Revocation for Delegatable Anonymous Credentials

Tolga Acar and Lan Nguyen

Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
tolga@microsoft.com, languyen@microsoft.com
http://research.microsoft.com/en-us/people/{tolga,languyen}

Abstract. This paper introduces and formalizes *homomorphic proofs* that allow 'adding' proofs and proof statements to get a new proof of the 'sum' statement. Additionally, we introduce a construction of homomorphic proofs, and show an *accumulator scheme with delegatable non-membership proofs* (ADNMP) as one of its applications with provable security. Finally, the proposed accumulator method extends the BC-CKLS scheme [1] to create a new provably secure *revocable delegatable anonymous credential* (RDAC) system. Intuitively, the new accumulator's delegatable non-membership (NM) proofs enable user A, without revealing her identity, to delegate to user B the ability to prove that A's identity is not included in a blacklist that can later be updated. The delegation is redelegatable, unlinkable, and verifiable.

1 Introduction

Proof systems play important roles in many cryptographic systems, such as signature, authentication, encryption, anonymous credential and mix-net. In a proof system between a prover and a verifier, an honest prover with a *witness* can convince a verifier about the truth of a *statement* but an adversary cannot convince a verifier of a false statement. Groth and Sahai [2] proposed a novel class of non-interactive proof systems (GS) with a number of desirable properties which are not available in previous ones. They are efficient and general. They do not require the random oracle assumption [3]. They can be randomized, i.e. one can generate a new proof from an existing proof of the same statement without knowing the witness. In this paper, we unveil another valuable feature of GS proofs: homomorphism.

Proof systems are used to construct *accumulators* [4,5,6,7,8]. An accumulator allows aggregation of a large set of elements into one constant-size *accumulator value*. The 'membership' proof system proves that an element is accumulated. An accumulator is *universal* if it has 'non-membership' proof system to prove that a given element is not accumulated in the accumulator value [9,10]. An accumulator is *dynamic* if the costs of adding and deleting elements and updating the accumulator value and proof systems' witnesses do not depend on the number of elements aggregated. Applications of accumulators include space-efficient time stamping, ad-hoc anonymous authentication, ring signatures, ID-Based systems, and membership *revocation* for identity escrow, group signatures and *anonymous credentials* [6].

In *anonymous credential* systems, a user can prove some credentials without revealing any other private information such as her identity. There have been several proposals [11,12,1]; applications such as in direct anonymous attestation (DAA) [13] and anonymous electronic identity (eID) token [14,15]; and implementations such as U-prove [15], Idemix [14] and java cards [16]. An anonymous credential system is *delegatable* [1] if its credential can be delegated from one user to another user so that a user can anonymously prove a credential which is delegated some levels away from the original issuer. Delegation is important for efficient credential management in organizations, as a person typically delegates certain authorities to colleagues to execute tasks on her behalf. *Revocation* is indispensable in credential systems in practice, as dispute, compromise, abuse, mistake, identity change, hacking and insecurity can make any credential become invalid before its expiration. The anonymity and delegation properties make revocation more challenging: the user must prove anonymously that her whole credential chain is not revoked. The primary revocation methods are based on accumulators [17,10], offering a constant cost for an unrevoked proof. However, the current schemes do not work for delegated anonymous credentials.

Contributions. We present three contributions in this paper, incrementally building on each other: *(i)* formal definition of homomorphic proofs and a construction based on GS proofs, *(ii)* dynamic universal accumulators with delegatable non-membership proof (ADNMP), and *(iii)* a revocable delegatable anonymous credential system (RDAC).

We first introduce and formally define the new notion of *homomorphic proofs*, which means there is an operation that 'adds' proofs, their statements and witnesses to produce a new proof of the 'sum' statement and the 'sum' witness. We present and prove a construction for homomorphic proofs from GS proofs [2]. The general nature of GS proofs partly explains the reason behind its numerous applications: group signatures, ring signatures, mix-nets, anonymous credentials, and oblivious transfers. Our homomorphic construction uses the most general form of GS proofs to maximize the range of possible applications.

Homomorphic proofs can be applied to homomorphic signatures [18], homomorphic authentication [19], that found applications in provable cloud storage [19], network coding [20,21], digital photography [22] and undeniable signatures [23]. Another possible application area is homomorphic encryption and commitment schemes that are used in mix-nets [24], voting [25], anonymous credentials [1] and other multi-party computation systems. Gentry's recent results on fully homomorphic encryption [26] allow computing any generic function of encrypted data without decryption and can be applied to cloud computing and searchable encryption.

Section 3.3 compares this work to the DHLW homomorphic NIZK (Non Interactive Zero Knowledge) recently proposed in [27]. While the DHLW scheme takes the traditional homomorphism approach, we employ Abelian groups and introduce a more general definition where proof systems satisfying the DHLW definition are a subset of the new proof systems. We note that DHLW's homomorphic NIZK definition and construction do not cover the new homomorphic proofs to build ADNMP and RDAC. From an application point of view, DHLW homomorphic NIZK targets leakage-resilient cryptography, and the new homomorphic proofs target accumulators and revocation.

Secondly, we introduce and build an *accumulator with delegatable non-membership proof* (ADNMP) scheme based on homomorphic proofs. We define security requirements for ADNMP, and give security proofs for the ADNMP scheme. The constructions in the SXDH (Symmetric External Diffie Hellman) or SDLIN (Symmetric Decisional Linear) instantiations of GS proofs allow the use of the most efficient curves for pairings in the new accumulator scheme [28].

To our knowledge, this is the first accumulator with a *delegatable* non-membership proof. Previously, there were only two accumulators with non-membership proofs, i.e. universal accumulators LLX [9] and ATSM [10]; both are not delegatable. Delegability allows us to construct delegatable revocation for delegatable anonymous credentials. Our accumulator uses GS proofs without random oracles where LLX and ATSM rely on the random oracle assumption for non-interactive proofs. LLX is based on the Strong RSA assumption and defined in composite-order groups, and ATSM is based on the Strong DH assumption and defined in prime-order bilinear pairing groups. Our scheme is also built in prime-order bilinear pairing groups that require storage much smaller than RSA composite-order groups. The new non-membership prover requires no pairing compared to ATSM's four pairings.

The main challenge in blacklisting delegatable anonymous credentials that can further be delegated is to create accumulators satisfying the following requirements. First, user A, without revealing private information, can delegate the ability to prove that her identity is not accumulated in any blacklist to user B so that such proofs generated by A and B are indistinguishable and the blacklist may change anytime. Second, the delegation must be unlinkable, i.e. it must be hard to tell if two such delegations come from the same delegator A. Third, user B is able to redelegate the ability to prove that A's credential is not blacklisted to user C, such that the information C obtains from the redelegation is indistinguishable from the information one obtains from A's delegation. Finally, any delegation information must be verifiable for correctness. The new ADNMP scheme satisfies these requirements.

By employing the ADNMP approach, our final contribution is to create the first *delegatable anonymous credential system with delegatable revocation* capability; an RDAC system. Traditionally, blacklisting of anonymous credentials relies on accumulators [8]. The identities of revoked credentials are accumulated in a blacklist, and verification of the accumulator's NM proof determines the credential's revocation status. A natural rule in a revoked delegatable credential, that our scheme also follows, is to consider all delegated descendants of the credential revoked. Applying that rule to delegatable anonymous credentials, a user must anonymously prove that all ancestor credentials are not revoked, even when the blacklist changes.

Homomorphic proofs bring delegability of proofs to another level. A proof's *statement* often consists of *commitments* of variables (witnesses) and *conditions*. Randomizable and malleable proofs introduced in [1] allows generation of a new proof and randomization of the statement's commitments without knowing the witness, but the statement's conditions always stay the same. Homomorphic proofs allow generating a new proof for a new statement containing new conditions, without any witness. A user can delegate her proving capability to another user by revealing some homomorphic proofs. A linear combination of these proofs and their statements allows the delegatee to generate new proofs for other statements with different conditions (e.g., an updated blacklist in ADNMP). In short, the BCCKLS paper [1] deals with delegating proofs of the same statements' conditions, whereas this paper deals with delegating proofs of changing statements' conditions.

2 Background

Tech Report [29] provides more details of existing cryptographic primitives: Bilinear Map Modules, $\mathcal{R}$-module, Bilinear pairings, SXDH, Composable zero-knowledge (ZK), Randomizing proofs and commitments, Partial extractability, Accumulator, and Delegatable anonymous credentials.

Notation. PPT stands for Probabilistic Polynomial Time; CRS for Common Reference String; Pr for Probability; NM for non-membership; ADNMP for Accumulator with Delegatable NM Proofs; RDAC for Revocable Delegatable Anonymous Credential; and $\leftarrow$ for random output. For a group $\mathbb{G}$ with identity $\mathcal{O}$, let $\mathbb{G}^* := \mathbb{G}\backslash\{\mathcal{O}\}$. $Mat_{m\times n}(\mathcal{R})$ is the set of matrices with size $m \times n$ in $\mathcal{R}$. For a matrix Γ, $\Gamma[i,j]$ is the value in i^{th} row and j^{th} column. A vector $\boldsymbol{z}$ of l elements can be viewed as a matrix of l rows and 1 column. For a vector $\boldsymbol{z}$, $z[i]$ is the i^{th} element. For a function $\nu : \mathbb{Z} \to \mathbb{R}$, ν is *negligible* if $|\nu(k)| < k^{-\alpha}$, $\forall\alpha > 0$, $\forall k > k_0$, $\exists k_0 \in \mathbb{Z}^+$, $k \in \mathbb{Z}$.

Proof System. Let $\mathbf{R}$ be an efficiently computable relation of $(\mathit{Para}, \mathit{Sta}, \mathit{Wit})$ with setup parameters Para, a statement Sta, and a witness Wit. A non-interactive proof system for $\mathbf{R}$ consists of 3 PPT algorithms: a Setup, a prover Prove, and a verifier Verify. A non-interactive proof system (Setup, Prove, Verify) must be complete and sound. $\mathsf{Completeness}$ means that for every PPT adversary $\mathcal{A}$, $|\Pr[\mathit{Para} \leftarrow \mathsf{Setup}(1^k); (\mathit{Sta}, \mathit{Wit}) \leftarrow \mathcal{A}(\mathit{Para}); \mathit{Proof} \leftarrow \mathsf{Prove}(\mathit{Para}, \mathit{Sta}, \mathit{Wit}) : \mathsf{Verify}(\mathit{Para}, \mathit{Sta}, \mathit{Proof}) = 1 \text{ if } (\mathit{Para}, \mathit{Sta}, \mathit{Wit}) \in \mathbf{R}] - 1|$ is negligible. $\mathsf{Soundness}$ means that for every PPT adversary $\mathcal{A}$, $|\Pr[\mathit{Para} \leftarrow \mathsf{Setup}(1^k); (\mathit{Sta}, \mathit{Proof}) \leftarrow \mathcal{A}(\mathit{Para}) : \mathsf{Verify}(\mathit{Para}, \mathit{Sta}, \mathit{Proof}) = 0 \text{ if } (\mathit{Para}, \mathit{Sta}, \mathit{Wit}) \notin \mathbf{R}, \forall \mathit{Wit}] - 1|$ is negligible.

GS Proofs. Tech Report [29] provides a comprehensive summary of GS proofs and its instantiation in SXDH. Briefly, the GS *setup* algorithm generates Gk and CRS σ. Gk contains L tuples, each of which has the form (A_1, A_2, A_T, f) where A_1, A_2, A_T are $\mathcal{R}$-modules with map $f : A_1 \times A_2 \to A_T$. L is also the number of equations in a statement to be proved. CRS σ contains L corresponding tuples of $\mathcal{R}$-modules and maps $(B_1, B_2, B_T, \iota_1, \iota_2, \iota_T)$, where ι_j :

$A_j \rightarrow B_j$. A GS *statement* is a set of L corresponding tuples ($\boldsymbol{a} \in A_1^n, \boldsymbol{b} \in A_2^m, \Gamma \in Mat_{m\times n}(\mathcal{R}), t \in A_T$) satisfying $\boldsymbol{a} \cdot \boldsymbol{y} + \boldsymbol{x} \cdot \boldsymbol{b} + \boldsymbol{x} \cdot \Gamma \boldsymbol{y} = t$; where ($\boldsymbol{x} \in A_1^m, \boldsymbol{y} \in A_2^n$) is the corresponding witness (there are L witness tuples), and denote $\boldsymbol{a} \cdot \boldsymbol{y} = \sum_{j=1}^{n} f(a[j], y[j])$. The *proof* of the statement includes L corresponding tuples, each of which consists of commitments $\boldsymbol{c} \in B_1^m$ of $\boldsymbol{x}$ and $\boldsymbol{d} \in B_2^n$ of $\boldsymbol{y}$ with values $\boldsymbol{\pi}$ and $\boldsymbol{\psi}$. In the SXDH instantiation of GS proofs, *Para* includes bilinear pairing setup $Gk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ and CRS $\sigma = (B_1, B_2, B_T, F, \iota_1, p_1, \iota_2, p_2, \iota_1', p_1', \iota_2', p_2', \iota_T, p_T, \boldsymbol{u}, \boldsymbol{v})$ where $B_1 = \mathbb{G}_1^2$, $B_2 = \mathbb{G}_2^2$ and $B_T := \mathbb{G}_T^4$. The maps are $\iota_j : A_j \rightarrow B_j$, $p_j : B_j \rightarrow A_j$, $\iota_j' : \mathbb{Z}_p \rightarrow B_j$ and $p_j' : B_j \rightarrow \mathbb{Z}_p$. Vectors $\boldsymbol{u}$ of $u_1, u_2 \in B_1$ and $\boldsymbol{v}$ of $v_1, v_2 \in B_2$ are commitment keys for $\mathbb{G}_1$ and $\mathbb{G}_2$.

3 Homomorphic Proofs

3.1 Formalization

Recall that an Abelian group must satisfy 5 requirements: Closure, Associativity, Commutativity, Identity Element and Inverse Element.

Definition 1. *Let* (Setup, Prove, Verify) *be a proof system for a relation* $\mathbf{R}$ *and* $Para \leftarrow \mathsf{Setup}(1^k)$. *Consider a subset* Π *of all* $(Sta, Wit, Proof)$ *such that* $(Para, Sta, Wit) \in \mathbf{R}$ *and* $\mathsf{Verify}(Para, Sta, Proof) = 1$, *and an operation* $+_\Pi : \Pi \times \Pi \rightarrow \Pi$. Π *is a set of* ***homomorphic*** *proofs if* $(\Pi, +_\Pi)$ *satisfies the 3 requirements: Closure, Associativity and Commutativity.*

Consider an $I_\Pi := (Sta_0, Wit_0, Proof_0) \in \Pi$. Π *is a set of* ***strongly homomorphic*** *proofs if* $(\Pi, +_\Pi, I_\Pi)$ *forms an Abelian group where* I_Π *is the identity element.*

Note that if Π is strongly homomorphic, then Π is also homomorphic. If $+_\Pi$ $((Sta_1, Wit_1, Proof_1), (Sta_2, Wit_2, Proof_2)) \mapsto (Sta, Wit, Proof)$, we have the following notations:

$(Sta, Wit, Proof) \leftarrow (Sta_1, Wit_1, Proof_1) +_\Pi (Sta_2, Wit_2, Proof_2)$, $Sta \leftarrow Sta_1 +_\Pi Sta_2$, $Wit \leftarrow Wit_1 +_\Pi Wit_2$, and $Proof \leftarrow Proof_1 +_\Pi Proof_2$.

We also use the multiplicative notation $n(Sta, Wit, Proof)$ for the self addition for n times of $(Sta, Wit, Proof)$. Similarly, we also use $\sum_i n_i$ $(Sta_i, Wit_i, Proof_i)$ to represent linear combination of statements, witnesses and proofs. These homomorphic properties are particularly useful for randomizable proofs: one can randomize a proof computed from the homomorphic operation to get another proof, which is indistinguishable from a proof generated by Prove.

3.2 GS Homomorphic Proofs

Consider a GS proof system (Setup,Prove,Verify) of L equations. Each map $\iota_i : A_i \rightarrow B_i$ satisfies $\iota_i(x_1 + x_2) = \iota_i(x_1) + \iota_i(x_2)$, $\forall x_1, x_2 \in A_1$ and $i \in \{1, 2\}$.

We first define the **identity** $I_{GS} = (Sta_0, Wit_0, Proof_0)$. Sta_0 consists of L GS equations $(\boldsymbol{a}_0, \boldsymbol{b}_0, \Gamma_0, t_0)$, Wit_0 consists of L corresponding GS variables $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, $Proof_0$ consists of L corresponding GS proofs $(\boldsymbol{c}_0, \boldsymbol{d}_0, \boldsymbol{\pi}_0, \boldsymbol{\psi}_0)$, and there are L tuples of corresponding maps (ι_1, ι_2). They satisfy:

◇ Let m be the dimension of $\boldsymbol{b}_0$, $\boldsymbol{x}_0$ and $\boldsymbol{c}_0$. $\exists M \subseteq \{1, ..., m\}$ such that $\forall i \in M$, $b_0[i] = 0$; $\forall j \in \bar{M}$, $x_0[j] = 0$ and $c_0[j] = \iota_1(0)$, where $\bar{M} := \{1, ..., m\} \backslash M$.
◇ Let n be the dimension of $\boldsymbol{a}_0$, $\boldsymbol{y}_0$ and $\boldsymbol{d}_0$. $\exists N \subseteq \{1, ..., n\}$ such that $\forall i \in N$, $a_0[i] = 0$; $\forall j \in \bar{N}$, $y_0[j] = 0$ and $d_0[j] = \iota_2(0)$, where $\bar{N} := \{1, ..., n\} \backslash N$.
◇ For both $(\forall i \in \bar{M}, \forall j \in \bar{N})$ and $(\forall i \in M, \forall j \in N)$: $\Gamma_0[i, j] = 0$.
◇ $t_0 = 0$, $\boldsymbol{\pi}_0 = 0$, and $\boldsymbol{\psi}_0 = 0$.

We next define a **set** Π_{GS} of tuples (*Sta*, *Wit*, *Proof*) from the identity I_{GS}. *Sta* consists of L GS equations $(\boldsymbol{a}, \boldsymbol{b}, \Gamma, t)$ (corresponding to Sta_0's $(\boldsymbol{a}_0, \boldsymbol{b}_0, \Gamma_0, t_0)$ with m, n, M, N); *Wit* consists of L corresponding GS variables $(\boldsymbol{x}, \boldsymbol{y})$; *Proof* consists of L corresponding GS proofs $(\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\pi}, \boldsymbol{\psi})$; satisfying:

◇ $\forall i \in M$, $x[i] = x_0[i]$ and $c[i] = c_0[i]$. $\forall j \in \bar{M}$, $b[j] = b_0[j]$.
◇ $\forall i \in N$, $y[i] = y_0[i]$ and $d[i] = d_0[i]$. $\forall j \in \bar{N}$, $a[j] = a_0[j]$.
◇ If $(i \in \bar{M}) \vee (j \in \bar{N})$, then $\Gamma[i, j] = \Gamma_0[i, j]$. That means $\forall i \in \bar{M}, \forall j \in \bar{N}$: $\Gamma[i, j] = 0$.

We finally define **operation** $+_{GS} : \Pi_{GS} \times \Pi_{GS} \rightarrow \Pi_{GS}$. For $i \in \{1, 2\}$ and $(Sta_i, Wit_i, Proof_i) \in \Pi_{GS}$, Sta_i consists of L GS equations $(\boldsymbol{a}_i, \boldsymbol{b}_i, \Gamma_i, t_i)$ corresponding to Sta_0's $(\boldsymbol{a}_0, \boldsymbol{b}_0, \Gamma_0, t_0)$, Wit_i consists of L corresponding GS variables $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, and $Proof_i$ consists of L corresponding GS proofs $(\boldsymbol{c}_i, \boldsymbol{d}_i, \boldsymbol{\pi}_i, \boldsymbol{\psi}_i)$. We compute (*Sta*, *Wit*, *Proof*) $\leftarrow$ $(Sta_1, Wit_1, Proof_1) +_{GS} (Sta_2, Wit_2, Proof_2)$ of corresponding $(\boldsymbol{a}, \boldsymbol{b}, \Gamma, t)$, $(\boldsymbol{x}, \boldsymbol{y})$ and $(\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\pi}, \boldsymbol{\psi})$ as follows.

◇ $\forall i \in M$: $x[i] := x_1[i]$; $c[i] := c_1[i]$; $b[i] := b_1[i] + b_2[i]$. $\forall j \in \bar{M}$: $b[j] := b_1[j]$; $x[j] := x_1[j] + x_2[j]$; $c[j] := c_1[j] + c_2[j]$.
◇ $\forall i \in N$: $y[i] := y_1[i]$; $d[i] := d_1[i]$; $a[i] := a_1[i] + a_2[i]$. $\forall j \in \bar{N}$: $a[j] := a_1[j]$; $y[j] := y_1[j] + y_2[j]$; $d[j] := d_1[j] + d_2[j]$.
◇ If $(i \in \bar{M}) \vee (j \in \bar{N})$, then $\Gamma[i, j] := \Gamma_1[i, j]$. Otherwise, $\Gamma[i, j] := \Gamma_1[i, j] + \Gamma_2[i, j]$.
◇ $t = t_1 + t_2$, $\boldsymbol{\pi} = \boldsymbol{\pi}_1 + \boldsymbol{\pi}_2$, and $\boldsymbol{\psi} = \boldsymbol{\psi}_1 + \boldsymbol{\psi}_2$.

Theorem 1. *In the definitions above, Π_{GS} is a set of strongly homomorphic proofs with operation $+_{GS}$ and the identity element I_{GS}.*

Proof of theorem 1 can be found in Tech Report [29]. The proof validates the closure, associativity, commutativity, identity element, and inverse element properties of abelian groups.

3.3 Comparison with the DHLW Homomorphic NIZK

We compare our homomorphic proof approach with the independently proposed DHLW homomorphic NIZK [27]. Intuitively, DHLW defines that a NIZK proof system is homomorphic if for any $(Para, Sta_1, Wit_1), (Para, Sta_2, Wit_2) \in \mathbf{R}$: $\mathsf{Prove}(Para, Sta_1, Wit_1)_{Rand_1} + \mathsf{Prove}(Para, Sta_2, Wit_2)_{Rand_2} = \mathsf{Prove}(Para, Sta_1 + Sta_2, Wit_1 + Wit_2)_{Rand_1 + Rand_2}$, where $\mathsf{Prove}(\ldots)_{Rand}$ is the output of $\mathsf{Prove}()$ with randomness *Rand*. The new definition in this paper requires homomorphism for a subset of proofs generated by Prove, and differs from DHLW's homomorphism requirement for all such proofs, covering more proof systems.

The DHLW's homomorphic NIZK construction a special case of our construction above. It is for statements of 'one-sided' GS equations $\{\boldsymbol{x}_k \cdot \boldsymbol{b}_k = t_k\}_{k=1}^{L}$ whereas our construction generalizes to statements of 'full' GS equations $\{\boldsymbol{a}_k \cdot \boldsymbol{y}_k + \boldsymbol{x}_k \cdot \boldsymbol{b}_k + \boldsymbol{x}_k \cdot \Gamma \boldsymbol{y}_k = t_k\}_{k=1}^{L}$. As shown later, the ADNMP and RDAC are based on a GS homomorphic proof system of 'full' equations $\{(y_1 + y_2)X_{j1} + y_{j3}A_1 = T_{j1} \wedge X_{j3} - y_{j3}A_2 = 0 \wedge y_{j3}X_{j2} = T_{j2}\}_{j=1}^{m}$.

4 Accumulator with Delegatable NM Proofs - ADNMP

We refer to a universal accumulator as (Setup, ProveNM, VerifyNM, CompNMWit, Accu), that consists of only algorithms for setup; generating, verifying and computing witnesses for **non-membership** proofs; and accumulating, respectively. This paper does not deal with membership proofs. Tech Report [29] provides more details on accumulators.

The delegating ability to prove statements allows another user to prove the statements on one's behalf without revealing the witness, even if the statements' conditions change over time. For privacy reasons, adversaries should not be able to distinguish different delegations from different users. The delegatee can verify a delegation and unlinkably redelegate the proving ability further to other users. Thus, delegating an accumulator's NM proofs should meet 4 conditions formalized in Definition 2. *Delegability* means that an element *Ele*'s owner can delegate her ability to prove that *Ele* is not accumulated without trivially revealing *Ele*. Even if the set of accumulated elements change overtime, the delegatee does not need to contact the delegator again to generate the proof. The owner gives the delegatee a key *De* generated from *Ele*. The proof generated from *De* by CompNMProof is indistinguishable from a proof generated by ProveNM. *Unlinkability* means that a delegatee should not be able to distinguish whether or not two delegating keys originate from the same element. It implies that it is computationally hard to find an element from its delegating keys. *Redelegability* means that the delegatee can redelegate *De* as *De′* to other users, and still maintains indistinguishability of *De* and *De′*. *Verifiability* means that one is able to validate that a delegating key *De* is correctly built.

Definition 2. *A universal accumulator (*Setup*,* ProveNM*,* VerifyNM*,* CompNMWit*,* Accu*) is a secure ADNMP (Accumulator with Delegatable NM Proofs) if there exist PPT algorithms*

– Dele*: takes public parameters Para and an element Ele and returns its delegating key De;*
– Rede*: takes Para and a delegating key De and returns another delegating key De′;*
– Vali*: takes Para and a delegating key De and returns* 1 *if De is valid or* 0 *otherwise;*
– CompNMProof*: takes Para, De, an accumulator set AcSet and its accumulator value AcVal and returns an NM proof that the element Ele corresponding to De is not accumulated in AcSet;*

satisfying:

- *Delegability: For every PPT algorithm* $(\mathcal{A}_1, \mathcal{A}_2), |Pr[\,(Para, Aux) \leftarrow \mathsf{Setup}(1^k);$ $(Ele, AcSet, state) \leftarrow \mathcal{A}_1(Para);$ $AcVal \leftarrow \mathsf{Accu}(Para, AcSet);$ $Wit \leftarrow \mathsf{CompNMWit}(Para, Ele, AcSet, AcVal);$ $Proof_0 \leftarrow \mathsf{ProveNM}(Para, AcVal, Wit);$ $De \leftarrow \mathsf{Dele}(Para, Ele);$ $Proof_1 \leftarrow \mathsf{CompNMProof}(Para, De, AcSet, AcVal);$ $b \leftarrow \{0,1\};$ $b' \leftarrow \mathcal{A}_2(state, AcVal, Wit, De, Proof_b)$: $(Ele \notin AcSet) \wedge b = b'] - 1/2|$ *is negligible.*
- *Unlinkability: For every PPT algorithm* $\mathcal{A}$, $|Pr[\,(Para, Aux) \leftarrow \mathsf{Setup}(1^k);$ $(Ele_0, Ele_1) \leftarrow Dom_{Para};$ $De \leftarrow \mathsf{Dele}(Para, Ele_0);$ $b \leftarrow \{0,1\};$ $De_b \leftarrow \mathsf{Dele}(Para, Ele_b);$ $b' \leftarrow \mathcal{A}(Para, De, De_b)$: $b = b'] - 1/2|$ *is negligible.*
- *Redelegability: For every PPT algorithms* $(\mathcal{A}_1, \mathcal{A}_2)$, $|Pr[\,(Para, Aux) \leftarrow \mathsf{Setup}(1^k);$ $(Ele, state) \leftarrow \mathcal{A}_1(Para);$ $De \leftarrow \mathsf{Dele}(Para, Ele);$ $De_0 \leftarrow \mathsf{Dele}(Para, Ele);$ $De_1 \leftarrow \mathsf{Rede}(Para, De);$ $b \leftarrow \{0,1\};$ $b' \leftarrow \mathcal{A}_2(state, De, De_b)$: $b = b'] - 1/2|$ *is negligible.*
- *Verifiability: For every PPT algorithm* $\mathcal{A}$, $|Pr[\,(Para, Aux) \leftarrow \mathsf{Setup}(1^k);$ $Ele \leftarrow \mathcal{A}(Para);$ $De \leftarrow \mathsf{Dele}(Para, Ele)$: $\mathsf{Vali}(Para, De) = 1$ *if* $Ele \in Dom_{Para}] - 1|$ *and* $|Pr[(Para, Aux) \leftarrow \mathsf{Setup}(1^k); De' \leftarrow \mathcal{A}(Para) : \mathsf{Vali}(Para, De') = 0$ *if* $De' \notin \{De | De \leftarrow \mathsf{Dele}(Para, Ele');\ Ele' \in Dom_{Para}\}] - 1|$ *are negligible.*

Unlinkability combined with Redelegability generalizes the Unlinkability definition allowing an adversary $\mathcal{A}$ access an oracle $\mathcal{O}(Para, De)$ that returns another delegating key De' of the same element corresponding to De. That means $\mathcal{A}$ can get several delegating keys of Ele_0 and of Ele_b using $\mathcal{O}$. Rede can be used for such an oracle.

For any ADNMP, given an element Ele and a delegating key De, one can tell if De is generated by Ele as follows. First, she does not accumulate Ele and uses De to prove that De's element is not accumulated. Then she accumulates Ele and tries to prove again that De's element is not accumulated. If she cannot prove that anymore, she can conclude that Ele is De's element. Due to this restriction, in ADNMP's applications, Ele should be a secret that only its owner knows. This is related to the discussion in Tech Report [29] about the general conflict between delegability and anonymity.

5 An ADNMP Scheme

We propose a dynamic universal ADNMP. Its Setup, Accu and UpdateVal are generalized from [7,10].

◇ Setup: We need GS instantiations where GS proofs of this accumulator are composable ZK. We can use either the SXDH or SDLIN (Symmetric DLIN) [28] instantiations. We use SXDH as an example. Generate parameters $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ and CRS σ with perfectly binding keys for the SXDH instantiation of GS proofs (Sections 2), and auxiliary information $Aux = \delta \leftarrow \mathbb{Z}_p^*$. For the proof, generate $A \leftarrow \mathbb{G}_1$ and $\tau := \iota_2'(\delta)$. For efficient accumulating without Aux, a tuple $\varsigma = (P_1, \delta P_1, \ldots, \delta^{q+1} P_1)$ is needed, where

$q \in \mathbb{Z}_p^*$. The domain for elements to be accumulated is $\mathbb{D} = \mathbb{Z}_p^* \backslash \{-\delta\}$. We have $Para = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, A, \sigma, \varsigma, \tau)$.

- ◇ Accu: On input $AcSet = \{a_1, ..., a_Q\} \subset \mathbb{D}$, compute $m = \lceil Q/q \rceil$. If $Aux = \delta$ is available, the output $AcVal$ is a set of m component accumulator values $\{V_j\}_{j=1}^m$ computed as $V_j = \prod_{i=(j-1)q+1; i<Q}^{jq} (\delta + a_i)\delta P_1$. If Aux is not available, $AcVal$ is efficiently computable from ς and $AcSet$.
- ◇ UpdateVal: In case $a' \in \mathbb{D}$ is being accumulated; from 1 to m, find the first V_j that hasn't accumulated q elements, and update $V_j' = (\delta + a')V_j$; if such V_j isn't found, add $V_{m+1} = (\delta + a')\delta P_1$. In case a' is removed from $AcVal$, find V_j which contains a' and update $V_j' = 1/(\delta + a')V_j$.

In previous accumulators [7,10], the accumulator value is a single value $V = \prod_{a_i \in AcSet}(\delta + a_i)\delta P_1$ and they require that q of ς is the upper bound on the number of elements to be accumulated, i.e. $m = 1$. The above generalization, where the accumulator value is a set of V instead, relaxes this requirement and allows the ADNMP scheme to work even when q is less than the number of accumulated elements. It also allows smaller q at setup.

5.1 NM Proof

We need to prove that an element $y_2 \in \mathbb{D}$ is not in any component accumulator value V_j of $AcVal$ $\{V_j\}_{j=1}^m$. Suppose V_j accumulates $\{a_1, ..., a_k\}$ where $k \leq q$, denote $Poly(\delta) := \prod_{i=1}^k (\delta + a_i)\delta$, then $V_j = Poly(\delta)P_1$. Let y_{j3} be the remainder of polynomial division $Poly(\delta)$ mod $(\delta + y_2)$ in Z_p, and X_{j1} be scalar product of the quotient and P_1. Similar to [10], the idea for constructing NM proofs is that y_2 is not a member of $\{a_1, ..., a_k\}$ if and only if $y_{j3} \neq 0$. We have the following equation between δ, y_2, y_{j3} and X_{j1}: $(\delta + y_2)X_{j1} + y_{j3}P_1 = V_j$. Proving this equation by itself does not guarantee that y_{j3} is the remainder of the polynomial division above. It also needs to prove the knowledge of $(y_{j3}P_2, y_{j3}A)$ and the following Extended Strong DH (ESDH) assumption. It is a variation of the Hidden Strong DH (HSDH) assumption [30], though it is not clear which assumption is stronger. It is in the extended uber-assumption family [31] and can be proved in generic groups, similar to HSDH.

Definition. *q-**ESDH**: Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ be bilinear parameters, $A \leftarrow \mathbb{G}_1^*$ and $\delta \leftarrow \mathbb{Z}_p^*$. Given P_1, δP_1, ..., $\delta^{q+1}P_1, A, P_2, \delta P_2$, it is computationally hard to output $(\frac{y_3}{\delta + y_2}P_1, y_2, y_3P_2, y_3A)$ where $y_3 \neq 0$.*

We will show later that if one can prove the knowledge of $(y_{j3}P_2, y_{j3}A)$ satisfying $(\delta + y_2)X_{j1} + y_{j3}P_1 = V_j$ and y_2 is accumulated in V_j but $y_{j3} \neq 0$, then she can break the assumption. To prove the knowledge of $(y_{j3}P_2, y_{j3}A)$, we need equation $X_{j3} - y_{j3}A = 0$. To verify $y_{j3} \neq 0$, we need equation $T_j = y_{j3}X_{j2}$ and the verifier checks $T_j \neq 0$. We now present the NM proof and its security in theorem 2. Proof of theorem 2 can be found in Tech Report [29].

- ◇ CompNMWit takes y_2, and for each component accumulator value V_j of $AcVal$ $\{V_j\}_{j=1}^m$, computes remainder y_{j3} of $Poly(\delta)$ mod $(\delta + y_2)$ in Z_p which is efficiently computable from $\{a_1, ..., a_k\}$ and y_2. It then computes $X_{j1} = (Poly(\delta) - y_{j3})/(\delta + y_2)P_1$, which is efficiently computable from $\{a_1, ..., a_k\}, y_2$ and ς. The witness includes y_2 and $\{(X_{j1}, X_{j3} = y_{j3}A, y_{j3})\}_{j=1}^m$. UpdateNMWit is for one V_j at a time and similar to [10] with the extra task of updating $X_{j3} = y_{j3}A$.

◇ ProveNM generates $X_{j2} \leftarrow \mathbb{G}_1^*$ and outputs $T_j = y_{j3}X_{j2}$ for each V_j and a GS proof for the following equations of variables $y_1 = \delta, y_2, \{(X_{j1}, X_{j3}, X_{j2}, y_{j3})\}_{j=1}^m$. $\bigwedge_{j=1}^m((y_1 + y_2)X_{j1} + y_{j3}P_1 = V_j \wedge X_{j3} - y_{j3}A = 0 \wedge y_{j3}X_{j2} = T_j)$.
Note that the prover does not need to know y_1. From τ, it is efficient to generate a commitment of δ and the proof.

◇ VerifyNM verifies the proof generated by ProveNM and checks that $T_j \neq 0$, $\forall j \in \{1, \ldots, m\}$. It accepts if both of them pass or rejects otherwise.

Theorem 2. *The proof system proves that an element is not accumulated. Its soundness depends on the ESDH assumption. Its composable ZK depends on the assumption underlying the GS instantiation (SXDH or SDLIN).*

The proof in [29] follows the GS SXDH instantiation and shows that the NM proof system for this accumulator is *composable* ZK. The *completeness* comes from GS, and *soundness* relies on the ESDH assumption.

5.2 NM Proofs Are Strongly Homomorphic

We can see that for the same constant A, the same variables δ, y_2 and X_{j2} with the same commitments, the set of NM proofs has the form of strongly homomorphic GS proofs constructed in Section 3. For constructing delegatable NM proofs, we just need them to be homomorphic. More specifically, 'adding' 2 proofs of 2 sets of equations (with the same commitments for δ, y_2 and X_{j2})
$\bigwedge_{j=1}^m((\delta + y_2)X_{j1}^{(1)} + y_{j3}^{(1)}P_1 = V_j^{(1)} \wedge X_{j3}^{(1)} - y_{j3}^{(1)}A = 0 \wedge y_{j3}^{(1)}X_{j2} = T_j^{(1)})$ and
$\bigwedge_{j=1}^m((\delta + y_2)X_{j1}^{(2)} + y_{j3}^{(2)}P_1 = V_j^{(2)} \wedge X_{j3}^{(2)} - y_{j3}^{(2)}A = 0 \wedge y_{j3}^{(2)}X_{j2} = T_j^{(2)})$ form a proof of equations
$\bigwedge_{j=1}^m((\delta + y_2)X_{j1} + y_{j3}P_1 = V_j \wedge X_{j3} - y_{j3}A = 0 \wedge y_{j3}X_{j2} = T_j)$
where $X_{j1} = X_{j1}^{(1)} + X_{j1}^{(2)}$, $X_{j3} = X_{j3}^{(1)} + X_{j3}^{(2)}$, $y_{j3} = y_{j3}^{(1)} + y_{j3}^{(2)}$, $V_j = V_j^{(1)} + V_j^{(2)}$ and $T_j = T_j^{(1)} + T_j^{(2)}$.

5.3 Delegating NM Proof

We first explain the idea behind the accumulator's delegatable NM proof construction. We write the component accumulator value $V = \prod_{i=1}^k(\delta + a_i)\delta P_1$ as $V = \sum_{i=0}^k b_i\delta^{k+1-i}P_1$ where $b_0 = 1$ and $b_i = \sum_{1 \leq j_1 < j_2 < \ldots < j_i \leq k}\prod_{l=1}^i a_{j_l}$. Thus, V can be written as a linear combination of $\delta P_1, \ldots, \delta^{k+1}P_1$ in ς.

Next, we construct homomorphic proofs for $(\delta + y_2)X_1^{(i)} + y_3^{(i)}P_1 = \delta^i P_1 \wedge X_3^{(i)} - y_3^{(i)}A = 0 \wedge y_3^{(i)}X_2 = T^{(i)}$ where $i \in \{1, ..., k+1\}$. Using the same linear combination of $\delta P_1, \ldots, \delta^{k+1}P_1$ for V, we linearly combine these proofs to get a proof for $(\delta + y_2)X_1 + y_3P_1 = V \wedge X_3 - y_3A = 0 \wedge y_3X_2 = T$, where

$X_1 = \sum_{i=0}^{k} b_i X_1^{(k+1-i)}$, $X_3 = \sum_{i=0}^{k} b_i X_3^{(k+1-i)}$, $y_3 = \sum_{i=0}^{k} b_i y_3^{(k+1-i)}$ and $T = \sum_{i=0}^{k} b_i T^{(k+1-i)}$. This is the same as the NM proof for each of the component accumulator value provided above.

We now provide the algorithms for delegating NM proofs and its security theorem. We also add UpdateProof to be used in place of CompNMProof when possible for efficiency.

- ◇ Dele($Para, Ele$). For each $i \in \{1, ..., q+1\}$, compute remainder $y_3^{(i)}$ of δ^i mod $(\delta + y_2)$ in Z_p, and $X_1^{(i)} = (\delta^i - y_3^{(i)})/(\delta + y_2)P_1$, which are efficiently computable from y_2 and ς. In fact, we have $y_3^{(i)} = (-1)^i y_2^i$ and $X_1^{(i+1)} = \sum_{j=0}^{i} (-1)^j y_2^j \delta^{i-j} P_1 = \delta^i P_1 - y_2 X_1^{(i)}$ (so the cost of computing all $X_1^{(i)}$, $i \in \{1, ..., q+1\}$ is about q scalar products). Generate $X_2 \leftarrow \mathbb{G}_1^*$, the delegation key De includes $\{T^{(i)} = y_3^{(i)} X_2\}_{i=1}^{q+1}$ and a GS proof of equations $\bigwedge_{i=1}^{q+1} ((\delta + y_2)X_1^{(i)} + y_3^{(i)} P_1 = \delta^i P_1 \wedge X_3^{(i)} - y_3^{(i)} A = 0 \wedge y_3^{(i)} X_2 = T^{(i)})$.
- ◇ Rede($Para, De$). For each $i \in \{1, ..., q+1\}$, extract proof $Proof_i$ of $y_3^{(i)} X_2 = T^{(i)}$ in De. In each $Proof_i$, for the same $y_3^{(i)}$ and its commitment, $Proof_i$ is of homomorphic form. So generate $r \leftarrow Z_p^*$ and compute $Proof_i' = rProof_i$ which is a proof of $y_3^{(i)} X_2' = T'^{(i)}$, where $X_2' = rX_2$ and $T'^{(i)} = rT^{(i)}$. Note that commitments of $y_3^{(i)}$ stay the same. For every $i \in \{1, ..., q+1\}$, replace $T^{(i)}$ by $T'^{(i)}$ and $Proof_i$ by $Proof_i'$ in De to get a new GS proof, which is then randomized to get the output De'.
- ◇ Vali($Para, De$). A simple option is to verify the GS proof De. An alternative way is to use batch verification: Divide De into proofs $NMProof_i$ of $(\delta + y_2)X_1^{(i)} + y_3^{(i)} P_1 = \delta^i P_1 \wedge X_3^{(i)} - y_3^{(i)} A = 0 \wedge y_3^{(i)} X_2 = T^{(i)}$ for $i \in \{1, ..., q+1\}$. Generate $q+1$ random numbers to linearly combine $NMProof_i$s and their statements and verify the combined proof and statement.
- ◇ CompNMProof($Para, De, AcSet, AcVal$). Divide De into proofs $NMProof_i$ as in Vali. For each component accumulator value V of $\{a_1, ..., a_k\}$, compute b_i for $i \in \{0, ..., k\}$ as above. $NMProof_i$s belong to a set of homomorphic proofs, so compute $NMProof = \sum_{i=0}^{k} b_i NMProof_{k+1-i}$, which is a proof of $(\delta + y_2)X_1 + y_3 P_1 = V \wedge X_3 - y_3 A = 0 \wedge y_3 X_2 = T$ where X_1, X_3, y_3, T and V are as explained above.

  Extract proof $SubProof$ of $y_3 X_2 = T$ in $NMProof$. For the same y_3 and its commitment, $SubProof$ is of homomorphic form. So generate $r \leftarrow Z_p^*$ and compute $SubProof' = rSubProof$ which is a proof of $y_3 X_2' = T'$, where $X_2' = rX_2$ and $T' = rT$. Note that y_3's commitment stays the same. Replace T by T' and $SubProof$ by $SubProof'$ in $NMProof$ to get a new proof $NMProof'$.

  Concatenate those $NMProof'$ of all V in $AcVal$ and output a randomization of the concatenation.
- ◇ UpdateProof($Para$,De,$AcSet$,$AcVal$,$Proof$,$Opens$). $Proof$ is the proof to be updated and $Opens$ contains openings for randomizing commitments of $y_1 = \delta$ and y_2 from De to $Proof$. Suppose there is a change in accumulated elements of a component value V, we just compute $NMProof'$ for the

updated V as in CompNMProof. Randomize $NMProof'$ so that its commitments of y_1 and y_2 are the same as those in $Proof$ and put it in $Proof$ in place of its old part. Output a randomization of the result.

To prove that this construction provides an ADNMP, we need the following Decisional Strong Diffie Hellman (DSDH) assumption, which is not in the uber-assumption family [31], but can be proved in generic groups similarly to the PowerDDH assumption [32]. Proof of theorem 3 is in Tech Report [29].

Definition. *q-**DSDH**: Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ be bilinear parameters, $B_0, B_1 \leftarrow \mathbb{G}_1^*$, $x_0, x_1 \leftarrow \mathbb{Z}_p^*$ and $b \leftarrow \{0, 1\}$. Given $B_0, x_0B_0, \ldots, x_0^qB_0$, B_1, x_bB_1, $\ldots$, $x_b^qB_1$, no PPT algorithm can output $b' = b$ with a probability non-negligibly better than a random guess.*

Theorem 3. *The accumulator is a secure ADNMP, based on ESDH, DSDH and the assumption underlying the GS instantiation (SXDH or SDLIN).*

6 Revocable Delegatable Anonymous Credentials - RDAC

6.1 Model

This is a model of RDAC systems, extended from BCCKLS [1] which is briefly described in Tech Report [29]. Participants include users and a Blacklist Authority (BA) owning a blacklist BL. For each credential proof, a user picks a new nym indistinguishable from her other nyms. We need another type of nym for revocation, called *r-nym*, to distinguish between two types of nyms. When an r-nym is revoked, its owner cannot prove credentials anymore. The PPT algorithms are:

- Setup(1^k) outputs public parameters $Para_{DC}$, BA's secret key Sk_{BA}, and an initially empty blacklist BL. Denote BL_e an empty blacklist.
- KeyGen$(Para_{DC})$ outputs a secret key Sk and a secret r-nym Rn for a user.
- NymGen$(Para_{DC}, Sk, Rn)$ outputs a new nym Nym with an auxiliary key $Aux(Nym)$. A user O can become a root credential issuer by publishing a nym Nym_O and a proof that her r-nym Rn_O is not revoked that O has to update when BL changes.
- Issue$(Para_{DC}, Nym_O, Sk_I, Rn_I, Nym_I, Aux(Nym_I), Cred, DeInf, Nym_U, BL, L) \leftrightarrow$ Obtain$(Para_{DC}, Nym_O, Sk_U, Rn_U, Nym_U, Aux(Nym_U), Nym_I, BL, L)$ lets user I issue a level $L+1$ credential to user U. Sk_I, Rn_I, Nym_I and $Cred$ are the secret key, r-nym, nym and level L credential rooted at Nym_O of issuer I. Sk_U, Rn_U and Nym_U are the secret key, r-nym and nym of user U. I gets no output and U gets a credential $Cred_U$.

  Delegation information $DeInf$ is optional. When it is included, U also gets delegation information $DeInf_U$ to later prove that r-nyms of all delegators in her chain are not revoked. If $L = 0$ then $Cred$ is omitted and $DeInf = 1$ is optionally included.

- $\mathsf{Revoke}(Para_{DC}, Sk_{BA}, Rn, BL)$ updates BL so that a revoked user Rn can no longer prove, delegate or receive credentials. Denote $Rn \in BL$ or $Rn \notin BL$ that Rn is blacklisted or not, respectively.
- $\mathsf{CredProve}(Para_{DC}, Nym_O, Cred, DeInf, Sk, Rn, Nym, Aux(Nym), BL, L)$ takes a level L credential $Cred$, Sk, Rn and optionally $DeInf$ to output $CredProof$, which proves that: (i) a credential level L is issued to Nym's owner. (ii) Nym's Rn is not revoked. (iii)(optional, when $DeInf$ is included) all r-nyms on the credential's chain are not revoked.
- $\mathsf{CredVerify}(Para_{DC}, Nym_O, CredProof, Nym, BL, L)$ verifies if $CredProof$ is a valid proof of the above statements.

The differences with the model for delegatable anonymous credentials without revocation [1] are the introductions of BA with Sk_{BA} and BL; r-nyms; delegation information $DeInf$; Revoke; and the two $CredProof$'s conditions (ii) and (iii). Note that $DeInf$'s inclusion in the algorithms is optional and allows a user the choice to either just prove that she is not blacklisted or fully prove and delegate that all users on her credential chain are not blacklisted. We can use one of traditional methods for BA to obtain r-nyms to revoke (Tech Report [29]).

Tech Report [29] formally defines RDAC security. Briefly, there are 3 requirements extended from the security definition of delegatable anonymous credentials [1]: Correctness, Anonymity and Unforgeability. Tech Report [29] discusses the trade offs between delegability and anonymity.

7 An RDAC Scheme

7.1 Overview

We first describe intuitions of the BCCKLS delegatable anonymous credential scheme in [1], and then show how ADNMP extends it to provide revocation.

BCCKLS uses an F-Unforgeable certification secure authentication scheme $\mathcal{AU}$ of PPT algorithms $\mathsf{AtSetup}$, AuthKg, Authen, $\mathsf{VerifyAuth}$. $\mathsf{AtSetup}(1^k)$ returns public parameters $Para_{At}$, $\mathsf{AuthKg}(Para_{At})$ generates a key Sk, $\mathsf{Authen}(Para_{At}, Sk, \boldsymbol{m})$ produces an authenticator $Auth$ authenticating a vector of messages $\boldsymbol{m}$, and $\mathsf{VerifyAuth}(Para_{At}, Sk, \boldsymbol{m}, Auth)$ accepts if and only if $Auth$ validly authenticates $\boldsymbol{m}$ under Sk. The scheme's *security* requirements include *F-Unforgeability* [12] for a bijective function F, which means $(F(\boldsymbol{m}), Auth)$ is unforgeable without obtaining an authenticator on $\boldsymbol{m}$; and *certification security*, which means no PPT adversary, even after obtaining an authenticator by the challenge secret key, can forge another authenticator. An adversary can also have access to two oracles. $\mathcal{O}_{Authen}(Para_{At}, Sk, \boldsymbol{m})$ returns $\mathsf{Authen}(Para_{At}, Sk, \boldsymbol{m})$ and $\mathcal{O}_{Certify}(Para_{At}, Sk^*, (Sk, m_2, \ldots, m_n))$ returns $\mathsf{Authen}(Para_{At}, Sk^*, (Sk, m_2, \ldots, m_n))$. BCCKLS also uses a secure two party computation protocol ($\mathsf{AuthPro}$) to obtain a NIZKPK of an authenticator on $\boldsymbol{m}$ without revealing anything about $\boldsymbol{m}$.

In BCCKLS, a user U can generate a secret key $Sk \leftarrow \mathsf{AuthKg}(Para_{At})$, and many nyms $Nym = \mathsf{Com}(Sk, Open)$ by choosing different values $Open$. Suppose U has a level $L+1$ credential from O, let $(Sk_0 = Sk_O, Sk_1, \ldots, Sk_L, Sk_{L+1} = Sk)$ be the keys such that Sk_i's owner delegated the credential to Sk_{i+1}, and let H :

$\{0,1\}^* \to Z_p$ be a collision resistant hash function. $r_i = H(Nym_O, atributes, i)$ is computed for a set of attributes for that level's credential. U generates a proof of her delegated credential as
$CredProof \leftarrow \mathsf{NIZKPK}[Sk_O \text{ in } Nym_O, Sk \text{ in } Nym]$
$\{(F(Sk_O), F(Sk_1), ..., F(Sk_L), F(Sk), auth_1, ..., auth_{L+1}) :$
$\mathsf{VerifyAuth}(Sk_O, (Sk_1, r_1), auth_1) \wedge$
$\mathsf{VerifyAuth}(Sk_1, (Sk_2, r_2), auth_2) \wedge ... \wedge$
$\mathsf{VerifyAuth}(Sk_{L-1}, (Sk_L, r_L), auth_L) \wedge$
$\mathsf{VerifyAuth}(Sk_L, (Sk, r_{L+1}), auth_{L+1})\}$.

Now we show how ADNMP extends BCCKLS to provide revocation. Using ADNMP, BA's blacklist BL includes an accumulated set of revoked Rns and its accumulator value. Beside a secret key Sk, user U has a secret r-nym Rn in the accumulator's domain, and generates nyms $Nym = (\mathsf{Com}(Sk, Open_{Sk}),$ $\mathsf{Com}(Rn, Open_{Rn}))$. ADNMP allows delegation and redelegation of a proof that an Rn is not accumulated in a blacklist $Rn \notin BL$. U generates a proof of her delegated credential and validity of the credential's chain as follows.

$$\begin{aligned}CredProof \leftarrow \mathsf{NIZKPK}[&Sk_O \text{ in } Nym_O[1], Sk \text{ in } Nym[1], Rn \text{ in } Nym[2]]\\ &\{(F(Sk_O), F(Sk_1), F(Rn_1), ..., F(Sk_L), F(Rn_L), F(Sk), F(Rn),\\ &\quad auth_1, ..., auth_L, auth_{L+1}) :\\ &\mathsf{VerifyAuth}(Sk_O, (Sk_1, Rn_1, r_1), auth_1) \wedge (Rn_1 \notin BL) \wedge\\ &\mathsf{VerifyAuth}(Sk_1, (Sk_2, Rn_2, r_2), auth_2) \wedge (Rn_2 \notin BL) \wedge ... \wedge\\ &\mathsf{VerifyAuth}(Sk_{L-1}, (Sk_L, Rn_L, r_L), auth_L) \wedge (Rn_L \notin BL) \wedge\\ &\mathsf{VerifyAuth}(Sk_L, (Sk, Rn, r_{L+1}), auth_{L+1}) \wedge (Rn \notin BL)\}.\end{aligned}$$

Delegability allows a user, on behalf of the user's delegators without any witness, to prove that the user's ancestor delegators are not included in a changing blacklist. The proofs a user and a delegator generates are indistinguishable from each other. Redelegability allows a user to redelegate those proofs on the delegators to the user's delegatees. Unlinkability prevents colluding users to link delegations of the same delegator. Verifiability allows a user to validate the correctness of a delegation token.

7.2 Description

The RDAC scheme has several building blocks. (i) An ADNMP with a malleable NM proof system (NMPS) of AcSetup, ProveNM, VerifyNM, CompNMWit, Accu, Dele, Rede, Vali, CompNMProof, with commitment ComNM. (ii) Those from BCCKLS, including $\mathcal{AU}$; AuthPro; H; and a malleable NIPK credential proof system (CredPS) of PKSetup, PKProve, PKVerify, RandProof, with commitment Com. (iii) A malleable proof system (EQPS), with PKSetup and AcSetup in setup, to prove that two commitments Com and ComNM commit to the same value.

Assume that a delegating key De contains a commitment of element Ele. CompNMProof and Rede randomize the commitment in De and generate Ele's commitment. Elements of the accumulator domain and the authenticator's key space can be committed by Com. The following algorithm inputs are the same as in the model and omitted.

- Setup: Use $\mathsf{PKSetup}(1^k)$, $\mathsf{AtSetup}(1^k)$ and $\mathsf{AcSetup}(1^k)$ to generate $Para_{PK}$, $Para_{At}$, and ($Para_{Ac}$, Aux_{Ac}). The blacklist includes an accumulated set of revoked r-nyms and its accumulator value. Output an initial blacklist BL with an empty accumulator set and its initial accumulator value, $Para_{DC} = (Para_{PK}, Para_{At}, Para_{Ac}, H)$, and $Sk_{BA} = Aux_{Ac}$.
- KeyGen: Run $\mathsf{AuthKg}(Para_{At})$ to output a secret key Sk. Output a random r-nym Rn from the accumulator's domain.
- NymGen: Generate random $Open_{Sk}$ and $Open_{Rn}$, and output nym $Nym = (\mathsf{Com}(Sk, Open_{Sk}),\mathsf{Com}(Rn, Open_{Rn}))$ and $Aux(Nym) = (Open_{Sk}, Open_{Rn})$.
- The credential originator O publishes a Nym_O and a proof $NMProof_O$ that Rn_O is not revoked. O updates the proof when BL changes.
- $\mathsf{Issue} \leftrightarrow \mathsf{Obtain}$: If $L = 0$ and $Nym_O \neq Nym_I$, aborts. Issue aborts if $Nym_I \neq (\mathsf{Com}(Sk_I, Open_{Sk_I}), \mathsf{Com}(Rn_I, Open_{Rn_I}))$ or $\mathsf{PKVerify}(Para_{PK}, (Nym_0, (\mathsf{Com}(Sk_I, 0), \mathsf{Com}(Rn_I, 0))), Cred)$ rejects, or $Rn_I \in BL$, or Nym_U is invalid. Obtain aborts if $Nym_U \neq (\mathsf{Com}(Sk_U, Open_{Sk_U})$, $\mathsf{Com}($ Rn_U, $Open_{Rn_U}))$ or $Rn_U \in BL$. Otherwise, each of Issue and Obtain generates a proof and verifies each other's proofs that $Rn_I \notin BL$ and $Rn_U \notin BL$ using ($\mathsf{ProveNM}$, $\mathsf{VerifyNM}$) with EQPS (to prove that $\mathsf{Com}(Rn_I)$ in Nym_I and $\mathsf{ComNM}(Rn_I)$ generated by $\mathsf{ProveNM}$ commit to the same value Rn_I, and similarly for Rn_U). They then both compute $r_{L+1}{=}H(Nym_O, attributes, L{+}1)$ for a set of attributes for that level's credential. They run $\mathsf{AuthPro}$ for the user U to receive: $Proof_U \leftarrow \mathsf{NIZKPK}[Sk_I$ in $Nym_I[1]$, Sk_U in $\mathsf{Com}(Sk_U, 0)$, Rn_U in $\mathsf{Com}(Rn_U, 0)]$ $\{(F(Sk_I), F(Sk_U), F(Rn_U), auth) : \mathsf{VerifyAuth}(Sk_I, (Sk_U, Rn_U, r_{L+1}), auth)\}$. U's output is $Cred_U = Proof_U$ when L = 0. Otherwise, suppose the users on the issuer I's chain from the root are 0 (same as O), 1, 2,..., L (same as I). I randomizes $Cred$ to get a proof $CredProof_I$ (containing the same Nym_I) that for every Nym_j on I's chain ($j \in \{1, ..., L\}$), Sk_j and Rn_j are authenticated by Sk_{j-1} (with r_j). U verifies that $\mathsf{PKVerify}(Para_{PK}, (Nym_0, Nym_I), CredProof_I)$ accepts, then concatenates $Proof_U$ and $CredProof_I$ and projects Nym_I from statement to proof to get $Cred_U$.

  The optional $DeInf$ includes a list of delegating keys De_js generated by the accumulator's Dele to prove that each Rn_j is not accumulated in the blacklist, and a list of $EQProof_j$ for proving that two commitments of Rn_j in $Cred$ and De_j commit to the same value Rn_j, for $j \in \{1, ..., L-1\}$. Verifying $DeInf$ involves checking $\mathsf{Vali}(Para_{Ac}, De_j)$ and $EQProof_j$, for $j \in \{1, ..., L-1\}$. When $DeInf$ is in the input, Issue would aborts without interacting with Obtain if verifying $DeInf$ fails. Otherwise, it uses $\mathsf{CompNMProof}$ to generate a proof $NMChainProof$ that each Rn_j's on I's chain of delegators is not accumulated in the blacklist. U aborts if its verification on $NMChainProof$ fails. Otherwise, I Redes these delegating keys, randomizes $EQProof_j$ to match commitments in the new delegating keys and $Cred_U$, and adds a new delegating key De_I to prove that Rn_I is not revoked and a proof $EQProof_I$ that two commitments of Rn_I in $Nym_I[2]$ and De_I commit to the same value. The result $DeInf_U$ is sent to and verified by U.

- Revoke: Add Rn to the accumulated set and update the accumulator value.
- CredProve: Abort if $Nym \neq (\mathsf{Com}(Sk, Open_{Sk}), \mathsf{Com}(Rn, Open_{Rn}))$, or $\mathsf{PKVerify}(Para_{PK}, (Nym_0, (\mathsf{Com}(Sk, 0), \mathsf{Com}(Rn, 0))), Cred)$ rejects, or verifying $DeInf$ fails. Otherwise, use ProveNM to generate a proof $NMProof$ that Rn is not blacklisted. Generate $EQProof'_L$ that Rn's commitments in $NMProof$ and in $Nym[2]$ both commit to the same value. Randomize $Cred$ to get a proof which contains Nym. Concatenate this proof with $NMProof$ and $EQProof'_L$ to get $CredProof'$. If the optional $DeInf$ is omitted, just output $CredProof'$.

  Otherwise, use CompNMProof to generate a proof $NMChainProof$ that each Rn_j's on the user's chain of delegators is not accumulated in the blacklist. For $j \in \{1, ..., L-1\}$, update and randomize $EQProof_j$ of $DeInf$ to get $EQProof'_j$ which proves Rn_j's commitments in $NMChainProof$ and $CredProof'$ both commit to the same value. Concatenate $NMChainProof$, $CredProof'$ and $EQProof'_j$ for $j \in \{1, ..., L-1\}$ to output $CredProof$ as described in (1).
- CredVerify runs PKVerify on the randomization of $Cred$, VerifyNM on $NMProof$ and

  $NMChainProof$, and verifies $EQProof'_j$ for $j \in \{1, ..., L\}$ to output accept or reject.

Theorem 4. *If the authentication scheme is F-unforgeable and certification-secure; the ADNMP is secure; CredPS, NMPS and EQPS are randomizable and composable ZK; CredPS is also partially extractable; and H is collision resistant, then this construction is a secure revocable delegatable anonymous credential system.*

Proof of theorem 4 is given in Tech Report [29]. Instantiation of the building blocks are given in Tech Report [29]. Briefly, a secure ADNMP is presented in Section 5; the BCCKLS building blocks can be instantiated as in [1]; and an EQPS can be constructed from [12,1].

References

1. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
2. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993: 1st Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, pp. 62–73. ACM Press, New York (1993)
4. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
5. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
6. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
7. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
8. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
9. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
10. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 295–308. Springer, Heidelberg (2009)
11. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
12. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
13. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, Washington D.C., USA, October 25-29, pp. 132–145. ACM Press, New York (2004)
14. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: Atluri, V. (ed.) ACM CCS 2002, Washington D.C., USA, November 18-22, pp. 21–30. ACM Press, New York (2002)
15. Microsoft: U-prove community technology preview (2010), `https://connect.microsoft.com/`
16. Bichsel, P., Camenisch, J., Groß, T., Shoup, V.: Anonymous credentials on a standard java card. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 2009, Chicago, Illinois, USA, November 9-13, pp. 600–610. ACM Press, New York (2009)
17. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: PEREA: towards practical TTP-free revocation in anonymous authentication. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS 2008, Alexandria, Virginia, USA, October 27-31, pp. 333–344. ACM Press, New York (2008)
18. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)

19. Ateniese, G., Kamara, S., Katz, J.: Proofs of storage from homomorphic identification protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 319–333. Springer, Heidelberg (2009)
20. Charles, D., Jain, K., Lauter, K.: Signatures for network coding. International Journal on Information and Coding Theory (2006)
21. Yun, A., Cheon, J., Kim, Y.: On homomorphic signatures for network coding. Transactions on Computer (2009)
22. Johnson, R., Walsh, L., Lamb, M.: Homomorphic signatures for digital photographs. Suny Stony Brook (2008)
23. Monnerat, J., Vaudenay, S.: Generic homomorphic undeniable signatures. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 354–371. Springer, Heidelberg (2004)
24. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002)
25. Fouque, P.-A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
26. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC Annual ACM Symposium on Theory of Computing, Bethesda, Maryland, USA, May 17-20, pp. 169–178. ACM Press, New York (2009)
27. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Cryptography against continuous memory attacks (2010)
28. Ghadafi, E., Smart, N.P., Warinschi, B.: Groth sahai proofs revisited. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 177–192. Springer, Heidelberg (2010)
29. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. Technical Report MSR-TR-2010-170, Microsoft Research, One Microsoft Way, Redmond, WA 98052 (December 2010)
30. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
31. Boyen, X.: The uber-assumption family (invited talk). In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008)
32. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)