Secure Multi-party Computation-based Privacy-Preserving Authentication for Smart Cities

Victor Sucasas*, Abdelrahaman Aly*†, Georgios Mantas§, Jonathan Rodriguez¶, Najwa Aaraj*‡
*Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
†imec-COSIC, KU Leuven, Leuven, Belgium
§University of Greenwich, Chatham Maritime ME4 4TB, UK
¶University of South Wales, Pontypridd CF37 1DL, UK
‡Okinawa Institute of Science and Technology, Okinawa, Japan

***Abstract*—The increasing concern for identity confidentiality in the Smart City scenario has fostered research on privacy-preserving authentication based on pseudonymization. Pseudonym systems enable citizens to generate pseudo-identities and establish unlinkable anonymous accounts in cloud service providers. The citizen's identity is concealed, and his/her different anonymous accounts cannot be linked to each other. Unfortunately, current pseudonym systems require a trusted certification authority (CA) to issue the cryptographic components (e.g. credentials, secret keys, or pseudonyms) to citizens. This CA, generally a Smart City governmental entity, has the capability to grant or revoke privacy rights at will, hence posing a serious threat in case of corruption. Additionally, if the pseudonym system enables de-anonymization of misusers, a corrupted CA can jeopardize the citizens' privacy. This paper presents a novel approach to construct a pseudonym system without a trusted issuer. The CA is emulated by a set of Smart City service providers by means of secure multi-party computation (MPC), which circumvents the requirement of assuming an honest CA. The paper provides a full description of the system, which integrates an MPC protocol and a pseudonym-based signature scheme. The system has been implemented and tested.**

I. Introduction

The Smart City paradigm evolves from the digitization of city management and governance processes. Urban subsystems such as energy, water supply, lighting or traffic monitoring will move to the cloud [1],[2],[3]. But in parallel, and supported by the Smart City infrastructure, novel cloud-based citizen-centric applications will also emerge, enabling the online management of public and private services. Online applications such as public parking management, intelligent transportation, digital libraries, augmented reality, leisure events management or location-based services [4],[5],[6],[7] will enrich the user experience. Also, user-centric distributed sensing applications [8] or eHealth [9] will be supported by the Smart City ecosystem. However, citizen-centric applications will also entail a privacy concern. Citizens are required to create user accounts in different cloud service providers (CSPs), potentially linked to their real identity. Thus, CSPs will be able to trace citizens' accounts and perform user profiling, which can discourage citizens from adopting these services [10].

In this framework, privacy-preserving authentication mechanisms have been proposed to support identity confidentiality and prevent user profiling [11],[12],[13]. Specifically, pseudonym systems enable users to create and maintain long-lasting anonymous accounts by holding an unlikable pseudo-identity (i.e. pseudonym) with each CSP [14],[15],[16]. Pseudonym systems do not pose a trade-off between users' privacy and CSP security since it is possible to control the users' pseudonym generation capabilities to prevent sybil attacks (i.e. users generating several accounts in the same service) [17],[16],[15]. Also, some systems cater for revocation mechanisms to enable de-anonymization of misusers [18],[19]. Unfortunately, current pseudonym systems have a common limitation. They need a trusted certification authority (CA) to issue the cryptographic components to users, e.g. credentials or private keys. The CA is the only entity able to choose who is granted privacy rights, and also responsible to enforce de-anonymization of misusers. CSPs and citizens do not have control over that. Thus, current systems require a leap of faith in a CA, from both the CSPs and the users, which also adds up to the operational cost and infrastructure complexity.

In this paper we propose a pseudonym system where the CA is not required. The role of the CA is adopted by a group of CSPs by means of secure multi-party computation (MPC) [20],[21],[22],[23]. CSPs can securely generate credentials for citizens and revoke misuers' privileges in a distributive fashion. All CSPs are required to grant credentials to users or revoke privacy privileges, thus ensuring that no third party can issue credentials to illegitimate users or de-anonymize honest users. Only one CSP is required to be honest, to ensure that no information about the users' pseudonyms is leaked. Additionally, a user (i.e. citizen) can self-generate an unlimited number of unlinkable pseudonyms without contacting the distributed CA. Any CSP can individually verify the pseudonym validity without linking pseudonyms to users' identities. Thus, citizens can generate long-lasting anonymous accounts using pseudonyms, without relying on a trusted entity.

The paper is structured as follows: i) section II describes related work; ii) section III provides the system model, threat model and system protocols; iii) section IV contains the preliminaries; iv) section V discusses the adaptations to Overdrive [21], a well-known MPC protocol adopted in our system; v) section VI details the pseudonym-based signature scheme used in our system, a modification of our previous scheme [15]; vi) section VII describes the distributive protocols for credential management; vii) section VIII provides a protocol to expand the cloud service set; viii) section IX provides the security analysis; ix) section X details the performance evaluation; x) section XI caters for a detailed comparison with similar approaches; and xi) section XII concludes this paper.

II. Related Work

A. Pseudonym-based authentication systems

Pseudonym-based systems enable users to create long lasting anonymous sessions with CSPs. A user can create an account linked to a pseudonym and access that account at any time by proving the possession of a secret value or credential associated with the pseudonym. There are several primitives that can be leveraged to implement a pseudonym-based system. In some scenarios, some basic and practical techniques can be applied [24], namely random number generators, counters, authentication codes or symmetric key encryption. For more complex scenarios, like the Smart City ecosystem previously described, more advanced techniques are recommended, such as: i) anonymous credential systems; ii) group/ring signature schemes; and iii) public-key or identity-based cryptosystems.

1) public-key and identity-based cryptosystems: A Pubic Key-based solution for pseudonymity has been proposed in scenarios that require low latency, e.g. vehicular networks (included in current standard for VANETs [25]). It is also simpler when it comes to implementing revocation mechanisms. Basically, it consists on granting users a batch of public key certificates representing their pseudonyms [26],[27],[28]. Pseudonym traceability is avoided by refreshing public keys periodically [29], [30]. However, this approach requires a CA to generate and distribute, and periodically renew, a large number of public key certificates. These certificates must also be transmitted together with the signed messages. Identity-based cryptography (IBC) [31] can be adopted following the same approach as public key certificates. IBC considers the user's identity as the public key, hence users can sign messages, whose signature can be verified using the corresponding identities. However, if the identity of the user is replaced by a pseudonym, then the user can sign messages with that pseudonym [32]. In this scenario, conditional privacy (i.e. de-anonymization in case of misbehavior) is still possible when the trusted CA issues both the secret keys and the pseudonyms [33]. Similarly to the previous solutions, a trusted CA is still a requirement since a corrupted CA can trace or impersonate users.

2) anonymous credential systems: Anonymous credential (AC) systems [34][35],[36] are normally composed of two processes: an issuing process and a showing process. The former consists of a blind signature from a Certification Authority (CA) on a user's committed value, which composes the user's credential. The latter is a mechanism enabling the credential holder to prove, in zero-knowledge, the possession of such a credential. ACs cater for constructions that enable pseudonymity in different ways. Pseudonyms can represent long-lasting relationships between the user and different organizations [37], or be constructed in the form of short-lived tokens that can be optionally linkable (i.e. enabling both long lasting or short anonymous sessions) [17],[19]. Also, revocation mechanisms can be implemented in AC systems [38], but without de-anonymization; i.e., revoked pseudonyms get rejected during the showing process without identifying the user. Some systems adopt homomorphic encryption instead of a blind signature scheme in the credential issuing process [14],[15], and cater for constructions that enable anonymous, yet auditable, centralized pseudonym management (the users can track how their pseudonyms are generated and linked by service providers).

There are efficient implementations, such as IDEMIX [39] or U-Prove [40] from IBM and Microsoft respectively. However, almost all systems based on digital anonymous credentials require authorities, either centralized or federated (i.e. one authority or several authorities issuing credential components in the form of attributes). Revocation is also managed by these authorities independently. Hence, even if credentials (or attributes) are issued anonymously (thus credential holders' identity is not jeopardized), the credential generation process and the revocation can be triggered illicitly by dishonest authorities. It is worth mentioning that there are constructions with a fully decentralized credential generation [41]. This approach leverages a distributed ledger. Users can directly create their own credentials. The issuing process involves the user including a commitment in an accumulated value placed in the public ledger. Then, during the showing process users can prove possession of a valid credential linked to the accumulated value. This approach however cannot be applied in the Smart City scenario since the citizens themselves should create their own credentials and verify each credential issuing process. Generally, the main advantage of ACs over other techniques is the impossibility for corrupted CAs to de-annonymize users. However, this is also their main limitation in the Smart City scenario, since it impedes de-anonymization of misusers.

3) group signature schemes: Group signature (GS) schemes, similar to digital anonymous credentials, allow users to remain unidentifiable under the anonymity set formed by the group members. Users are granted a private key that allows them to generate signatures on the group's behalf [42],[34]. The main advantage of GS schemes over AC systems is the possibility to de-anonymize misusers (i.e., dishonest users). AC systems can revoke credentials (e.g. by including them in a revocation list) but cannot de-anonymize misusers. However, this also poses a higher requirement of trust in the CA side. Anonymity revocation can come in two flavours. Some systems include the user's identity encrypted in the signature, which can be decrypted by the authority in case of misbehaviour. Others require the CA to check membership of a signature with respect to all stored credentials. The first approach caters for fast revocation but requires longer signatures, whereas the second approach enables shorter signatures at the cost of increasing the complexity of the revocation mechanism.

GS-based systems can integrate pseudonyms, they are embedded as part of the signature verification key [43],[44]. Although frequently these schemes only allow one pseudonym at a time, and the pseudonym change happens in a time-slot basis, some schemes allow unlimited number of pseudonyms per user [16]. Unlimited pseudonymity improves the system scalability since users need only one credential to access numerous services with different pseudo-identities. However, unlimited pseudonymity also allows a user to access the same service with different pseudo-identities, which may be deleterious in some scenarios, i.e. a sybil attack. Other works propose pseudonym-based signature schemes with indexed pseudonyms as a manner to control the users' pseudonym self-generation capabilities [15],[45]. In this paper, we adopt our proposed solution in [15], since it supports de-anonymization

of misusers by the CA and pseudonym self-generation controlled by verifiers (i.e. the CSPs), hence it is accountable, scalable and prevents sybil attacks. This solution still requires a trusted CA, but we eliminate this requirement by means of MPC.

4) Distributed GS and ACs: Extending previous literature on ACs and GS, it is worth highlighting the latest research on distributed versions of group signatures DGS [46][47] and anonymous credentials DACs [48]. These are the most similar works to our proposed system in terms of functionality, since they cater for distributed credential issuance, and in the case of [46][47] also distributed anonymity revocation. Although these works do not aim at providing pseudonym-based authentication, they could be potentially extended to fulfil that aim. Hence, we dedicate a specific section (Sec. XI) to describe the differences with respect to our proposed system.

B. Privacy-preserving authentication in Smart Cities

Privacy-preserving authentication (PPA) constitutes a promising approach to provide authentication, enhanced with privacy preservation features [49],[10]. However, the majority of existing PPA protocols for Smart Cities in literature focus on anonymity against external adversaries, i.e. they cater for solutions that protect the user identity from eavesdroppers while ensuring identification from service providers.

The authors, in [50], presented a biometrics-based authentication scheme for multi-server environment, using elliptic curve cryptography (ECC), to address the weaknesses in [51] and [52] at the cost of increasing communication and computational cost slightly. However, the authors in [53] analyzed the authentication scheme proposed in [50] and demonstrated that it is vulnerable to a known session-specific temporary information attack and impersonation attack. Nevertheless, both schemes in [50] and [53] cannot be considered as suitable for mobile services as they require a Registration Center to be constantly online to execute mutual authentication [49]. Therefore, the authors in [12] presented a PPA scheme for distributed mobile cloud computing services, without the need for an online Registration Center. Specifically, their scheme requires only a single private key. Nonetheless, in [13] authors showed that it cannot resist the service provider impersonation attack and thus, an adversary can impersonate the service provider to the user, i.e. it presents flaws for mutual-authentication. To solve these issues, and also to reduce communication costs, authors in [13] proposed a new privacy-aware authentication scheme using an identity-based signature scheme. Finally, Wu et al., in [54], proposed a new identity-based anonymous authentication protocol, based on two party computation, designed to provide both secure key agreement and key protection for mobile authentication, which yields better security and efficiency for mobile Internet environment.

As previously mentioned, previous schemes support user anonymity and user untraceability against external adversaries, but the user gets fully identified and authenticated towards the service provider. Similarly, our previous works that embed pseudonymity into OAuth2.0 protocol flow [55],[56] aim at providing identity privacy against eavesdroppers and not against service providers. Real implementations enabling privacy against service providers, in the context of smart cities, could be constructed with the technologies described in previous section (sec. II-A), such as [57] and [15]. However, these solutions require trust anchors, i.e. the presence of a CA.

In this context, it is worth highlighting recent efforts on decentralized identification [58],[59], which aim to provide an ecosystem where users can obtain credentials from a decentralized community of issuers without relying on trust anchors. Some works propose MPC to prevent key escrows [60] or to enable distributive credential issuance [61]. The system in [61] supports privacy-preserving authentication, it provides a rich set of features including sybil-resistance (pseudonym generation is controlled by the committee), accountability (enables revocation of credentials and de-anonymization of misusers), and privacy (credential and pseudonym issuance is anonymous). Also, it supports authentication of user attributes. However, the system does not provide pseudonym unlinkability since any member of the issuing committee can link pseudonyms to the same credential holder. This is because the system is designed for a scenario where the committee members are assumed honest. Also, each pseudonym is generated by the committee through an MPC protocol. This is not suitable for the Smart City scenario described, since: i) CSPs acting as issuers are also verifiers, hence CSPs can link pseudonyms to the same credential and effectively trace anonymous users; and ii) each pseudonym must be generated through a complex MPC protocol (involving fuzzy matching [62],[63]), hence it is not scalable in a Smart City scenario where users require many pseudonyms or periodic pseudonym switching.

III. System model and Threat model

This section defines the different entities present in the system, the trust model and the algorithms and protocols executed by the different entities.

A. Entities and Functionalities

The proposed privacy-preserving authentication system, depicted in Fig. 1, considers the following Smart City entities with their respective functionality:

- **User**: a citizen provided with a valid credential that allows himself to: i) self-generate unlimited number of unlinkable pseudonyms. Each pseudonym is linked to a public index value and is unique for that index; ii) sign any number of arbitrary messages using the credential and any valid pseudonym. The user can generate or access an anonymous account by presenting a valid signature and pseudonym.
- **Verifier**: an Authentication Server (AS) on the CSP side. It is responsible for checking the validity of signatures and pseudonyms.
- **Emulated CA (ECA)**: a CA emulated by a set of CSPs by means of MPC. It is in charge of: i) generating the verification keys of the signature scheme ii) issuing credentials to registered users (one credential per user); and ii) revoking the privacy rights of misusers, this involves: a) retrieving the real identity; b) including one or several pseudonyms of a revoked user in a revocation list.

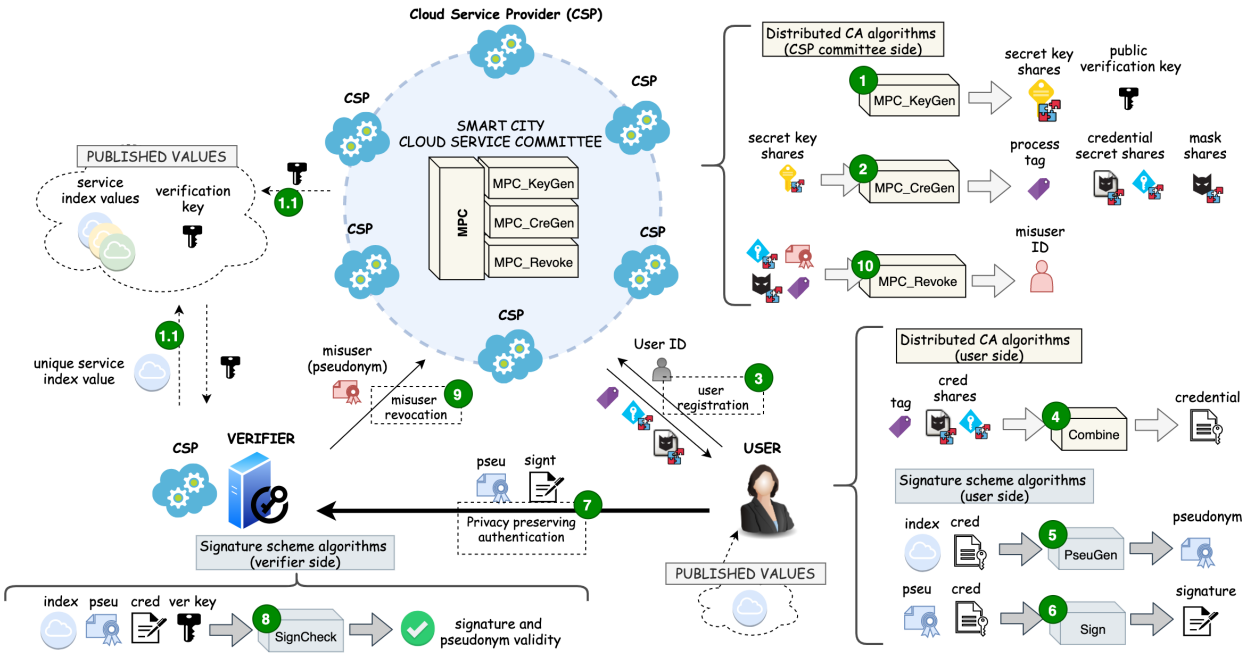

Fig. 1. System model for the proposed distributed credential management scenario. The CSP in the verifier side can also be part of the Smart City Cloud Service Committee, it has been located outside intentionally for the sake of clarify.

- **CSP**: Is a cloud service provider for a specific smart city service. It is equipped with a verifier and it can be part of the ECA. Users with valid pseudonyms can create anonymous accounts in the CSP. The CSP must publish a unique service index which is a numerical value used to construct the pseudonyms. If the service allows pseudonym switching, then the CSP can publish several indexes (e.g. crowdsensing [15]).

ECA brings significant benefits to Smart Cities as it is the key component for a more scalable privacy-preserving authentication approach by integrating a set of different CSPs, under the Smart City governance, to provide different Smart City services to registered citizens. Therefore, ECA addresses the scalability challenge of having one independent privacy-preserving authentication system for each Smart City service in order to enable identity confidentiality for citizens.

B. Threat Model

The Users, are considered *dishonest* and can perform several misbehaving actions. They may try to:

- Generate more than one pseudonym per specific service, i.e., the user may try to access services anonymously pretending to be several citizens, i.e. perform a sybil attack (USER 3 in Fig. 2).
- Use their anonymous state to misbehave and break the terms and conditions of the service. This is only effective if the anonymous state cannot be revoked by the service provider.

The CSPs, acting as verifiers and also participating in the ECA, are considered *dishonest*. CSPs may try to:

- Link pseudonyms to users' identities to effectively de-anonymize honest users (attack on USER 1 by CSPs 1 to 5 in Fig. 2).
- Link several pseudonyms to the same credential, i.e. to link pseudonyms of the same anonymous user in different services (attack on USER 1 by CSPs 1 to 5 in Fig. 2). This can be used to profile user activities and infer his/her identity.
- Extract the credential of the user. This will enable the CSP to trace or impersonate the user (attack on USER 1 by CSPs 1 to 5 in Fig. 2).
- Generate credentials for illegitimate users. This will enable a CSP to collude with rogue users to illegitimately grant them access to other cloud services (USER 2 in collusion with CSPs 1 to 5 in Fig. 2).

It is worth highlighting that CSPs can have both roles simultaneously: i) be part of the ECA; and ii) adopt the role of verifier. For completeness, in our threat model and security analysis (sec. IX) we consider a CSP with both roles. However, in a Smart City scenario a reduced number of CSPs could conform the ECA. This would imply that any specific CSP should trust at least one CSP conforming the ECA. The threat model is summarized in Fig. 2.

C. Protocols and Algorithms

This section describes the algorithms and protocols of the proposed pseudonym-based system with distributed credential management. The Client algorithms enable users to generate pseudonyms and pseudonym-based signatures, which can be used to generate anonymous accounts in CSPs. The Verifier algorithms enable CSPs to validate pseudonyms and signatures sent by users. The CA protocols enable the CSPs to generate credentials to users and de-anonymize misusers. The values are tagged as public value (meaning the value can be disclosed) and private value (the value must be kept secret by its holder). Values that are secret shared, i.e. represented with [[]] are inherently private.

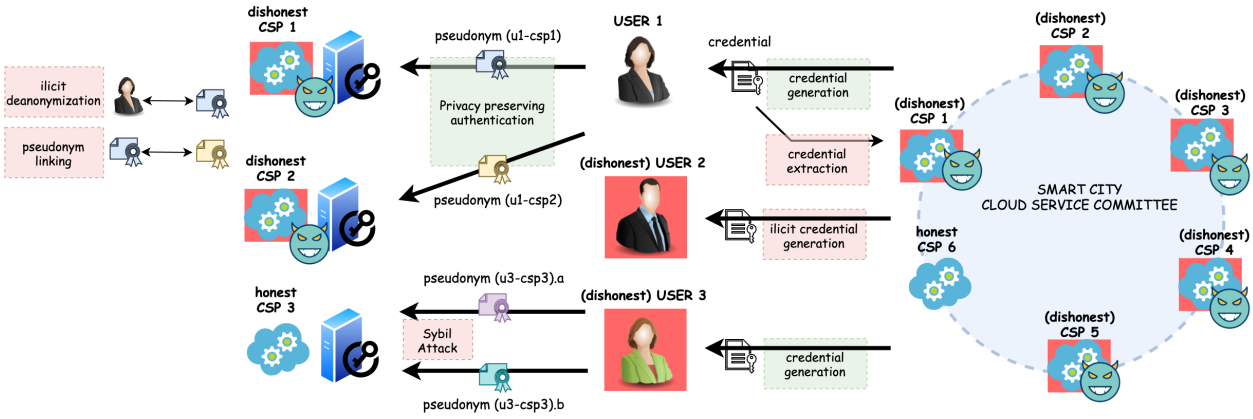

Fig. 2. Threat model. Note that CSPs can adopt both roles: i) verifier and ii) committee member. The figure separates both roles for the sake of clarity. The threat model considers that all committee members can be dishonest except one.

Client Algorithms:

- **PseuGen**: It takes as input the credential $cred$ (private value) and an index idx (public value). The idx is published by a CSP. The output is a pseudonym $pseu$ (public value) and a secret μ' (private value).
- **Sign**: It takes as input an arbitrary message m (public value), a pseudonym $pseu$, the credential $cred$, the secret value μ' (private value), and the idx value as input. The output is the signature σ (public value), which can be sent to a CSP to create an anonymous account or access an account previously created.

Verifier Algorithms:

- **SignCheck**: It takes as input the signature σ, the message m, the pseudonym $pseu$ and the index idx. The output is "valid" if the pseudonym and the signature were generated with a valid credential and with the correct index idx.

Emulated CA Protocols:

The following protocols are performed distributively by a set of CSPs to emulate the behaviour of the CA. The prefix **"MPC"** denotes that the protocol is performed by means of MPC by a set of CSPs in a distributive fashion:

- **MPC_KeyGen**: All CSPs take as input a security parameter k and a set of *public* parameters PP. The output is a *public* key W. Each party gets a secret share of the master secret key s, denoted as $[[s]]$.
- **MPC_CreGen**: It takes as input the secret shares of $[[s]]$ from all parties. The user gets a credential $cred$ composed of two values (μ, Su). All CSPs get shares of a random mask $[[r]]$, shares of the credential value μ, i.e. $[[\mu]]$. All CSPs also get the unique tag value tag (public value) identifying the credential generation process. Additionally, the CSPs will also learn the identity of the user, ID, which is stored in a database together with the computed values, i.e. (ID,tag,$[[\mu]]$,$[[r]]$).
- **MPC_Revoke**: It is an iterative protocol. Each iteration takes as input the revoked pseudonym $pseu$, and a tuple (ID,tag,$[[\mu]]$,$[[r]]$). The output in each iteration is a negative or positive match. In each iteration, the protocol takes a different tuple as input, but keeps the same pseudonym. It executes until a positive match is found. Optionally, the MPC_Revoke protocol can also output pseudonyms, different from the revoked pseudonym $pseu$, but associated to the same tuple (ID,tag,$[[\mu]]$,$[[r]]$) and hence to the same user.

IV. Preliminaries

A. Bilinear Maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order r, where the elliptic curve discrete logarithm problem (ECDLP) is hard. Let κ be a security parameter that defines the number of bits of r. Then, e is a bilinear map [31] in the groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, if it satisfies:

- Bilinearity: $\forall \alpha, \beta \in Z_r$, $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$, it holds that $e(\alpha G, \beta H) = e(G, H)^{\alpha\beta}$
- No-degeneracy: There is at least two elements G, H such that $e(G, H) \neq 1_{\mathbb{G}_T}$
- Complexity: It is possible to compute efficiently the bilinear map e

B. k-CAA and (n,k)-CAA Problems

Let $\mathbb{G}$ be a cyclic group of prime order, the k-traitors Collusion Attack Algorithm (k-CAA) [64], is defined as: given the set $\left(Sa_1 = \frac{1}{x+a_1}P, \ldots, Sa_k = \frac{1}{x+a_k}P\right)$ compute a value $Sa_u = \frac{1}{x+a_u}P$ different from the previous set of values. It is proven [64] that the k-CAA is only solvable if the k-weak Diffie-Hellman Algorithm (k-wDHA) exists. The k-wDHA is an algorithm that is able to compute $\frac{1}{x}P$ from k+1 values of the form $P, xP, x^2P, \ldots, x^kP$. It is also known that $1-wDHA$ is equivalent to DHA, and that $1-wDHA$ implies $k-wDHA$ for any value $k > 1$ [64].

In [16],[15] the k-CAA problem is extended to consider n examples $(n, k) - CAA$. This implies that, given a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, the set $\left(Sa_1 = \frac{1}{x+a_1}P, \ldots, Sa_k = \frac{1}{x+a_k}P\right)$, and n values $W_i =$

xP_i for some public P_i $i \in [1,n]$, and the set $\left(Sa_1 = \frac{1}{x+a_1}P, \ldots, Sa_k = \frac{1}{x+a_k}P\right)$, a value $Sa_u = \frac{1}{x+a_u}P$ is computed that is different from the previous set of values such that $e(P,P) = e(aP_i + W_i, Sa_u)$. In our proposed signature scheme, we limit n to 1, i.e. $(1,k) - CAA$, and consider an asymmetric pairing. Note that the only difference is that there is only one $W = sP \in \mathbb{G}_1$, whereas $Sa_u \in \mathbb{G}_2$. This does not invalidate the security considerations proposed in [64],[16],[15].

C. Commitment Schemes

A commitment scheme, Π_{commit}, allows an entity to commit to a value while keeping it secret. After publishing the commitment, the entity can prove that it has the secret value. Two properties must be fulfilled: the *hiding property* and the *binding property*. The former denotes that the commitment must reveal no information about the committed value. The latter must reveal that it is impossible for the committer to open the commitment with a different value than the one used to obtain the commitment. In this system, we require a bit commitment scheme:

- Bit commitment $\Pi_{commit,b}(x)$: given a hash function $H()$ and a secret value x, generate random bit string R and compute $\Pi_{commit,b}(x) = H(x|R)$.

D. MPC-Overdrive Protocol

An MPC protocol allows a set of entities to calculate any function over private inputs, i.e. no information about the inputs is disclosed except for what can be inferred from the output. Overdrive [21],[65] is a full-threshold MPC protocol, where all parties are required to participate in all secure computations. If only one party is honest, information leaks are prevented. Any party can input a value x, which is secret shared among all participants and denoted as $[[x]]$. Secret shared values can be linearly combined and multiplied. For each secret shared value $[[x]]$, each party P_i holds a tuple $(x^{(i)}, m(x)^{(i)})$, where $x^{(i)}$ is the secret share and $m(x)^{(i)}$ is the MAC share. The relations $x = \sum_{i=1}^{n} x^{(i)}$ and $x\Delta = \sum_{i=1}^{n} m(x)^{(i)}$ hold, where n is the number of parties and Δ is the MAC key. The key Δ is not known by any party, but all parties have a MAC key share $\delta^{(i)}$ such that $\sum_{i=1}^{n} \delta^{(i)} = \Delta$. The set of Overdrive's subprotocols is depicted below:

- $\Pi_{input}(x)$: called by a party P_j and accepted by all other parties P_i, $i \neq j$, the value x is secret shared amongst all parties, i.e. $[[x]]$
- $\Pi_{lcomb}([[x_1]],\ldots,[[x_l]], c_1,\ldots, c_l,c_{l+1})$: called by all parties, and all parties obtain a secret shared value $[[x]]$ s.t. $x = \sum_{i=1}^{l} c_i x_i + c_{l+1}$
- $\Pi_{mult}([[x_1]],[[x_2]])$: called by all parties, and all parties obtain a secret shared value $[[x]]$ s.t. $x = x_1 \cdot x_2$
- $\Pi_{open}([[x]])$: called by all parties, where each party reveals its secret share $x^{(i)}$ to all other parties. Hence, the secret can be reconstructed as $x = \sum_{i=1}^{n} x^{(i)}$. The validity of the opened value can be verified.

All secret shared values are in a prime field $\mathbb{F}_p$. No information is leaked about the secret shared values during the execution of any subprotocol Π_{input}, Π_{lcomb} or Π_{mult}. Only the $\Pi_{open}([[x]])$ discloses information, since it is used to reveal the value x. It is worth detailing how the $\Pi_{open}([[x]])$ protocol is implemented, since in this paper we need to extend this protocol with new functionalities to work over elliptic curve points. Precisely, to work over elliptic curve points, our construction requires that the prime field $\mathbb{F}_p$ is Z_r, where r is the prime order of the cyclic groups defined in sec. IV-A. The Π_{open} protocol is described in Fig. 3.

$\Pi_{open}([[x]])$: given a secret shared value $[[x]]$, reveal x and verify its correctness. The n parties, with value shares $(x^{(1)}, m(x)^{(1)}) \ldots (x^{(n)}, m(x)^{(n)})$, and MAC key shares $\delta^1 \ldots \delta^n$ perform:

- each party P_i reveals $x^{(i)}$
- all parties get $x = \sum_{i=1}^{n} x^{(i)}$
- each party P_i gets $\psi_i = m(x)^{(i)} - \delta^{(i)}x$
- each party P_i commits $\Pi_{commit,b}(\psi_i)$
- each party P_i opens the commitment to reveal ψ_i
- all parties verify that $\sum_{i=1}^{n} \psi_i = 0$

Output: If the last verification does not hold, the parties abort. Otherwise the value x is adopted as a valid opening of the secret shared $[[x]]$.

Fig. 3. Protocol Π_{open}

E. MPC-Overdrive Overhead

Given the interactive nature of MPC protocols, it has been observed that communications are the dominant factor i.e., Araki et al, refer to it as their bottleneck [66]. Indeed, the authors show how in even highly optimized protocols targeting the reduction of the communication cost, network latency can be up to 40% of the overall cost in high throughput networks. We note that in the case of SPDZ (online phase), the CPU and associated costs are negligible when compared to the communication latency [67]. Therefore, the overhead can be mainly measured in terms of communication rounds and the volume of data exchanged.

- **Rounds:** the number of times parties (i.e. CSPs) need to send information to each other. Rounds are bound by the ping time and the volume of data exchanged.
- **Volume:** is defined by the number of elements exchanged in a single round.

In multiparty protocols, such as the members of the SPDZ family, both items are affected by non-linearities. This is, any addition or subtraction does not add communication rounds, but multiplications and openings do. Hence, we detail the complexity of our protocols in these terms in section X-A.

Various authors have compared adversarial setups, constructions and frameworks for MPC, including SPDZ and `SCALE - MAMBA` [68], [69]. To the best of our knowledge, protocols on the SPDZ family are the fastest protocols to offer active security and dishonest majorities for arithmetic and arbitrary circuits. Table IV-E compares the protocols that have an open source implementation and are actively secure.

Protocol	Full Threshold	Arithmetic Circ.	Aribitrary Circ.
SPDZ [21]	✓	✓	✓
KRSW18 [70]	-	✓	✓
HSS17 [71]	✓	-	✓
KRSW18 [70]	-	✓	✓
TurboSPDZ [72]	✓	✓	-

TABLE I. ACTIVELY SECURE MPC PROTOCOLS WITH OPEN SOURCE

V. EXTENSION ON OVERDRIVE

As described in sec. IV-D, Overdrive caters for an Π_{open} subprotocol to reveal any secret shared value. However, our system requires the opening of a secret shared value as the logarithm of an elliptic curve point in a cyclic group $\mathbb{G}$. Hence, we extend the Overdrive protocol with the Π_{openEC} subprotocol, described in Fig. 4. In such subprotocol, for a secret share $[[x]]$, the parties disclose a point $W = xH$ where H is a public generator in a subgroup of EC points $\mathbb{G}$, with prime order r. Note that this is possible since $x \in Z_r$. This protocol follows a similar flow as Π_{open}, and its security is proven in sec. IX-C.

$\Pi_{openEC}([[x]], H)$: given a secret shared value $[[x]]$ and a generator H, reveal the EC point xH and verify its correctness. The n parties, with value shares $(x^{(1)}, m(x)^{(1)}) \dots (x^{(n)}, m(x)^{(n)})$, and MAC key shares $\delta^1 \dots \delta^n$, perform:

- each party P_i reveals $Q^{(i)} = x^{(i)}H$
- all parties get $Q = \sum_{i=1}^{n} Q^{(i)}$
- each party P_i gets $\Psi_i = m(x)^{(i)}H - \delta^{(i)}Q$
- each party P_i commits $\Pi_{commit,b}(\Psi_i)$
- each party P_i opens the commitment to reveal Ψ_i
- all parties verify that $\sum_{i=1}^{n} \Psi_i = 0$

Output: If the last verification does not hold, the parties abort. Otherwise the value Q is adopted as a valid opening of the secret shared $[[x]]$. The value x is not revealed.

Fig. 4. Protocol Π_{openEC}

VI. PSEUDONYM-BASED SIGNATURE SCHEME

This section describes the pseudonym-based signature scheme, composed of the KeyGen, CredGen, and Revoke algorithms. The scheme described in this section is a version of our previous work in [15], but adopting an asymmetric setting (specifically a BLS12-381 curve). Later sec. VII-B describes how to perform some of these algorithms in a cooperative fashion by means of MPC. The correctness and security of this asymmetric version is proven in sec. VI-G and sec. IX.

A. KeyGen

Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and Z_r be cyclic groups of prime order r as defined in sec. IV-B, and let e be a pairing s.t. $e() : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Also, let $H() : \{0,1\}^n \rightarrow Z_r$ be a hash function. The elements G, H and g are generators such that $\mathbb{G}_1 = \langle G \rangle$, $\mathbb{G}_2 = \langle H \rangle$ and $\mathbb{G}_T = \langle g \rangle$, where $g = e(G, H)$. Also, select another generator h, s.t. $h = g^z$ for an unknown z. Then select $s \xleftarrow{R} Z_r$ as the secret master key, and publish $W = sG$ as a public key. Section VII-A describes how to perform the KeyGen protocol distributively by means of an MPC protocol.

B. CredGen

An entity provided with the secret master key s can generate credentials by performing:

- select random $\mu \xleftarrow{R} Z_r$
- get $Su = \frac{1}{s+\mu}H$

The credential is the tuple (μ, Su) which is given to the user through a secure channel and must be kept secret. Several works have previously proposed credentials of this form for pseudonym generation [73],[14],[16],[15], and we refer interested readers to those works for a detailed description of pseudonym generation and signing algorithms. We adopt the approach of [15] since it enables pseudonym generation linked to index values, and also due to its simplicity when integrated with a MPC protocol. Section VII-B describes how this integration is performed.

C. PseuGen

To obtain a pseudonym for an index idx, any user with a valid credential (also to referred as credential holder) $(\mu, Su = \frac{1}{(\mu+s)}H)$ performs the following:

- get $d = H(idx)$
- set the secret value $\mu' = \frac{(d-\mu)}{2}$
- obtain $\tilde{P}u = \mu' Su$
- obtain $Pu = (\mu' + \mu)G$

The pseudonym is the tuple $(Pu, \tilde{P}u)$ and the associated secret value is μ'. The credential holder can generate unlimited unlinkable pseudonyms by selecting other values for idx. Two different pseudonyms cannot be generated with the same credential for the same idx value. The index value is a public value normally associated with the service that the user is accessing. Hence, it can be assumed public knowledge. We denote a pseudonym for a given index idx as $(Pu, \tilde{P}u)[idx]$[1].

D. Sign

A credential holder can use a pseudonym $(Pu, \hat{P}u)[idx]$, and the secret values Su, μ and μ', to sign a message. Given a message m of arbitrary length the signing algorithm is performed as follows:

- select random factors r_1, r_2, r_3, r_4, r_5, γ, $\delta \xleftarrow{R} Z_r^*$
- compute $T_{G_1} = r_1 G$
- compute $t_2 = [e(G, H + \hat{P}u)]^{r_2}$
- compute auxiliary public keys $\tilde{y}_1 = h^{\gamma} g^{\mu+\mu'}$ and $\tilde{y}_2 = h^{\delta} g^{\mu'}$
- compute $t_3 = h^{r_3} g^{-r_1}$, $t_4 = h^{r_4} g^{-r_2}$ and $t_5 = h^{r_5}$
- compute the challenge[2] $c = H_2(m||idx||\tilde{y}_1||\tilde{y}_2||T_{G_1}||t_2||t_3||t_4||t_5||Pu||\hat{P}u)$

[1] In the rest of this paper we omit the index and simply refer a pseudonym as $(Pu, \tilde{P}u)$ when the value of the index is not relevant

[2] where the operator $||$ represents concatenation.

- compute responses $s_1 = c(\mu+\mu')+r_1$, $s_2 = c\mu'+r_2$, $s_3 = -c\gamma+r_3$, $s_4 = -c\delta+r_4$, $s_5 = -c(\delta+\gamma)+r_5$
- The signature constitutes the auxiliary keys, the challenge and the responses, i.e the tuple $\sigma = (c, s_1, s_2, s_3, s_4, s_5, \tilde{y}_1, \tilde{y}_2)$

E. SignCheck

The algorithm runs at the verifier side, and checks the validity of a signature for a message m, with a pseudonym $(Pu, \hat{Pu})[idx]$. The verifier, provided with the public key W (published during KeyGen, sec. VI-A), performs the following operations:

- compute $d = H(idx)$
- compute $\bar{T}_{G_1} = s_1 G - cPu$
- compute $\bar{t}_2 = [e(G, H+\hat{Pu})]^{s_2}/e(Pu+W, \hat{Pu})^c$
- compute $\bar{t}_3 = h^{s_3} g^{-s_1} \tilde{y}_1^c$
- compute $\bar{t}_4 = h^{s_4} g^{-s_2} \tilde{y}_2^c$
- compute $\bar{t}_5 = h^{s_5}(\frac{\tilde{y}_1\tilde{y}_2}{g^d})^c$
- check whether the equality $c' = c$ holds, where: $c' = H(m||idx||\tilde{y}_1||\tilde{y}_2||\bar{T}_{G_1}||\bar{t}_2||\bar{t}_3||\bar{t}_4||\bar{t}_5||Pu||\hat{Pu})$
- the algorithm outputs "valid" if $c = c'$

F. Revoke

The Revoke algorithm revokes the privacy rights of a credential holder by linking a pseudonym to its owner. Ideally, a trusted authority should hold all users credentials and iteratively perform eq. 1 until a positive match is found. Specifically, a pseudonym $(Pu, \tilde{Pu})$ can be linked to a credential (μ, Su) by evaluating the following expression:

$$e(Pu+W, Su) = e(G, H+\tilde{Pu}) \tag{1}$$

G. Correctness

The signature scheme is correct, if and only if, for all signatures σ of a message m, generated by a *Sign* algorithm, using a valid pseudonym and valid credential, the output of SignCheck algorithm is always "valid". Although the correctness for this signature construction is already given in [15], this section caters for the correctness analysis for the case of an asymmetric pairing.

Given a valid signature $\sigma = (c, s_1, s_2, s_3, s_4, s_5, \tilde{y}_1, \tilde{y}_2)$ of a message m, with a pseudonym $(Pu, \hat{Pu})$, obtained from a credential (μ, Su), with associated secret value μ', and with a set of public parameters $(\mathbb{G}_1, \mathbb{G}_T, G, H, g, h, W)$, then $c = c'$ holds, i.e. the following relations must hold: i) $T'_{G_1} = T_{G_1}$; ii) $t_2 = t'_2$; iii) $t_3 = t'_3$; iv) $t_4 = t'_4$; and v) $t_5 = t'_5$. These relations always hold if the credential and pseudonyms are well constructed:

$$\begin{aligned} \bar{T}_{G_1} = s_1 G - cPu = \\ (c(\mu+\mu')+r_1)G - c(\mu+\mu')G = r_1 G = T_{G_1} \end{aligned} \tag{2}$$

$$\begin{aligned} \bar{t}_2 = \frac{[e(G, H+\hat{Pu})]^{s_2}}{e(Pu+W, \hat{Pu})^c} = \frac{[e(G,H)e(G,\hat{Pu})]^{s_2}}{e((\mu+\mu'+s)G, \frac{\mu'}{\mu+s}H)^c} = \\ \frac{[e(G,H)e(G,\hat{Pu})]^{s_2}}{[e((\mu+s)G, \frac{\mu'}{\mu+s}H)e(\mu' G, \frac{\mu'}{\mu+s}H)]^c} = \\ \frac{[e(G,H)e(G,\hat{Pu})]^{s_2}}{[e(G,H)e(G,\hat{Pu})]^{\mu' c}} = t_2 \end{aligned} \tag{3}$$

$$\begin{aligned} \bar{t}_3 = h^{s_3} g^{s_1} \tilde{y}_1^c = h^{-c\gamma+r_3} g^{-c(\mu+\mu')+r_1} (h^\gamma g^{(\mu+\mu')})^c = \\ h^{r_3} g^{-r_1} = t_3 \end{aligned} \tag{4}$$

$$\begin{aligned} \bar{t}_4 = h^{s_4} g^{s_2} \tilde{y}_2^c = h^{-c\delta+r_4} g^{-c(\mu')+r_2} (h^\delta g^{(\mu')})^c = \\ h^{r_4} g^{-r_2} = t_4 \end{aligned} \tag{5}$$

$$\begin{aligned} \bar{t}_5 = h^{s_5}[\frac{\tilde{y}_1\tilde{y}_2}{g^d}]^c = h^{-c(\delta+\gamma)+r_5}[\frac{h^\gamma g^{(\mu+\mu')} h^\delta g^{(\mu')}}{g^d}]^c = \\ h^{r_5} g^{(\mu+2\mu'-d)} = h^{r_5} = t_5 \end{aligned} \tag{6}$$

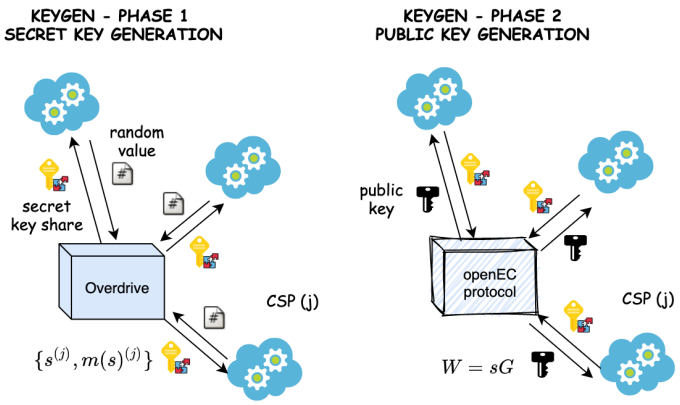

Fig. 5. MPC_KeyGen protocol.

VII. Distributed Credential Management

This section describes the distributive protocols **MPC_KeyGen**, **MPC_CredGen** and **MPC_Revoke** using Overdrive (sec. IV-D) with the extensions described in sec. V. These distributive protocols emulate their centralized version KeyGen, CredGen, and Revoke described in sec. VI. The following subsections detail the construction of the protocols and also give an intuition of how they work.

A. MPC_KeyGen

In the MPC_KeyGen protocol, depicted in Fig. 5, n parties $P = \{P_1, \ldots, P_n\}$ obtain a secret shared master secret key $[[s]]$ and a public key $W = sG$. To obtain $[[s]]$ the parties input random values s'_i. Then, with the authenticated shares of $[[s]]$, the parties can use the *openEC* extension to obtain $W = sG$. Each party P_i in the set $\{P_1, \ldots, P_n\}$ performs the following steps:

- each party P_i selects a random $s'_i \xleftarrow{\$} Z_r$

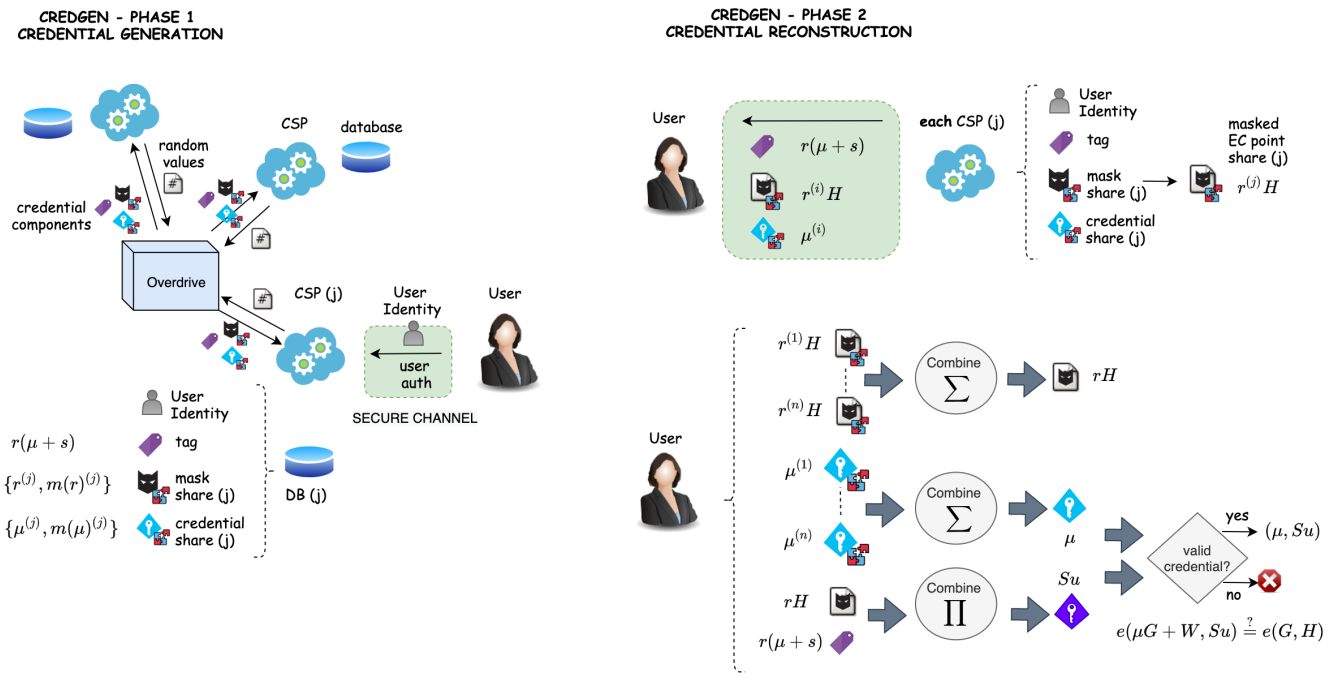

Fig. 6. MPC_CredGen protocol.

- each party P_i calls $\Pi_{input}(s'_i)$
- all parties call $\Pi_{lcomb}([[s'_1]],\ldots,[[s'_n]],1,\ldots,1)$ to obtain $[[s]]$ s.t. $s=\sum_{i=1}^{n} s'_i$
- all parties call $\Pi_{openEC}([[s]],G)$ to reveal $W=sG$

B. MPC_CredGen

In the MPC_CredGen protocol, depicted in Fig. 6, n parties $P=\{P_1,\ldots,P_n\}$ input their share of the master secret key $[[s]]$, and output shares of a credential $(\mu,\frac{1}{(s+\mu)}G)$, for some random μ. To give some intuition, the parties provide random inputs to select $[[\mu]]$ and combine it with $[[s]]$, to obtain $[[\mu+s]]$. This value is blinded with a random mask $[[r]]$, and revealed with the Π_{open} subprotocol to obtain $r(\mu+s)$ (learnt by the user and all parties). Then the parties disclose the shares $\mu^{(i)}$ and $r^{(i)}H$ to the user only. The user inverts $r(\mu+s)$ and multiplies by rH to obtain the second factor of the credential. In the whole process, the user cannot learn the master key s because the blinding mask r is never revealed. Similarly, the parties do not learn the credential because the values μ and rH are revealed to the user only. It is worth noting that the Π_{open} and Π_{openEC} subprotocols are not called to reveal μ and rH because their correctness is checked with eq. 7, hence the parties can directly send their shares to the credential holder.

Each party P_i performs steps the below:

- each party P_i selects a random mask $r_i \xleftarrow{\$} Z_p$ and random value $\mu_i \xleftarrow{\$} Z_p$
- each party P_i calls $\Pi_{input}(r_i)$ and $\Pi_{input}(\mu_i)$
- all parties call $\Pi_{lcomb}([[r_1]],\ldots,[[r_n]],1,\ldots,1)$ to get $[[r]]$ s.t. $r=\sum_{i=1}^{n} r_i$
- all parties call $\Pi_{lcomb}([[\mu_1]],\ldots,[[\mu_n]],1,\ldots,1)$ to get $[[\mu]]$ s.t. $\mu=\sum_{i=1}^{n}\mu_i$
- all parties call $\Pi_{lcomb}([[\mu]],[[s]],1,1)$ to get $[[\mu+s]]$
- all parties call $\Pi_{mult}([[\mu+s]],[[r]])$ to get $[[(\mu+s)\cdot r]]$
- all parties reveal their shares $\mu^{(i)}$ to the user. The user gets $\mu=\sum_{i=1}^{n}\mu^{(i)}$
- all parties use their share of $[[r]]$ to obtain $r^{(i)}H$, and send this value to the user. The user gets $rH=\sum_{i=1}^{n} r^{(i)}H$
- all parties call $\Pi_{open}([[(\mu+s)\cdot r]])$ to obtain $r(\mu+s)$ as the tag of the credential generation process. The value $tag=r(\mu+s)$ is sent to the user.

Finally, the user inverts $r(\mu+s)$ to get $\frac{1}{r(\mu+s)}$, and obtains $Su=\frac{1}{r(\mu+s)}rH$. The credential is the tuple (μ,Su), and its validity can be verified by evaluating whether the following equation holds:

$$e(\mu G+W,Su)=g \tag{7}$$

Before executing MPC_CredGen, the user should authenticate towards the CSPs and provide a real identity. Thus, the CSP can generate a tuple (ID,tag,$[[r]]$,$[[\mu]]$). This tuple is required for revocation and credential re-issuing.

C. MPC_Revoke

The MPC_Revoke links a pseudonym to a credential issuing process (MPC_CredGen) without disclosing the credential. Specifically, a pseudonym $(Pu,\tilde{Pu})$ is linked to a credential generation tuple (ID,tag,$[[r]]$,$[[\mu]]$). MPC_Revoke is an iterative protocol. In each iteration the parties input the same pseudonym to be revoked $(Pu,\tilde{Pu})$ and a different tuple (ID,tag,$[[r]]$,$[[\mu]]$). The result of each iteration is a negative or positive match. When a positive match is found, the protocol halts and outputs the value the identity of the user (i.e. the

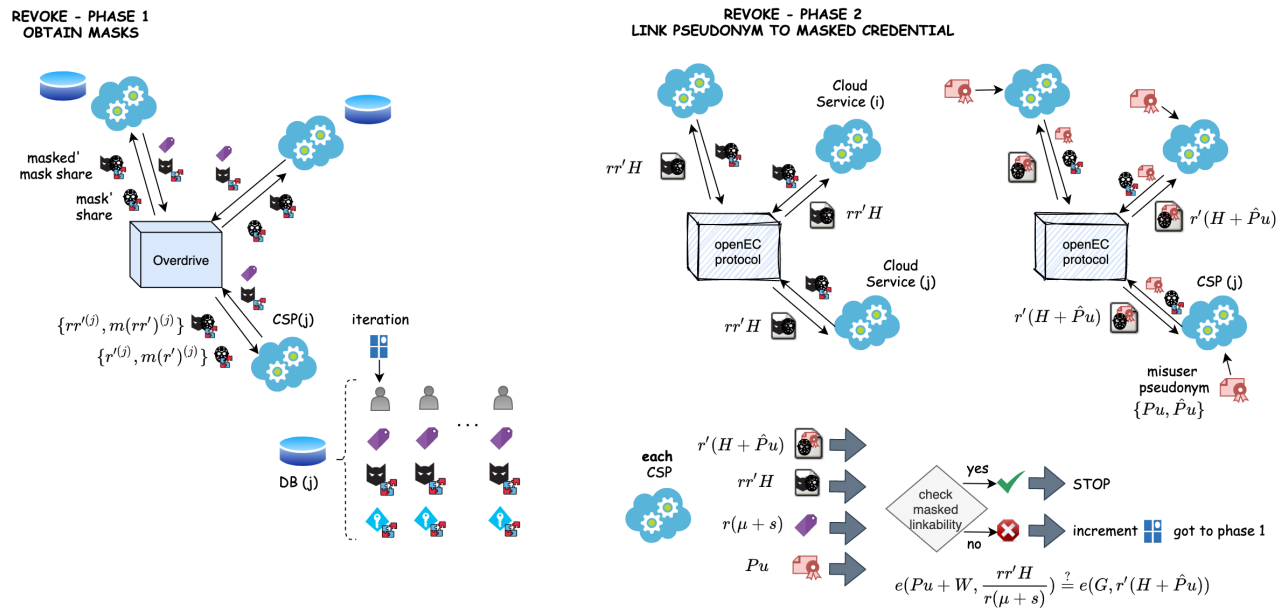

Fig. 7. A single iteration of the MPC_Revoke protocol.

value ID in the tuple). To give some intuition, this subprotocol performs the revocation check in eq. 1 in a blinded manner. The parties cannot open the value rH because the tag $r(\mu+s)$ is known by the parties, hence part of credential would be leaked. Instead, the parties obtain a new blinding mask r' and output $r'rH$. Additionally, they also output $r'(H+\tilde{P}u)$ in order to evaluate eq. 8 (blinded version of eq. 8).

Figure 7 depicts this protocol, which is also explained in what follows:

- each party P_i selects random value $r'_i \xleftarrow{\$} Z_r$
- each party P_i calls $\Pi_{input}(r'_i)$
- all parties call $\Pi_{lcomb}(r'_1,\ldots,r'_n,1,\ldots,1)$ to get $[[r']]$ s.t. $r' = \sum_{i=1}^{n} r'_i$
- all parties call $\Pi_{mult}([[r]],[[r']])$ to get $[[rr']]$
- all parties call $\Pi_{openEC}([[rr']],H)$ to get $R' = rr'H$
- all parties call $\Pi_{openEC}([[r']],H+\tilde{P}u)$ to get $Y = r'(H+\tilde{P}u)$

Finally, all parties can use values R' and Y to independently check whether a pseudonym $(Pu,\tilde{P}u)$ was generated with a credential associated with the tuple (ID,tag,$[[r]]$,$[[\mu]]$) by evaluating the following expression:

$$e(Pu+W,\frac{1}{(\mu+s)r}R') = e(G,Y) \tag{8}$$

Where W is the public key obtained during MPC_KeyGen. Note that:

$$\begin{aligned} e(Pu+W,\frac{1}{(s+\mu)r}R') = e\left(Pu+W,\frac{r'r}{(s+\mu)r}H\right) = \\ e((s+\mu)G,\frac{1}{(s+\mu)}H)^{r'}e(G,Su)^{\mu' r'} = \\ e(G,r'H)e(G,r'\tilde{P}u) = e(G,Y) \end{aligned} \tag{9}$$

At the end of the revoke protocol, all parties learn the identity of the user, i.e. the ID value, but the credential is not disclosed. It is worth highlighting that values $[[r']]$ and R' can be obtained before the Revoke protocol is called. Hence, this process can be pre-computed already during credential generation (MPC_CredGen). Only the protocol $\Pi_{openEC}([[r']],H+\tilde{P}u)$ and the check in eq. 8 should be executed when revocation is triggered, since it requires the revoked pseudonym as input. Namely, in Fig. 7, the Overdrive execution (phase 1) and the first OpenEC protocol in phase 2 are executed during credential generation after completion of MPC_CredGen.

Once a pseudonym has been revoked, it is possible to revoke more pseudonyms of the same misuser without disclosing the credential. This enables a fine-grained revocation capability since a misuser can be selectively excluded from some services, while keeping privacy in the rest. Let's assume that a pseudonym for an index idx has been revoked, i.e. the parties have obtained the tuple (ID,tag,$[[r]]$,$[[\mu]]$) associated with a pseudonym $(Pu,\tilde{P}u)[idx]$. To obtain another pseudnym $(Pu,\tilde{P}u)[idx']$ for which $idx \neq idx'$ the parties perform the following steps:

- all parties get $d = H(idx')$
- all parties perform $\Pi_{lcomb}([[\mu]],d,-\frac{1}{2},\frac{1}{2})$ to get $[[\nu]] = [[\frac{d-\mu}{2}]]$
- all parties perform $\Pi_{mult}([[\nu]],[[r]])$ to get $[[\rho]] = [[\frac{d-\mu}{2}r]]$

- all parties call $\Pi_{openEC}([[\rho]], H)$ to obtain $\frac{d-\mu}{2} rH$
- the parties invert $r(s+\mu)$ and obtain $\tilde{Pu}[idx'] = \frac{\frac{d-\mu}{2} r}{r(s+\mu)} H$

At the end of the execution, the parties obtain $\tilde{Pu}[idx']$, which uniquely identifies the pseudonym $(Pu, \tilde{Pu})[idx']$, and learns that such pseudonym belongs to the same credential holder as $(Pu, \tilde{Pu})[idx]$. The misusers' credential is not disclosed.

VIII. MPC SET EXPANSION PROTOCOL

Let us consider an scenario where the number of parties performing the computation needs to be increased. Let P' be the expanded set of parties of size $m = n + \delta$, such that $P \subset P'$. Parties in (the subset) P should be able to convert their additive shares of a secret shared value $[[x]]$ in P, to valid authenticated shares in P' under a new secret shared MAC key Δ'. We can achieve this, without loss of generality, by means of the protocol $\Pi_{input,exp}$ described in this section.

Naive algorithmic solutions for this scenario, such as resharing $[[x]]$ (i.e. resharing both x and $mac(x)$) from P to P', keeping the same MAC key Δ, violates our current security model. This is because a coalition of $|P'| - \delta$ parties could reconstruct the value Δ and tamper future computations (e.g. generation of new credentials). Protocol $\Pi_{input,exp}$ prevents this situation by translating secret values shared by parties in P, to values secret shared by parties in P' authenticated with a new Δ'. It is worth highlighting that after expanding the set, the parties in P would still be able to generate new credentials or revoke credentials generated before the expansion. This is because P has a qualified set of shares of the master key and previously issued credentials. However, the protocol $\Pi_{input,exp}$ ensures that all further computations occurring after the expansion to P' cannot be tampered by parties in P (e.g. revoking credentials issued by the new set).

To give some insights, we follow the principles introduced by [74], whereby P' is capable to provide some authenticated secret randomness r to P, a ephimeral MAC key r' and a public MAC $v = r \cdot r'$. Parties in P can then simply incorporate (and authenticate) the shares they receive as inputs, check the validity and correctness of r, via the MAC key r', and proceed to mask x, such that $y = x - r$. The protocol ends by broadcasting y to P', allowing its m parties to remove r, and hence obtaining authenticated shares of x. The protocol $\Pi_{input,exp}$ is defined below:

Parties in P': the expanded set of parties P', of size m, act as dealer and performs the following:

- sample authenticated random inputs $[[r]]$, $[[r']]$ where $r, r' \in \mathbb{F}_p$
- obtain $[[v]] = \Pi_{mult}([[r]], [[r']])$
- open the validation tag $v = \Pi_{open}([[v]])$
- every party i on P receives the reshares of $[[r]]$ as $[r_i]$ and the reshares of $[[r']]$ as $[r'_i]$ from the m parties in P', such that:

$$[r_i] = (r_1^i, \ldots, r_m^i),$$
$$[r'_i] = (r_1'^i, \ldots, r_m'^i).$$

- parties in P' also send v to parties in P.

Parties in P: the set of parties P compute inputs as indicated below:

- every party i in P reconstructs a share of r and r' as indicated below:

$$r_i = \sum_j^m r_j^i,$$
$$r'_i = \sum_j^m r_j'^i.$$

such that $r = \sum_i^n r_i$ and $r' = \sum_i^n r'_i$

- every party i in P calls $\Pi_{input}(r_i)$ and $\Pi_{input}(r'_i)$
- all parties get

$$[[r]] = \Pi_{lcomb}([[r_0]], \ldots, [[r_n]], 1 \ldots, 1)$$
$$[[r']] = \Pi_{lcomb}([[r'_0]], \ldots, [[r'_n]], 1 \ldots, 1)$$

- obtain $[[v]] = \Pi_{mult}([[r]], [[r']])$
- obtain $[[\sigma]] = \Pi_{lcomp}([[v]], -v)$
- validates $[[r]]$ if $\Pi_{open}([[\sigma]])$ is 0, and abort otherwise
- mask $[[x]]$ by substracting $[[r]]$:

$$[[y]] = \Pi_{lcomb}([[x]], [[r]], 1, -1)$$

- call $\Pi_{open}([[y]])$ to open and validate the masked value y
- parties in P send y to parties in P'.

Parties in P': perform the last step to obtain $[[x]]$ under the new MAC key Δ':

- Parties in P' obtain $[[x]]]$ by unmasking (note that $y = x - r$):

$$[[x]] = \Pi_{lcomb}([[r]], y)$$

If no party aborted during the protocol, then each party P'_i holds an authenticated share of x, now secret shared among m participants with a new MAC key Δ'. Transferring the state between MPC clouds presents a series of challenges when preserving authentication, i.e. crossed authentication and state tampering. This is the motivation behind sharing authenticated randomness from P' to P. Note that there is no means through which P can prove the correctness of r without Δ', else parties in P' could cheat, dealing invalid shares of r. Using a second random value r' as stated allows P to early abort, as well as, prevents any attempt of tampering x. Once masked, y can be validated and made public with the open subprotocol of the Overdrive protocol.

It is worth commenting that our protocol is capable of supporting cloud migration. Let P'' be the powerset (set of all

subsets) of P, then the protocol $\Pi_{input,exp}$, allows to transfer shares to any P' regardless $P_i'' \subset P'$, for any i in P''; allowing implementers to expand their use cases beyond what we have described in this section. On the other hand, the adoption of Overdrive as MPC protocol does not allow the reduction of parties in the set, since the protocol is full threshold (i.e. all parties are required to reconstruct secrets). However, such feature a would be possible if we adopt an MPC protocol based on a threshold secret sharing scheme [75],[76],[77], and adapt our sub-protocols (MPC_KeyGen, MPC_CredGen, and MPC_Revoke) and extensions (Π_{openEC}) accordingly.

IX. Security Analysis

This section provides the security analysis for the proposed system. First we provide a system security analysis addressing the potential attacks described in the threat model (sec. III-B). Then we describe the security proofs for the signature scheme and the Π_{openEC} extension. The security proof for the pseudonym-based signature scheme can be found in [15]. However, we have modified the signature scheme to be constructed over an asymmetric pairing, hence, we find relevant to present the proof in the asymmetric setting in sec. IX-B. The security of the MPC protocol (Overdrive) is detailed in [21]. In that paper, the security proof (actually a proof sketch) is in part based on the whole proof described in [78] for the Mascot protocol (previous version of the SPDZ protocol based on oblivious transfer). We have extended the protocol with a new Π_{openEC} subprotocol. In sec. IX-C, we prove that using elliptic curve points in the opening does not provide additional capabilities to a dishonest party, and hence its security still relies on the same assumptions as the open subprotocol from Overdrive and Mascot.

A. System Security Analysis

As described in sec III-B a rogue user may try to perform sybil attacks (pretend to be several citizens) or use a valid pseudonym that cannot be de-anonymized. For a sybil attack to be possible, the user should be able to generate two different valid pseudonyms for the same CSP index. This is proven impossible for this signature scheme in [15]. The adoption of an asymmetric paring does not invalidate the proof presented in [15]. Alternatively, the user could try to obtain more than one credential. This is avoided by performing a user authentication process during the credential issuing. Note that for each credential issued, the parties store the tuple (ID,tag,$[[r]]$,$[[\mu]]$). The ID of the user is stored in the clear, hence a double credential request from the same user would be detected. Finally, to prevent de-anonymization, a misuser may try to use a valid credential to forge a new credential not registered by the parties. This is not possible since the scheme is unforgeable, as proven in sec. IX-B.

Rogue CSPs can try to link pseudonyms to the same credential, but this is only possible if the credential is known [15]. The security of the MPC protocol ensures that no information leaks are possible if at least one party is honest. The MPC protocol is proven secure in [21] and the proposed protocol extension is proven secure in sec. IX-C. Thus, rogue CSPs with credential shares cannot reconstruct credentials or obtain any valuable information about them. This also prevents impersonation attacks. Similarly, the master key cannot be extracted if only one CSP is honest. Thus new credentials cannot be generated by a subset of rogue CSPs holding master key shares. Following the same reasoning, the de-anonymization process cannot be triggered by a subset of rogue CSPs.

B. Pseudonym-based Signature Security Analysis

The extended version of this proof for a symmetric setting can be found in [15]. In this section, we present a brief version for the asymmetric setting. Due to space constraints, we omit the details of the game description involving the queries from the adversary A (representing a user) to the challenger C (representing the credential issuer provided with a master key) described in [15]. This proof shows that if a user is able to forge a signature, then it is also able to solve the K-CAA problem, which is considered intractable.

Let's assume that the user (A) presents a signature $(c, s_1, s_2, s_3, s_4, s_5, \tilde{y}_1, \tilde{y}_2)$ that is valid for a pseudonym $(Pu, \hat{Pu})$ obtained with a credential that was never generated by the credential issuer (C). Note that the challenger C can verify whether the presented pseudonym is linked to a previously generated credential by evaluating the relation $e(Pu + W, Su) = e(G, H + \hat{Pu})$ (i.e. the Revoke algorithm). If the signature is valid, then for the Forking Lemma [79][3] the adversary can output another valid signature with the same commitments but different challenge $c \neq c'$, [16]. Hence, there is another valid signature σ' with commitments $T_{G_1} = T'_{G_1}$, $t_2 = t'_2$, $t_3 = t'_3$, $t_4 = t'_4$, $t_5 = t'_5$ and keys $(\tilde{y}_1, \tilde{y}_2)$, but with different challenge $c \neq c'$ and responses $s_1 \neq s'_1$, $s_2 \neq s'_2$, $s_3 \neq s'_3$, $s_4 \neq s'_4$, $s_5 \neq s'_5$.

If this is true, then A can solve the K-CAA problem:

$$\begin{aligned} &T_{G_1} = T'_{G_1} \Rightarrow s_1 G - cPu = s'_1 G - c' Pu \Rightarrow \\ &(s_1 - s'_1)G = (c - c')Pu \Rightarrow Pu = \tfrac{(s_1 - s'_1)}{(c - c')} G \end{aligned} \tag{8}$$

$$\begin{aligned} &t_2 = t'_2 \Rightarrow \tfrac{\left[e(G,H)e(G,\hat{Pu})\right]^{s_2}}{e(Pu+G,\hat{Pu})^{c}} = \tfrac{\left[e(G,H)e(G,\hat{Pu})\right]^{s'_2}}{e(Pu+W,\hat{Pu})^{c'}} \Rightarrow \\ &e(G,H)^{(s_2 - s'_2)/(c - c')} = e\left(Pu - \tfrac{s_2 - s'_2}{c - c'} G + W, \hat{Pu}\right) \Rightarrow \\ &e\left(\tfrac{(s_1 - s'_1) - (s_2 - s'_2)}{c - c'} G + sG, \tfrac{c - c'}{s_2 - s_2} \hat{Pu}\right) = e(G,H) \end{aligned} \tag{9}$$

The adversary A can find a solution $(a, \frac{1}{s+a} H)$ of the asymmetric (1,K)-CAA problem where $a = \frac{(s_1 - s'_1) - (s_2 - s'_2)}{c - c'}$ and $\frac{1}{s+a} H = \frac{c - c'}{s_2 - s_2} \hat{Pu}$. Hence, forging a signature is equally hard as solving the K-CAA problem in G_2.

Moreover, the adversary A provided with a valid credential should not be able to output two valid pseudonyms with the same index. This feature is proven in [15]. We do not include it here since the asymmetric pairing does not modify the steps of this proof. Also, it is left to be proven that pseudonyms and signatures are unlinkable, but such claim is straight-forward to prove since all elements in the signature are randomized [15].

[3] According to the Forking lemma, if an algorithm can yield an output, from some inputs obtained from a given distribution, and this output has some property with non-negligible probability, then the adversary has a non negligible probability of producing another output with the same property provided that the inputs are chosen from the same distribution.

C. OpenEC Subpprotocol Security Analysis

In the Overdrive's open subprotocol $\Pi_{open}([[x]])$, each party P_i in a set $\{P_1, \ldots, P_n\}$ discloses its share $x^{(i)}$ such that all parties obtain $x = \sum_i^n x^{(i)}$. At this step, any dishonest party can lie. In a second step, each party first commits the value $\psi_i = m(x)^{(i)} - x\delta^{(i)}$, where $\delta^{(i)}$ is the share of party P_i of an unknown key Δ, i.e. $\sum_i^N \delta^{(i)} = \Delta$, and $\sum_i^N m(x)^{(i)} = \Delta x$ (see sec. IV-D). Then, all parties open the commitments to reveal ψ_i. The value x is accepted as a valid opening of $[[x]]$ if the MAC check holds, this is if $\sum_i^n \psi_i = 0$. It is proven in [78] that breaking the security of this check is equally hard as extracting the key Δ from the encrypted $\delta^{(i)}$ shares. Although our proof follows different steps, in this section we prove that breaking the security of the proposed Π_{openEC} subprotocol also implies extracting Δ.

In the proposed $\Pi_{openEC}([[x]], G)$ subprotocol, the parties reveal their shares as logarithms of a common generator G. Without loss of generality and for simplicity of notation let's assume that only one party P_j is dishonest. All honest parties P_i for $i \neq j$ reveal $x^{(i)}G$ whereas P_j reveals another point X' for which it knows the logarithm with respect to G, i.e. $X' = x'G$. All parties obtain as the opened point $\sum_{i \neq j} x^{(i)}G + X'$. Then, in the MAC check, honest parties send the value:

$$\Psi_i = m(x)^{(i)}G - (\sum_{i \neq j} x^{(i)}G + X')\delta^{(i)}$$

To succeed the MAC check, the dishonest party has to submit a value Ψ_j such that $\sum_{i \neq j}(\Psi_i) + \Psi_j = 0$. However, if the dishonest party P_j is successful, then P_j can extract the key Δ. Note that:

$$\begin{aligned}\sum_{i \neq j}(\Psi_i) + \Psi_j &= \sum_{i \neq j}\left[m(x)^{(i)}G - (\sum_{i \neq j}(x^{(i)}) + x')\delta_i G\right] + \Psi_j \\ &= G\left[\sum_{i \neq j} m(x)^{(i)} - \sum_{i \neq j}(x + \tilde{x})\delta_i + \omega\right]\end{aligned} \tag{10}$$

In eq. 10, $\tilde{x}$ is the shift from the correct opened value introduced by party P_j and ω is logarithm of Ψ_j w.r.t. to G, i.e. $\Psi_j = \omega G$. Both vales can be considered known by P_j and chosen at will. With some mathematical manipulation eq. 10 can be expressed as:

$$\begin{aligned}G\left[\sum_{i \neq j}\left(m(x)^{(i)} - x\delta_i\right) + \tilde{x}\sum_{i \neq j}\delta_i + \omega\right] = \\ G\left[\sum_{i \neq j}\psi_i + \left(\tilde{x}\sum_{i \neq j}\delta^{(i)} + \omega\right)\right]\end{aligned} \tag{11}$$

It is straight forward to see that for $\sum_{i \neq j}(\Psi_i) + \Psi_j = 0$ to hold, then $\sum_{i \neq j}\psi_i + \left(\tilde{x}\sum_{i \neq j}\delta^{(i)}i + \omega\right) = 0$ must hold as well. Also note, that for the last relation to hold then we have that $\left(\tilde{x}\sum_{i \neq j}\delta^{(i)} + \omega\right) = \psi_j = m(x)^{(j)} - x\delta^{(j)}$. We also know, for the Forking Lemma, that if the party P_j is successful in the MAC check then it will be also successful in another opening with the same input x and different EC point $G \neq G'$, i.e. $\Pi_{OpenEC}([[x]], G')$. Following a similar reasoning, the dishonest party P_j could select different values $\tilde{x}_2$ and ω_2 such that $\left(\tilde{x}_2\sum_{i \neq j}\delta^{(i)} + \omega_2\right) = \psi_j = m(x)^{(j)} - x\delta^{(j)}$. However, this implies that P_j can obtain Δ:

$$\begin{aligned}\left(\tilde{x}\sum_{i \neq j}\delta^{(i)} + \omega\right) = \left(\tilde{x}_2\sum_{i \neq j}\delta^{(i)} + \omega_2\right) \implies \\ \sum_{i \neq j}\delta^{(i)} = \frac{\omega_2 - \omega}{\tilde{x} - \tilde{x}_2} \implies \Delta = \frac{\omega_2 - \omega}{\tilde{x} - \tilde{x}_2} + \delta^{(j)}\end{aligned} \tag{12}$$

Therefore, succeeding the MAC check subprotocol for the adversary is equally hard as extracting Δ, thus breaking the Π_{openEC} protocol is equally hard as breaking the original Π_{open} in [78].

X. Performance evaluation

The distributive protocols for credential management MPC_KeyGen, MPC_CredGen and MPC_Revoke (sec. VII) were implemented with Scale-Mamba framework [80], except for the proposed extension to the Overdrive protocol (sec. V) that was implemented with the C version of the MIRACL Core library [81]. The signature scheme (sec. VI) was also implemented with MIRACL Core. Specifically, we adopted the BLS12-381 curve. In this curve, G_1, G_2 and G_T are cyclic groups of prime order with elements of 48, 97 and 576 bytes respectively.

The distributive system for the emulated CA was deployed in 5 virtualized servers connected under VLANs. Each one has a static fixed memory allocation of 64 GB of RAM memory and 4 cores from an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GH chipset. In the user side, an Android application has been developed and tested in a Xiaomi Redmi Note 9 Pro. Also, a cloud service that acts as verifier and performs signature verifications has been implemented in a virtualized server with 4GB of RAM and an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz. Figure 8 shows the user interface in the CSP acting as verifier and the Android application in the user side.

Figure 9 shows the performance results in the user and verifier side. The user receives the credential components and performs: i) credential reconstruction; ii) credential validity check; iii) pseudonym generation; and iv) message signing. The CSP acting as verifier only performs the signature verification algorithm. The results are presented together with an execution of the same algorithms in a 2.3 GHz 8-Core Intel Core i9 with 16GB of RAM to provide a unified platform to compare the complexity of the different algorithms. The results show that the shift from a centralized credential generation approach in [15] to the distributive version proposed in this paper does not impact the performance in the user side. This is reflected by the low delay of the credential reconstruction step, which is the only step required in the user side due to

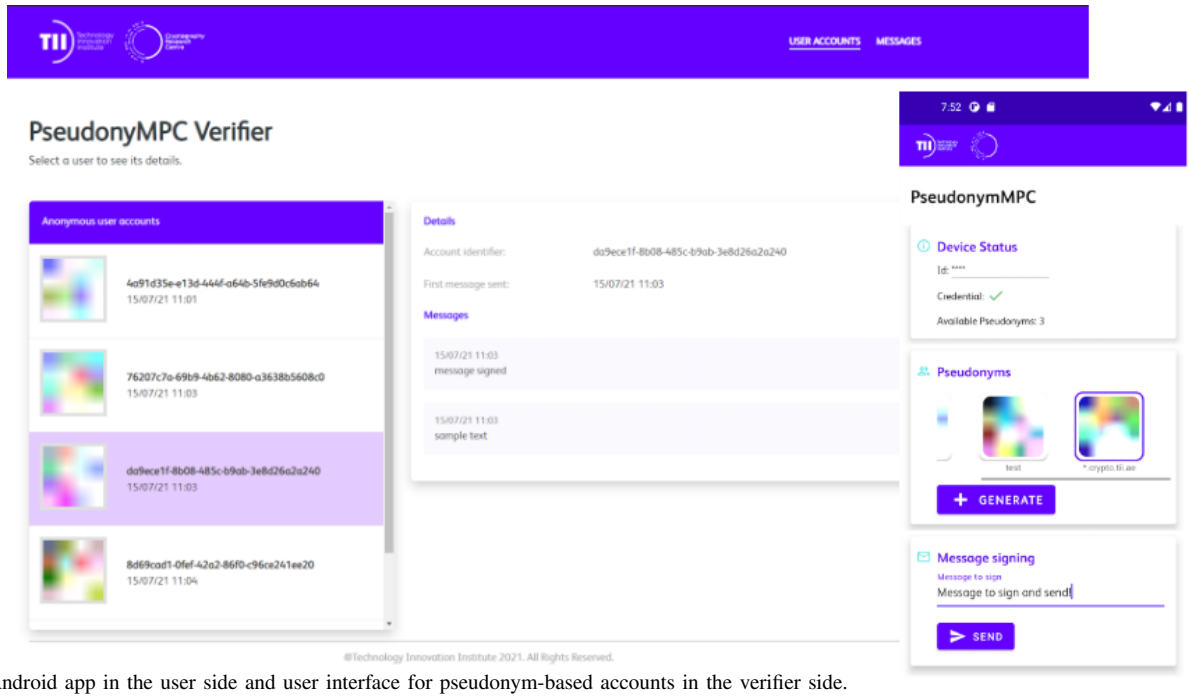

Fig. 8. Android app in the user side and user interface for pseudonym-based accounts in the verifier side.

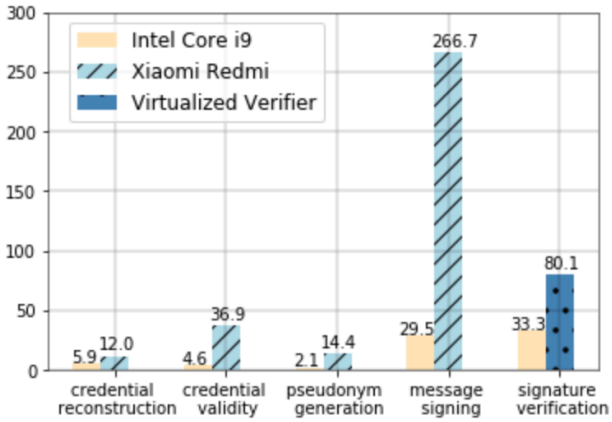

Fig. 9. Performance (ms) for the user and verifier algorithms.

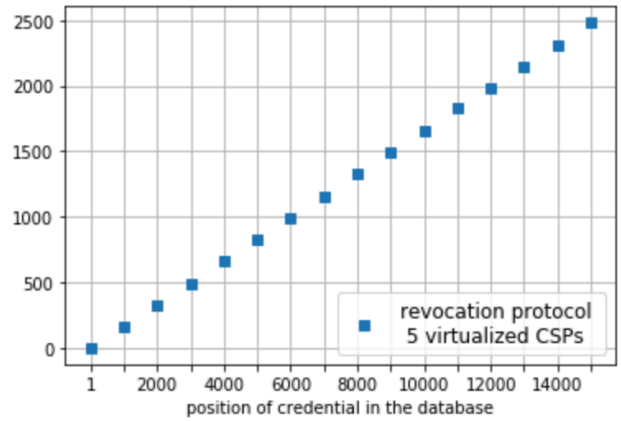

Fig. 10. Performance (s) for revocation of a credential depending on its position in the database.

the adoption of MPC for distributed credential issuance. It is also worth commenting that the user performs a credential validity check, so the user can detect dishonest parties sending wrong credential shares. The rest of the values in Figure 9 (i.e. pseudonym generation, message signing and signature verification) are faster than the previous scheme in [15]. This is because the distributed design does not affect the user side algorithms. Also, the new scheme adopts an asymmetric pairing instead of the symmetric version adopted in [15], hence this new version is more efficient[4][5].

The emualted CA perfoms credential generation and revocation protocols by means of MPC, specifically using Overdrive (sec IV-D) and our proposed extension (sec V). Overdrive is an MPC protocol in the pre-processing model, thus it consists of two phases: i) pre-processing phase where the parties' inputs are not yet defined and some cryptographic values are obtained; and ii) an online phase where the parties' inputs are defined and the private function is evaluated. The time complexity is mainly dominated by the pre-processing phase. The operations performed during the online phase, which are detailed in secs. VII-B and VII-C, are faster. Specifically, for credential generation our setup considers batches of 5000 credentials, and it takes 2h and 21min per batch considering pre-processing (1h and 42min) and online phase (39min) altogether. Additionally, the database update in each CSP takes 27min (timings of pre-processing, online phase and database update present a standard deviation of 31, 36 and 2018 seconds respectively).

Our implementation of the revocation protocol adopts the optimization described in sec. VII-C, i.e. the credential generation protocol precomputes some values required for revocation. Thus, during revocation only an OpenEC subprotocol

[4]comparison with real values was not presented since both schemes were implemented with different libraries MIRACL-Core (C++) and JPBC (Java)

[5]the scheme in [15] caters for a batch verification algorithm. Although it is not detailed in this paper, batch verification could be adopted in this asymmetric version

$\Pi_{openEC}([[r']], H+\tilde{P}u)$ and the check in eq. 8 are required per iteration. MPC_Revoke iterates over the stored tuples of secret-shared credentials until a match is found. It is worth noting that revocation time depends on the the number of users. This dependency is shown in the the performance evaluation results presented in Figure 10. Approximately, each credential check takes 160ms, hence for a database with 30K users revocation takes an average of 2500 sec[6]. It is worth clarifying that the complexity of the revocation process in the centralized version of the adopted signature scheme [15] also grows linearly with the number of credentials. This is a common case in previous pseudonym systems based on similar constructions [16][82][83] and other short group signature schemes [47].

A. Complexity Analysis

Let us consider the complexity of our 3 emulated CA Protocols that require MPC. From a theoretical perspective, SPDZ protocols require to send 2 elements to each party participating in the computation [67]. That implies a linear growth on the size of the CSPs. Non-sequential non-linearities can be then grouped in single rounds. Under those considerations we say that the number of elements exchanged is $\texttt{rounds} \cdot 2|CSP|$. We consider elements of up 128 bits.

- **MPC_KeyGen:**The protocol requires 1 round to share secret inputs $[[x]], [[x']]$ and 1 non-linearity. The protocol does not contain further non-linearites. A total of one round, thus round complexity $\mathcal{O}(1)$.
- **MPC_CreGen:** The protocol uses 1 round to secret share inputs and requires 2 non-linearities that are sequential. The round complexity of the protocol is also constant $\mathcal{O}(1)$, effective 3 rounds per credential (n).
- **MPC_Revoke:** As before, the protocol requires 1 round to secret share inputs, but it contains only 2 non-linearity. The total number of rounds is 3, raising again its complexity to $\mathcal{O}(1)$.

We refer the reader to Section X for a detailed performance revision. As previously mentioned, communication latency (ping time) has an important impact on overall performance of the underlying MPC Protocol. In the case of SPDZ, and given the limited number of rounds, ping time becomes the dominating factor. Indeed, regardless the setup, LAN 2-3 ms. or WAN 10+ ms, the CPU operations in our protocols remain constant [67], and the overall computation cost can be calculated as $\texttt{ping\_time} \cdot \texttt{rounds}$. This way we could trivially estimate computation times on different network setups.

XI. Comparison with DACs and DGS

Coconut [48] is the first AC system with fully distributive credential issuance that preserves all the properties of previous centralized attribute-based AC systems, and it counts with real implementations [84][85]. As an AC system, the credential is composed of a blind signature of a user's chosen secret value. The innovative contribution in Coconut is the blind signature generation is distributed. The authors adopt Pointcheval-Sanders signatures [86], but use a hash function on a user's committed value to ensure that all parties in the issuing federation generate compatible signature pieces, which can be later reconstructed into a valid credential in the user side. To use this credential the user proves, in zero-knowledge, that she has a signature on a secret value. The secret value is not revealed, but the signature has to be randomized to avoid linkability in multiple authentications. Coconut does not consider pseudonymity. Indexed pseudonyms as implemented in our system could be achieved with some manipulation in the scheme construction. Namely, a verifiable pseudorandom function, including the user's secret and the CSP index as inputs, could be used to generate indexed pseudonyms. The correct construction of such pseudonym should be included in the zero-knowledge proof used for authentication. However this would increase the signature size. Additionally, as an AC system, Coconut is not designed to support anonymity revocation.

The work of Gennaro et al [46] caters for a distributed group signature that provides both, distributed issuance and distributed anonymity revocation. However, their proposed scheme is based on a different paradigm than our work that yields longer signatures. To obtain a credential the user first publishes a value linked to her real identity, which is distributively signed by the credential issuing federation. During authentication, the user proves in zero knowledge the validity of that signature. To enable revocation, the user must attach a field with an encryption of the public value linked to her identity, and prove in zero-knowledge the validity of such encryption. Anonymity revocation can be triggered by decrypting that field. The scheme adopts a distributive decryption algorithm, hence it achieves distributive anonymity revocation. When compared with our proposed system, the scheme in [46] offers faster revocation, but at the cost of longer and more complex signatures. Also, the distributive credential issuance process involves a share conversion protocol that requires several pairwise communication rounds between parties. Hence, the communication complexity increases significantly for large federations. Additionally, the scheme does not include pseudonymity. Similar to Coconut, adding this feature would increase complexity.

Unlike [46], the work in [47] achieves both distributed credential issuance and anonymity revocation while keeping a short signature size. The penalty with respect to [46] is that revocation is more expensive. It requires iterating over the transcripts of all issued credentials until finding a positive match, which is similar to our proposed system. This is acceptable in scenarios where revocation does not happen often, and having short signatures is desirable. The scheme in [47] requires a public ledger to store the transcripts of credential issuances. Those transcripts must be included by users that want to be issued a credential, and require a zero-knowledge proof to be computed by the user per each party in anonymity revocation federation. This scheme does not support pseudonymity, but it could be included with some manipulation in the scheme construction at the cost of increasing complexity and signature size.

A main advantage of our proposed scheme with respect to [47], and that also applies to [48] and [46], is that computations can be pre-processed. In our system a credential issuance can

[6]the performance evaluation test measures this value by placing the revoked credential in the middle of the stored list

be performed before the user requests a credential. In the previous systems described in this section credential issuance requires the user's input, hence pre-processing is not feasible. Similarly, anonymity revocation in our system can be pre-processed except for one communication round and two pairing operations per credential, which is less expensive than the revocation protocol in [47]. Additionally, the research works detailed in this section do not include indexed pseudonyms in their original constructions. Adding this feature would increase complexity and yield longer signatures. Another significant difference between our system and [46][47] is that our system not only enables anonymity revocation, it also allows revoking several specific pseudonyms of a given user with only one signature (see sec. VII-C). Thus, our proposed system supports fine-grained revocation, which is desirable in scenarios where users may get banned from specific services while preserving access and privacy rights in others.

XII. Conclusions

This paper caters for a pseudonym-based privacy-preserving authentication system that eliminates the requirement for a trusted authority. By adopting MPC, a set of Smart City cloud services can grant credentials to users and de-anonymize misusers securely (without leaking credentials). Credential leaks, illicit credential generation and dishonest de-anonymization are prevented if only one cloud service is honest. When compared with previous solutions, the proposed system can provide both privacy and accountability without relying on a trusted CA. In terms of efficiency, the user is not impacted by the adoption of a distributive approach. On the cloud service side, credential generation is more complex when adopting MPC, but all the operations can be pre-computed prior to the the credential issuing process. The complexity of credential revocation and de-anonymization process grows linearly with the number of issued credentials. However, centralized counterparts also presents linear complexity for revocation. The whole system has been implemented and tested extensively using Scale-Mamba and MIRACL Core, and integrated into an Android application.

XIII. Acknowledgements

The authors would like to thank Dr. Florian Caullery and Xiaojie Zhu for the technical discussions and the work conducted in the initial phase of this project.

References

[1] Ashish Pandharipande and Paul Thijssen. Connected street lighting infrastructure for smart city applications. *IEEE Internet of Things Magazine*, 2(2):32–36, 2019.

[2] F. G. Brundu et al. Iot software infrastructure for energy management and simulation in smart cities. *IEEE Trans. on Industrial Informatics*, 13(2):832–840, April 2017.

[3] Rong Du, Paolo Santi, Ming Xiao, Athanasios V. Vasilakos, and Carlo Fischione. The sensable city: A survey on the deployment and management for smart city monitoring. *IEEE Communications Surveys Tutorials*, 21(2):1533–1560, 2019.

[4] Michael Vgler, Johannes M. Schleicher, Christian Inzinger, Schahram Dustdar, and Rajiv Ranjan. Migrating smart city applications to the cloud. *IEEE Cloud Computing*, 3(2):72–79, 2016.

[5] Muhammad Usman, Muhammad Rizwan Asghar, Imran Shafique Ansari, Fabrizio Granelli, and Khalid A. Qaraqe. Technologies and solutions for location-based services in smart cities: Past, present, and future. *IEEE Access*, 6:22240–22248, 2018.

[6] Syed R. Rizvi, Susan Zehra, and Stephan Olariu. Aspire: An agent-oriented smart parking recommendation system for smart cities. *IEEE Intelligent Transportation Systems Magazine*, 11(4):48–61, 2019.

[7] Samaneh Sanaeipoor et al. Smart city: Exploring the role of augmented reality in placemaking. In *SCIOT 2020*.

[8] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *IEEE Communications Magazine*, 51(6):112–119, June 2013.

[9] Jaganathan Venkatesh, Baris Aksanli, Christine S. Chan, Alper Sinan Akyurek, and Tajana Simunic Rosing. Modular and personalized smart health application design in a smart city environment. *IEEE Internet of Things Journal*, 5(2):614–623, 2018.

[10] D. Eckhoff and I. Wagner. Privacy in the smart cityapplications, technologies, challenges, and solutions. *IEEE Communications Surveys Tutorials*, 20(1):489–516, Firstquarter 2018.

[11] H. Lin. Traceable anonymous authentication and key exchange protocol for privacy-aware cloud environments. *IEEE Systems Journal*, 13(2):1608–1617, 2019.

[12] J. Tsai and N. Lo. A privacy-aware authentication scheme for distributed mobile cloud computing services. *IEEE Systems Journal*, 9(3):805–815, 2015.

[13] D. He, N. Kumar, M. K. Khan, L. Wang, and J. Shen. Efficient privacy-aware authentication scheme for mobile cloud computing services. *IEEE Systems Journal*, 12(2):1621–1631, 2018.

[14] J. Camenisch and A. Lehmann. Privacy-preserving user-auditable pseudonym systems. In *2017 IEEE Euro S&P*.

[15] V. Sucasas, G. Mantas, J. Bastos, F. Damiao, and J. Rodriguez. A signature scheme with unlinkable-yet-acountable pseudonymity for privacy-preserving crowdsensing. *IEEE Trans. on Mobile Computing*, 2019.

[16] Yong Zhang and Jun-Liang Chen. A delegation solution for universal identity management in soa. *Services Computing, IEEE Trans. on*, 4(1):70–81, Jan 2011.

[17] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. CCS '06.

[18] A. K. Mishra, A. K. Tripathy, D. Puthal, and L. T. Yang. Analytical model for sybil attack phases in internet of things. *IEEE Internet of Things Journal*, 6(1):379–387, 2019.

[19] M. Z. Lee, A. M. Dunn, J. Katz, B. Waters, and E. Witchel. Anon-pass: Practical anonymous subscriptions. *IEEE Security Privacy*, 12(3):20–27, May 2014.

[20] Yehuda Lindell. Secure multiparty computation (mpc). Cryptology ePrint Archive, Report 2020/300, 2020. https://eprint.iacr.org/2020/300.

[21] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making spdz great again. In *EUROCRYPT (3)*, pages 158–189. Springer, 2018.

[22] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. EUROCRYPT'00.

[23] Eduardo Soria-Vazquez. *Towards secure multi-party computation on the internet: Few rounds and many parties*. PhD thesis, University of Bristol, 2019.

[24] Cedric Lauradoux, Konstantinos Limniotis, Marit Hansen, Meiko Jensen, and Petros Eftasthopoulos. Data pseudonymisation advanced techniques & use cases. European Union Agency for Cybersecurity, January, 2021.

[25] IEEE std. 1609.2-2016 (Revision of IEEE Std 1609.2-2013). Eee standard for wireless access in vehicular environments - security services for applications and management messages. pages 1–240, 2016.

[26] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *ACM SASN 2005*.

[27] X. Liu, Z. Fang, and L. Shi. Securing vehicular ad hoc networks. In *2007 2nd International Conference on Pervasive Computing and Applications*, pages 424–429, July 2007.

[28] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. Sppear: Security & privacy-preserving architecture for participatory-sensing applications. In *ACM WiSec '14*.

[29] Leping Huang, Kanta Matsuura, Hiroshi Yamane, and Kaoru Sezaki. Enhancing wireless location privacy using silent period. In *IEEE WCNC 2005*.

[30] Bidi Ying, Dimitrios Makrakis, and Hussein T. Mouftah. Dynamic mix-zone for location privacy in vehicular networks. *IEEE Communications Letters*, 17(8):1524–1527, 2013.

[31] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. extended abstract in Crypto'01.

[32] Yanchao Zhang, Wei Liu, and Wenjing Lou. Infocom 2005. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*

[33] Rongxing Lu, Xiaodong Lin, Zhiguo Shi, and X.S. Shen. A lightweight conditional privacy-preservation protocol for vehicular traffic-monitoring systems. *Intelligent Systems, IEEE*, 28(3):62–65, May 2013.

[34] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, 2004.

[35] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13.

[36] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *Theory of Cryptography*, pages 356–374, 2008.

[37] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, SAC '99, pages 184–199, London, UK, UK, 2000. Springer-Verlag.

[38] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. EUROCRYPT '01, London, UK.

[39] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02.

[40] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1 (revision 3), December 2013.

[41] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. NDSS Symposium, 2014.

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

[43] Keita Emura and Takuya Hayashi. A light-weight group signature scheme with time-token dependent linking. In *Lightweight Cryptography for Security and Privacy*, 2016.

[44] Rahaman Sazzadur, Cheng Long, Yao Danfeng Daphne, Li He, and Park Jung-Min Jerry. Provably secure anonymous-yet-accountable crowdsensing with scalable sublinear revocation. In *Proceedings on Privacy Enhancing Technologies*, PETS-2017, pages 384–403, 2017.

[45] Victor Sucasas, Georgios Mantas, Maria Papaioannou, and Jonathan Rodriguez. Attribute-based pseudonymity for privacy-preserving authentication in cloud services. *IEEE Trans. on Cloud Computing*, 2021.

[46] Rosario Gennaro, Steven Goldfeder, and Bertrand Ithurburn. Fully distributed group signatures. 2020.

[47] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. Short threshold dynamic group signatures. In *Security and Cryptography for Networks*, pages 401–423, 2020.

[48] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. *ArXiv*, abs/1802.07344, 2018.

[49] JiLiang Li et al. Aep-ppa: An anonymous, efficient and provably-secure privacy-preserving authentication protocol for mobile services in smart cities. *Journal of Network and Computer Applications*, 2019.

[50] Debiao He and Ding Wang. Robust biometrics-based authentication scheme for multiserver environment. *IEEE Systems Journal*, 9(3):816–823, 2015.

[51] Eunjun Yoon and Kee-Young Yoo. Robust biometrics-based multi-server authentication with key agreement scheme for smart cards on elliptic curve cryptosystem. *The Journal of Supercomputing*, 2010.

[52] Hakhyun Kim et al. Cryptanalysis and improvement of a biometrics-based multi-server authentication with key agreement scheme. In *ICCSA 2012*, pages 391–406, 2012.

[53] Vanga Odelu, Ashok Kumar Das, and Adrijit Goswami. A secure biometrics-based multi-server authentication protocol using smart cards. *IEEE Trans. on Inf. Forensics and Security*, 10(9):1953–1966, 2015.

[54] Libing Wu, Jing Wang, Kim-Kwang Raymond Choo, and Debiao He. Secure key agreement and key protection for mobile device user authentication. *IEEE Trans. on Inf. Forensics and Security*, 2019.

[55] Victor Sucasas et al. A privacy-enhanced oauth 2.0 based protocol for smart city mobile applications. *Comput. Secur.*, 74:258–274, 2018.

[56] Victor Sucasas, Georgios Mantas, Ayman Radwan, and Jonathan Rodriguez. A lightweight privacy-preserving oauth2-based protocol for smart city mobile apps. In *2016 IEEE Globecom Workshops*.

[57] M. Papaioannou, J.C. Ribeiro, V. Monteiro, V. Sucasas, G. Mantas, and J. Rodriguez. A privacy-preserving user authentication mechanism for smart city mobile apps. In *IEEE CAMAD 2021*, 2021.

[58] Id2020: Digital identity alliance. https://id2020.org, 2020.

[59] Decentralized identity foundation. https://identity.foundation/, 2020.

[60] Weikeng Chen, Ryan Deng, and Raluca A. Popa. N-for-1 auth: N-wise decentralized authentication via one authentication. *IACR Cryptol. ePrint Arch.*, 2021:342, 2021.

[61] Deepak et al. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *IEEE S&P 2021*, pages 1348–1366.

[62] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.

[63] J. Hoepman and L. Chmielewski. Fuzzy private matching (extended abstract). In *ARES*, pages 327–334, Los Alamitos, CA, USA, mar 2008. IEEE Computer Society.

[64] Shigeo Mitsunari, R Sakai, and M Kasahara. A new traitor tracing. E85-A, 02 2002.

[65] Carsten Baum, Daniele Cozzo, and Nigel P. Smart. Using topgear in overdrive: A more efficient zkpok for spdz. In *Selected Areas in Cryptography – SAC 2019*, pages 274–302, Cham, 2020.

[66] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. CCS '16.

[67] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. In *ESORICS*, pages 1–18, 2013.

[68] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. CCS '20, pages 1575–1590, New York, NY, USA, 2020. Association for Computing Machinery.

[69] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. Sok: General purpose compilers for secure multi-party computation. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1220–1237, 2019.

[70] Marcel Keller, Dragos Rotaru, Nigel P. Smart, and Tim Wood. Reducing communication channels in mpc. In *SCN 2018*.

[71] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round mpc combining bmr and oblivious transfer. In *ASIACRYPT 2017*.

[72] Aner Ben-Efraim, Michael Nielsen, and Eran Omri. Turbospeedz: Double your online spdz! improving spdz using function dependent preprocessing. In *ACNS 2019*.

[73] Jan Camenisch and Anja Lehmann. (un)linkable pseudonyms for governmental databases. CCS 15, page 14671479, New York, NY, USA, 2015. Association for Computing Machinery.

[74] Ivan Damgård, Kasper Damgård, Kurt Nielsen, Peter Sebastian Nordholt, and Tomas Toft. Confidential benchmarking based on multiparty computation. In *FC*, pages 169–187. Springer, 2016.

[75] Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Tinykeys: A new approach to efficient multi-party computation. In *CRYPTO 2018*.

[76] Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-

Vazquez. Concretely efficient large-scale mpc with active security (or, tinykeys for tinyot). In *ASIACRYPT*, pages 86–117. Springer, 2018.

[77] Aner Ben-Efraim, Kelong Cong, Eran Omri, Emmanuela Orsini, Nigel P Smart, and Eduardo Soria-Vazquez. Large scale, actively secure computation from lpn and free-xor garbled circuits. In *EUROCRYPT*, pages 33–63. Springer, 2021.

[78] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. CCS '16, page 830842, New York, NY, USA, 2016. ACM.

[79] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *JOURNAL OF CRYPTOLOGY*, 13:361–396, 2000.

[80] Abdelrahaman Aly, K Cong, D Cozzo, M Keller, E Orsini, D Rotaru, O Scherer, P Scholl, N Smart, T Tanguy, et al. Scale–mamba v1. 12: Documentation, 2021.

[81] Michael Scott. The miracl core library. MIRACL Labs, 2021. https://github.com/miracl/core/blob/master/MIRACL_Core.pdf.

[82] Mark Manulis, Nils Fleischhacker, Felix Gunther, Franziskus Kiefer, and Bertram Poettering. Group signatures: Authentication with privacy. Bundesamt fur Sicherheit in der Informationstechnik. Tech. Rep., 2012.

[83] Vireshwar Kumar et al. Group signatures with probabilistic revocation: A computationally-scalable approach for providing privacy-preserving authentication. CCS '15.

[84] Denis J. Roio, Alberto Ibrisevic, and Andrea D'Intino. Reflow: Zero knowledge multi party signatures with application to distributed authentication. *ArXiv*, abs/2105.14527, 2021.

[85] Harry Halpin Claudia Diaz and Aggelos Kiayias. The nym network. the next generation of privacy infrastructure. 2021.

[86] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *CT-RSA 2016*.

XIV. Biographies

Victor Sucasas is a Cryptography Engineer and Senior Director of the Confidential Computing Team at the Cryptography Research Centre at the Technology Innovation Institute (TII), Abu Dhabi, EAU. He received his Ph.D. on Electrical and Electronic Engineering from the 5G Innovation Centre at the University of Surrey (UK) in 2016. He was also a Marie Curie ESR and post-doctoral researcher at Instituto de Telecomunicações (Portugal). His research interests cover privacy-preserving authentication, secure multi-party computation and privacy-preserving machine learning. He has also worked on wireless networks, underwater networks, secure cognitive radio systems and physical layer security. He is a Senior IEEE member and an EAI Fellow.

Abdelrahaman Aly is a Senior Cryptographer at the Cryptography Research Centre, Technology Innovation Institute (TII), Abu Dhabi, UAE. He is also an Associate Researcher at imec-Cosic at KU Leuven, Leuven, Belgium. His research is focused on Applied aspects of Cryptography, more specifically, on topics related to secure distributed computation, including MPC. He obtained his Ph.D at the University Catholique de Louvain and then a postdoctoral fellow at KU Leuven, under the supervision of Bart Preneel and Nigel Smart. A poll analysis enthusiast and 4v4 Diamond League SC2 player, He holds an Engineering Diploma from the Escuela Politecnica del Ejercito, Ecuador.

Georgios Mantas received his M.Sc. degree in Information Networking from Carnegie Mellon University, PA, USA, in 2008, and his Ph.D. degree in Electrical and Computer Engineering from the University of Patras, Greece, in 2012. In 2014, he became a Researcher at the Telecomunicações, Aveiro, Portugal, where he has been involved in several ICT security research projects, such as H2020-MSCA-ITN-SECRET, ECSEL-IA-SemI40, CATRENE-MobiTrust, CATRENE-NewP@ss, and ARTEMIS-ACCUS. Currently, he is a Senior Lecturer in Digital Security and Management in the School of Engineering at the University of Greenwich, UK. He is an HEA Fellow, and a member of IEEE Communications Society, IEEE Computer Society, and IEEE Technology and Engineering Management Society.

Jonathan Rodriguez received his Ph.D from the University of Surrey (UK) in 2004. In 2005, he became a researcher at the Instituto de Telecomunicaes (Portugal) and Senior Researcher in 2008. He has served as Project Coordinator for major international research projects (Eureka LOOP, FP7 C2POWER, H2020-MSCA-SECRET), whilst acting as technical manager for FP7 COGEU and FP7 SALUS. He is currently a Project Director for the PHYSEC project funded under the NATO Science for PEACE and Security program. He is author of more than 600 published works on security and radio communications, and currently serving as Associate Editor for IEEE Access and IET Communications journal. Since 2017, he is Full Professor in Mobile Communications at the University of South Wales (UK). His professional affiliations include IEEE Senior Member, CEng. (2013), FIET (2015) and FRSA (2022).

Najwa Aaraj is a Chief Researcher at the Technology Innovation Institute (UAE) and Adjunct Professor at Okinawa Institute of Science and Technology (JAPAN). She holds a Ph.D. and Masters degree in Computer Engineering from Princeton University. Her expertise lies in applied cryptography, trusted platforms, secure embedded systems, software exploit detection/prevention, machine learning and biometrics. She has over 17 years of experience working in the United States, Australia, Middle East, Africa, and Asia with global firms. Before joining TII, Dr. Aaraj was Senior Vice President of Cryptographic technologies development at DarkMatter, a cyber-security firm based in the UAE. She was formerly at Booz & Co., where she led engagements in the telecommunications and information technology industry for clients globally. She also held research positions with the Embedded Systems Security Group at IBM T.J. Watson, at Intel Portland, Oregon, and at NEC Laboratories in Princeton, New Jersey.