of a reduction breaking *simulation context hiding*. Now the distributions for $b = 0$ and $b = 1$ are the same.

The *extractability* of the policy-based signature scheme follows by a trivial reduction to *simulation unforgeability*. We simulate INITIALIZE, SKEYGEN, and SIMSIGN using the simulation unforgeability challenge and oracle queries. We forward the input to FINALIZE as the message and signature for which we want to break simulation unforgeability. Whenever FINALIZE returns true, our reduction wins the simulation unforgeability game as $p \notin Q_K$ corresponds to $\{p\} \not\subseteq Q_m$ and $\mathsf{PC}((p, m), w) = 0$ corresponds to $m \neq T_{(p,m,w)}(p)$ or $T_{(p,m,w)} \notin \mathcal{T}$. ∎

2) Delegatable functional signatures: A Delegatable functional signature (DFS) scheme supports the delegation of signing capabilities to a party called the evaluator with respect to a functionality $\mathcal{F}$. In a DFS, the signer chooses an evaluator, specifies how the evaluator can modify the signature, allows additional input and decides how the evaluator can further delegate its capabilities.

To construct delegatable functional signatures given a hard relation and a malleable signature scheme for an appropriate transformation class, we first use the hard relation to define public and secret keys. To issue a delegatable signature with evaluator pk_{ev}, function f, and message m, a signer computes a malleable signature on the message $m_{malleable} = (pk_{ev}, f, m)$. This signature scheme should be simulatable, malleable simulation context hiding, and simulation unforgeable with respect to the following transformation class $\mathcal{T}$: for every valid key pair (pk_{ev}, sk_{ev}) and for every pk' and α, $\mathcal{T}$ includes a transformation with description $\langle T \rangle = (sk_e v, pk', \alpha)$ such that for every f and m, $T(pk_e v, f, m) = (pk', f', m')$ where $(f', m') \leftarrow \mathcal{F}(\lambda, f, \alpha, pk', m)$, and $T(x, f, m) = \perp$ for all $x \neq pk_{ev}$. $\mathsf{Eval}_{\mathcal{F}}$ then simply finds the appropriate transformation and applies it to the signature using $\mathsf{SigEval}$.

Unforgeability follows from the fact that we can extract a valid transformation, which includes the secret key sk_{ev}; hardness of the relation then guarantees that an adversary cannot modify a signature unless he has the appropriate secret key. Privacy follows in a straightforward way from simulatability and simulation context hiding.

VII. CONCLUSIONS AND FUTURE WORK

We give new definitions for malleable signatures that are suitable for a larger class of applications. We demonstrate this by instantiating delegatable anonymous credentials and by investigating the relations between malleable signatures and other primitives like policy-based signatures and delegatable functional signatures.

We instantiate malleable signatures generically based on malleable zero-knowledge proofs which can be instantiated either very succinctly using SNARGS [24] or using Groth-Sahai proofs [12]. An interesting open question is the relationship between this new instantiation of the above primitives and existing constructions. When instantiated using Groth-Sahai proofs, they are superficially similar in that they often use structure-preserving signatures and OR proofs to achieve simulation-soundness, but there are some subtle differences in what is signed and proved. A detailed comparative performance and security analysis of the two approaches would thus be very valuable.

ACKNOWLEDGMENTS

Anna Lysyanskaya was supported by NSF grants 1012060, 0964379, 0831293, and by a Sloan Foundation fellowship, and Sarah Meiklejohn was supported in part by a MURI grant administered by the Air Force Office of Scientific Research and in part by a graduate fellowship from the Charles Lee Powell Foundation.

REFERENCES

[1] S. Haber, Y. Hatano, Y. Honda, W. G. Horne, K. Miyazaki, T. Sander, S. Tezoku, and D. Yao, "Efficient signature schemes supporting redaction, pseudonymization, and data deidentification," in *ASIACCS*, 2008, pp. 353–362.

[2] C. Brzuska, H. Busch, Ö. Dagdelen, M. Fischlin, M. Franz, S. Katzenbeisser, M. Manulis, C. Onete, A. Peter, B. Poettering, and D. Schröder, "Redactable signatures for tree-structured data: Definitions and constructions," in *ACNS*, 2010, pp. 87–104.

[3] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, abhi shelat, and B. Waters, "Computing on authenticated data," in *Proceedings of TCC 2012*, ser. LNCS, vol. 7194. Springer-Verlag, 2012, pp. 1–20.

[4] N. Attrapadung, B. Libert, and T. Peters, "Computing on Authenticated Data: New Privacy Definitions and Constructions," in *Asiacrypt 2012*, ser. Lecture Notes on Computer Science, vol. 7658. Springer, 12 2012.

[5] D. Chaum, "Security without identification: transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[6] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Proceedings of Eurocrypt 2001*, ser. LNCS, vol. 2045. Springer-Verlag, 2001, pp. 93–118.

[7] ——, "Signature schemes and anonymous credentials from bilinear maps," in *Proceedings of Crypto 2004*, ser. LNCS, vol. 3152. Springer-Verlag, 2004, pp. 56–72.

[8] M. Chase and A. Lysyanskaya, "On signatures of knowledge," in *CRYPTO*, ser. Lecture Notes in Computer Science, C. Dwork, Ed., vol. 4117. Springer, 2006, pp. 78–96.

[9] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Delegatable anonymous credentials," in *Proceedings of Crypto 2009*, ser. LNCS, vol. 5677. Springer-Verlag, 2009, pp. 108–125.

[10] L. Sweeney, "*k*-anonymity: a model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[11] P. Golle, "Revisiting the uniqueness of simple demographics in the us population," in *Proceedings of the 5th ACM workshop on Privacy in electronic society*, ser. WPES '06. New York, NY, USA: ACM, 2006, pp. 77–80. [Online]. Available: http://doi.acm.org/10.1145/1179601.1179615

[12] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn, "Malleable proof systems and applications," in *Proceedings of Eurocrypt 2012*, 2012, pp. 281–300.

[13] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *Proceedings of Crypto 1986*, ser. LNCS, vol. 263. Springer-Verlag, 1986, pp. 186–194.

[14] M. Bellare and S. Goldwasser, "New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs," in *Proceedings of Crypto 1989*, 1989, pp. 194–211.

[15] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Proceedings of CT-RSA 2002*, 2002, pp. 244–262.

[16] K. Miyazaki, G. Hanaoka, and H. Imai, "Digitally signed document sanitizing scheme based on bilinear maps," in *Proceedings of ASIACCS 2006*, 2006, pp. 343–354.

[17] E.-C. Chang, C. L. Lim, and J. Xu, "Short redactable signatures using random trees," in *Proceedings of CT-RSA 2009*, 2009, pp. 133–147.

[18] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," in *ESORICS*, 2005, pp. 159–177.

[19] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk, "Security of sanitizable signatures revisited," in *Public Key Cryptography*, 2009, pp. 317–336.

[20] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder, "Unlinkability of sanitizable signatures," in *Public Key Cryptography*, 2010, pp. 444–461.

[21] R. Steinfeld, L. Bull, and Y. Zheng, "Context extraction signatures," in *Proceedings of ICISC 2001*, 2001, pp. 285–304.

[22] S. Micali and R. Rivest, "Transitive signature schemes," in *Proceedings of CT-RSA 2002*, 2002, pp. 236–243.

[23] M. Bellare and G. Neven, "Transitive signatures: new schemes and proofs," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 2133–2151, 2005.

[24] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn, "Succinct malleable NIZKs and an application to compact shuffles," in *Proceedings of TCC 2013*, 2013, pp. 100–119.

[25] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proceedings of Crypto 2004*, ser. LNCS, vol. 3152. Springer-Verlag, 2004, pp. 41–55.

[26] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn, "Malleable signatures: Complex unary transformations and delegatable anonymous credentials," Cryptology ePrint Archive, Report 2013/179, 2013, http://eprint.iacr.org/.

[27] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: the case of hashing and signing," in *Proceedings of Crypto 1994*, 1994, pp. 216–233.

[28] E. Boyle, S. Goldwasser, and I. Ivan, "Functional signatures and pseudorandom functions," Cryptology ePrint Archive, Report 2013/401, 2013, http://eprint.iacr.org/.

[29] M. Backes, S. Meiser, and D. Schröder, "Delegatable functional signatures," Cryptology ePrint Archive, Report 2013/408, 2013, http://eprint.iacr.org/.

[30] M. Bellare and G. Fuschbauer, "Policy-based signatures," Cryptology ePrint Archive, Report 2013/413, 2013, http://eprint.iacr.org/2013/413.

[31] D. X. Charles, K. Jain, and K. Lauter, "Signatures for network coding," *IJICoT*, vol. 1, no. 1, pp. 3–14, 2009.

[32] D. Boneh, D. M. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, S. Jarecki and G. Tsudik, Eds., vol. 5443. Springer, 2009, pp. 68–87.

[33] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," in *Public Key Cryptography*, 2010, pp. 142–160.

[34] N. Attrapadung and B. Libert, "Homomorphic network coding signatures in the standard model," in *Public Key Cryptography*, 2011, pp. 17–34.

[35] D. Boneh and D. M. Freeman, "Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures," in *Public Key Cryptography*, 2011, pp. 1–16.

[36] ——, "Homomorphic signatures for polynomial functions," in *Proceedings of Eurocrypt 2011*, ser. LNCS, vol. 6632. Springer-Verlag, 2011, pp. 149–168.

[37] D. M. Freeman, "Improved security for linearly homomorphic signatures: A generic framework," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, M. Fischlin, J. Buchmann, and M. Manulis, Eds., vol. 7293. Springer, 2012, pp. 697–714.

[38] N. Attrapadung, B. Libert, and T. Peters, "Efficient completely context-hiding quotable and linearly homomorphic signatures," in *Proceedings of PKC 2013*, 2013, pp. 386–404.

[39] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Proceedings of Eurocrypt 2008*, ser. LNCS, vol. 4965. Springer-Verlag, 2008, pp. 415–432.

[40] G. Fuchsbauer, "Commuting signatures and verifiable encryption and an application to non-interactively delegatable credentials," *IACR Cryptology ePrint Archive*, vol. 2010, p. 233, 2010.

[41] M. Abe and M. Ohkubo, "A framework for universally composable non-committing blind signatures," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Springer, 2009, pp. 435–450.

[42] M. Abe, K. Haralambiev, and M. Ohkubo, "Signing on elements in bilinear groups for modular protocol design," Cryptology ePrint Archive, Report 2010/133, 2010, http://eprint.iacr.org/2010/133.

[43] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo, "Structure-preserving signatures and commitments to group elements," in *Proceedings of Crypto 2010*, ser. LNCS, vol. 6223, 2010, pp. 209–236.

[44] A. de Santis, G. di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai, "Robust non-interactive zero knowledge," in *Proceedings of Crypto 2001*, ser. LNCS, vol. 2139. Springer-Verlag, 2001, pp. 566–598.

[45] J. Groth, "Simulation-sound NIZK proofs for a practical language and constant size group signatures," in *Proceedings of Asiacrypt 2006*, ser. LNCS, vol. 4284. Springer-Verlag, 2006, pp. 444–459.

[46] D. Chaum and T. Pedersen, "Transferred cash grows in size," in *Advances in Cryptology - EUROCRYPT' 92*, ser. Lecture Notes in Computer Science, R. Rueppel, Ed. Springer Berlin Heidelberg, 1993, vol. 658, pp. 390–407. [Online]. Available: http://dx.doi.org/10.1007/3-540-47555-9_32

[47] M. Blum, A. de Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge," *SIAM Journal of Computing*, vol. 20, no. 6, pp. 1084–1118, 1991.

[48] J. Groth, R. Ostrovsky, and A. Sahai, "Perfect non-interactive zero-knowledge for NP," in *Proceedings of Eurocrypt 2006*, ser. LNCS, vol. 4004. Springer-Verlag, 2006, pp. 339–358.

[49] U. Feige, D. Lapidot, and A. Shamir, "Multiple non-interactive zero knowledge proofs under general assumptions," *SIAM Journal of Computing*, vol. 29, no. 1, pp. 1–28, 1999.