IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 13, 2021, accepted September 21, 2021, date of publication October 4, 2021, date of current version October 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117452

Privacy-Enhancing Group Signcryption Scheme

SARA RICCI, PETR DZURENDA, JAN HAJNY, AND LUKAS MALINA
Department of Telecommunications, FEEC, Brno University of Technology, 616 00 Brno, Czech Republic

Corresponding author: Sara Ricci (ricci@vut.cz)

This work was supported in part by the European Union, Ministry of Education, Youth and Sports, Czech Republic and Brno, University of Technology under International Mobility Project MeMoV under Grant CZ.02.2.69/0.0/0.0/16_027/00083710; in part by the Technology Agency of Czech Republic under "Legal and technical tools for privacy protection in cyberspace" under Project TL02000398; and in part by the European Union's Horizon 2020 Research and Innovation Program (Project SPARTA) under Agreement 830892.

ABSTRACT In the last decades, several signcryption schemes have been developed for different privacy-enhancing purposes. In this paper, we propose a new privacy-enhancing group signcryption scheme that provides: unforgeability, confidentiality, ciphertext and sender anonymity, traceability, unlinkability, exculpability, coalition-resistance, and unforgeable tracing verification. It is important to notice that the proposed scheme allows a signer to anonymously signcrypt a message on the group's behalf (i.e., sender's anonymity). The security analysis of the scheme is also provided. Our proposal is proven to be strongly existentially unforgeable under an adaptive chosen message attack, indistinguishable under an adaptive chosen ciphertext attack, and to provide ciphertext anonymity under an adaptive chosen ciphertext attack. Furthermore, the scheme is extended to work in a multi-receiver scenario, where an authorized group of receivers is able to unsigncrypt the ciphertext. The experimental results show that our scheme is efficient even on computationally restricted devices and can be therefore used in many IoT applications. The `Signcrypt` protocol on smart cards takes less than 1 s (including communication overhead). The time of the `Unsigncrypt` protocol on current ARM devices is negligible (less than 40 ms).

INDEX TERMS Anonymity, embedded devices, group signature, privacy-enhancing technology, signcryption protocol, smart cards.

I. INTRODUCTION

A signcryption scheme [45] combines a digital signature and a public-key encryption scheme with a lower computational and communication overhead than traditional sing-then-encrypt scheme. Most of the traditional signcryption protocols are based on the Diffie-Hellman problem. These schemes guarantee data confidentiality and integrity, as well as signature unforgeability. In signcryption protocols, users' privacy is basically achieved by *ciphertext anonymity* which means that the ciphertext reveals no information about who created it nor about whom it is intended to [45]. In other words, the problem is to hide sender's and receiver's identity to an outsider. The use of bilinear pairing in a signcryption protocol allows achieving ciphertext anonymity property at the expense of speed, e.g., see [14], [37], [39]. However, many schemes require an even stronger anonymity. For instance, in the case of e-voting, the voter's (sender's) identity has to be hidden also to the receiver as well as in the case of video streaming applications where anonymous users (senders) broadcast live video to the Internet. In other words, we should be able to identify malicious users, e.g., users who broadcast a video with prohibited content, while keeping honest-user identity hidden. Group signatures can help us with that. In fact, group signatures allow providing data authenticity without disclosing users' identities. In particular, a user can anonymously sign a message on behalf of the group. Therefore, our scheme uses group signature and bilinear maps in order to provide ciphertext anonymity plus sender anonymity.

The associate editor coordinating the review of this manuscript and approving it for publication was Pierluigi Gallo.

II. STATE OF THE ART

Most of the *standard* (i.e., one-to-one) *signcryption protocols* propose a bilinear pairing strategy in order to reach stronger anonymity property. In fact, the use of bilinear pairing in a signcryption protocol allows achieving ciphertext anonymity property at the expense of speed. Libert and Quisquater [28], [45] propose a scheme based on pairing which is only partially anonymous. In fact, an outsider cannot identify who was the sender but knows who the receiver is, and the receiver needs sender's public-key to

unsigncrypt the message. Therefore, the scheme does not achieve sender's anonymity. Later, Chaudhari and Das [14] introduce a pairing-based scheme where the sender and the receivers identities are protected against an outsider, i.e., the scheme guarantees ciphertext anonymity. This proposal can be suitable for a multi-receiver environment, where only authorized receivers can decrypt the ciphertext and verify the signature. However, this scheme does not also provide sender's anonymity. Finally, Braeken and Touhafi [10] propose a fast signcryption scheme based on the elliptic curve discrete logarithm problem. As in any non-pairing scheme, the anonymity is only partially achieved, i.e. the sender's identity is known by the receiver.

Most of the *multi-receiver* (i.e., one-to-many) *signcryption schemes* generate different encryptions of the same message, that is one ciphertext for each authorized receiver. These ciphertexts are then concatenated in one, which is broadcasted. Therefore, if some part of the ciphertext goes wrong during transmission, only some authorized receivers can decrypt the message correctly while the rest cannot. This leads to the unfair decryption problem [38]. Pang *et al.* [37] present a pairing-based scheme where each receiver needs the whole ciphertext for decryption. However, the identity of the sender is disclosed by an authorized receiver after decryption. Moreover, in order to hide receivers' identity to an outsider, the Lagrange interpolation polynomial was also considered [24]. The unique ciphertext can be decrypted by any authorized receiver who owns a root of the interpolation polynomial. Later, this method was used in several *anonymous multi-receiver signcryption schemes* [27], [38], [44]. Unluckily, Li and Pang [26] pointed out that any scheme based on Lagrange interpolation polynomial methodology cannot achieve the receiver's anonymity and, accordingly, ciphertext anonymity. In fact, every authorized receiver can determine whether the other is one of the authorized receivers. To our knowledge, no current signcryption scheme could combine ciphertext anonymity and fair decryption.

Ring signcryption schemes were presented more recently. Huang *et al.* [20] propose to combine pairing-based signcryption scheme with ring signature. In this case, a sender can anonymously signcrypt a message on behalf of the group. However, the receiver's identity is not hidden to an outsider. Saraswat *et al.* [39] also present an anonymous proxy signcryption scheme based on pairing and ring signature. This scheme works in a different scenario, and it is only required ciphertext anonymity. Li *et al.* [25] also propose a scheme where sender's and receiver's identities are hidden to an outsider. Their scheme is designed to be efficient on the sender side and suitable for wireless body area networks.

At last, only a few articles dealt with *group signcryption schemes*. Mu and Varadharajan [33] propose a distributed signcryption scheme based on ElGamal encryption and Schnorr's digital signature. The scheme is then extended to a group signcryption protocol. However, Kwak *et al.* [23] proves that Mu-Varadharajan does not provide exculpability security property, i.e., the group manager can signcrypt on the behalf of other group members. Furthermore, Kwak and Moon develop a new distributed signcryption scheme with sender anonymity and extend it to a group signcryption scheme [22]. However, Bao *et al.* [3] demonstrate that Kwak's and Moon's scheme is insecure. In particular, the scheme does not provide unforgeability, coalition-resistance, and traceability security properties. Then Kwak *et al.* overcome the aforementioned security flaws in [23]. They present a new encrypted group signature scheme based on Ateniese-Camenisch-Joye-Tsudik (ACJT) group signature [2], Bresson-Chevassut-Essiari-Pointcheval (BCEP) group key agreement protocol [11], and ElGamal cryptosystem [16]. The scheme is defined by the authors as an ''encrypted group signature scheme'' and follows the traditional sign-then-encrypt mechanism. At first, the user generates the ACJT group signature on the message. Then the user encrypts it, together with the message, by using the symmetric cipher. The encryption key is encrypted by ElGamal cryptosystem and delivered to the targeted group, where each member knows the decryption key. Moreover, the decryption key is distributed within the group by BCEP protocol. The scheme provides data confidentiality and unforgeability similarly to a signcryption scheme. Unfortunately, this scheme does not provide lower computational and communicational overhead due to its sequential sign-then-encrypt nature. Furthermore, the scheme is not suitable for constrained devices in the Internet of Thing (IoT) since it is based on Integer Factorization (IF) problem which is not portable to Elliptic Curve (EC) constructions. The ACJT scheme has also high computational requirements as shown in [30]. Plain Kwak *et al.* [23] encrypted group signature scheme is then used by Cho and Toshiba [15] to build a verifiable group sygncryption scheme with deduplicable properties for data stored in a cloud service provider. It is remarkable that our scheme can be an efficient alternative to be deployed in Cho-Toshiba's scheme. At last, Mohanty *et al.* [32] present a signcryption protocol based on the Diffie-Hellman problem. The scheme tries to provide also the user's anonymity. In particular, the identity of the sender is hidden to the receiver and an outsider. In order to achieve this kind-of-anonymity, the scheme requires an active group manager who is involved in the signcryption phase. This manager's involvement can lead to privacy leakage and slow down the computation, and therefore, it is normally avoided. However, the scheme presents several security flows as discussed in Appendix E.

Table 1 shows a comparison of existing signcryption schemes. Observe that in the table two anonymity types are considered: 1) ciphertext anonymity, i.e., sender's identity is hidden to an outsider as well as receiver's identity to an outsider, and 2) sender anonymity, i.e., sender's identity is hidden not only to an outsider but also to the receiver. The level of privacy achieved by the below schemes depends on how many anonymity types they cover. Note that the property ''receiver's identity is hidden to the sender'' is not contemplated in the previous list because it is normally supposed that the sender knows the identity of the receiver of its message.

TABLE 1. Main features of related work on anonymous signcryption schemes: scheme, security assumption, anonymity property, multi-receiver scenario support and number of bilinear pairings used in the corresponding scheme. "DH" stands for "Diffie-Hellman problem", "pair." for "bilinear Diffie-Hellman problem", "interp-pair." for "bilinear Diffie-Hellman problem combined with Lagrange interpolation polynomial method", "ring-pair." for "bilinear Diffie-Hellman problem combined with ring signature", "EC" for "Elliptic curve discrete logarithm problem", "group-pair." for "bilinear Diffie-Hellman problem combined with group signature", "group-DH" for "Diffie-Hellman problem combined with group signature", "S-to-R" for "sender's identity is hidden to the receiver", similarly for "S-to-O" and "R-to-O" where "O" stands for "outsider". The number of pairings is given depending on who is computing it. For instance, "3S+7R" stands for "3 pairings computed by the sender and 7 by the receiver"; "n" is the number of users in the ring.

Scheme	Problem	Sender Anonymity	Ciphertext Anonymity		Multi-receiver	# pairing
		S-to-R	S-to-O	R-to-O		
Signcryption Scheme						
Libert [28]	pair.	✗	✓	✗	✗	$2R$
Chaudhari [14]	pair.	✗	✓	✓	✓	$3S+7R$
Pang [38]	interp-pair.	✓	✓	✗	✓	$4R$
Pang [37]	ring-pair.	✗	✓	✓	✓	$nS+3R$
Huang [20]	ring-pair.	✓	✓	✗	✗	$(n+1)S+3R$
Saraswat [39]	ring-pair.	✗	✓	✓	✓	$1S+3R$
Li [25]	pair.	✗	✓	✓	✗	$2R$
Braeken [10]	EC	✗	✓	✓	✗	-
Encrypted Group Signature Scheme						
Kwak [23]	group-DH	✓	✓	✓	✓	-
Group Signcryption Scheme						
Our Proposal	group-pair.	✓	✓	✓	✓	$2R$

Note: ✓– the algorithm provides this property, ✗ –the algorithm does not provide this property.

TABLE 2. Security comparison of available group signcryption schemes.

Security property	Mu [33]	Kwak [22]	Kwak [23]	Mohanty [32]	Our
Correctness	✓	✓	✓	-	✓
Unforgeability	✓	✗	✓	✗	✓
Confidentiality	✓	✓	✓	✗	✓
Ciphertext anonymity	✓	✓	✓	-	✓
Sender's anonymity	✓	✓	✓	-	✓
Unlikability	✓	✓	✓	-	✓
Exculpability	✗	✓	✓	✗	✓
Traceability	✓	✗	✓	✗	✓
Coalition-resistance	✓	✗	✓	✗	✓
Unforgeable tracing verification.	✗	-	✓	-	✓

Note: ✓– the algorithm has this property, ✗ – the algorithm does not provide this property, - – information not available.

In Table 1, we consider only provable secure signcryption schemes. In particular, Mu and Varadharajan sheme [33] does not provide exculpability, and Kwak *et al.* [22] scheme does not provide traceability, coalition-resistance and unforgeability. Furthermore, we also prove that the Mohanty *et al.* [32] scheme presents security flows, i.e., it does not provide unforgeability, confidentiality, exculpability nor traceability. See Appendix E for more details. The security of the state-of-the-art schemes is depicted in detail in Table 2.

A. CONTRIBUTION AND PAPER STRUCTURE

In [19], we proposed a novel group signature scheme based on the weak Boneh-Boyen signature [9] and the efficient proofs of knowledge [13]. This scheme has fast signature generation and provides all the main privacy-enhancing signature features, i.e., anonymity, unlinkability, traceability, and coalition-resistance. The present article extends this work, where the proposed signature is included in our signcryption scheme. Accordingly, the new signcryption scheme holds all the properties of the aforementioned group signature scheme. Our lightweight privacy-preserving group signcryption scheme can find use in particular in IoT environments, where many computationally and memory-constrained devices are employed.

Our novel signcryption scheme guarantees ciphertext and sender anonymity. This is achieved by combining the Elliptic Curve Integrated Encryption Scheme (ECIES) [18] with our group signature [19]. Furthermore, our signcryption scheme supports the multi-receiver scenario and guarantees fair decryption to all authorized receivers.

The main properties of the scheme are summarized below.

Privacy-enhancing main features:

- **ciphertext anonymity**, i.e., sender and receiver identity is hidden to an outsider;
- **sender anonymity**, i.e., the sender's identity is hidden not only to an outsider but also to the receiver. In this way, instead of sender authentication, group authentication is provided to achieve message integrity and verification of the sender;
- **traceability**, i.e., the manager is able to trace which user signcrypt the message.
- **unlinkability**, i.e., two or more signcryptions cannot be addressed to the same or different senders;

Other features:

- the `Signcrypt` algorithm is *fast*: it requires no bilinear pairing and only 6 exponentiations;
- the `Unsigncrypt` algorithm is efficient: it requires only 2 pairings;
- the group manager is able to identify the signer by opening the signcryption;
- the scheme is compatible with current revocation techniques such as [13];
- the scheme can be adapted to a multi-receiver scenario;

- the scheme is built by using primitives with formal security proofs;
- security analyses of the scheme are provided.

The rest of this article is organized as follows. Section III discusses some preliminaries. Section IV lists the signcryption properties and security models. Section V shows the basic structure of the proposed scheme and lists the integrated cryptographic primitives with their functionalities. Section VI presents the proposed scheme. Section VII shows how the scheme can be adapted to a multi-receiver scenario. Section VIII provides the security analysis of the scheme. Section IX discusses possible use cases for our proposal. Section X shows the comparison with closely-related signcrytion schemes. Section XI reports the experimental results. The final section contains the conclusions.

III. PRELIMINARIES

In this section, at first, we outline the used notation and the security assumptions needed to understand our scheme and our security proofs. At second, we briefly introduce bilinear pairing maps and weak Boneh-Boyen (wBB) signature which are used throughout all sections. Then we review the protocols on which our scheme is based, namely our lightweight group signature [19], a Non-Interactive Zero-Knowledge Proof of Knowledge (NIZKPK) [7], the Elliptic Curve Integrated Encryption Scheme (ECIES) [18], and the BCEP group key agreement protocol [11]. At last, we refresh the structure of a signcryption protocol.

From now on, the symbol ":" means "such that", "$|x|$" is the bitlength of x and "$\|$" denotes the concatenation of two binary strings. We write $a \xleftarrow{\$} A$ when a is sampled uniformly at random from A. A secure hash function is denoted as $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, where κ is a security parameter. We describe the proof of knowledge protocols (PK) using the notation introduced by Camenisch and Stadler (CS) [12]. The protocol for proving the knowledge of discrete logarithm of c with respect to g is denoted as $\text{PK}\{\alpha : c = g^\alpha\}$.

A. HARD PROBLEMS

In this section, we describe some security assumptions used in the proposed scheme. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order q, g be a generator of $\mathbb{G}_1$, and g_2 be a generator of $\mathbb{G}_2$. In the first assumption, $\mathbb{G}_1$ is taken equal to $\mathbb{G}_2$ and, therefore, $g = g_2$.

1) DECISIONAL DIFFIE-HELLMAN (DDH) PROBLEM

Given $\langle g, g^a, g^b, g^c \rangle$ for some $a, b, c \in \mathbb{Z}_q$, determine whether $c \equiv ab \mod q$.

Definition 1 (DDH Assumption): Let B be an algorithm with output in $\{0,1\}$, which has advantage

$$Adv_B^{DDH} = Pr[u, v \leftarrow \{0, \dots q\} : B(g, g^u, g^v, g^{uv}) = 1] - Pr[u, v \leftarrow \{0, \dots q\}; h \leftarrow B : B(g, g^u, g^v, h) = 1]$$

in solving the DDH problem. If for any t-time algorithm the advantage Adv_B^{DDH} is negligible ($\leq \epsilon$), we say that the (q, t, ϵ)-DDH assumption holds.

See [41] for more details on the DDH assumption.

2) STRONG DIFFIE-HELLMAN (p-SDH) PROBLEM

Given as input a $(p+3)$-tuple of elements

$$(g, g^x, g^{x^2}, \dots, g^{x^p}, g_2, g_2^x) \in \mathbb{G}_1^{p+1} \times \mathbb{G}_2^2,$$

compute a pair $(c, g^{1/(x+c)}) \in \mathbb{Z}_q \times \mathbb{G}_1$ for some value $c \in \mathbb{Z}_q \setminus \{x\}$.

Definition 2 (p-SDH Assumption): Let B be an algorithm with advantage

$$Adv_B^{p\text{-}SDH} = Pr[B(g, g^x, g^{x^2}, \dots, g^{x^p}, g_2, g_2^x) = (c, g^{1/(x+c)})]$$

in solving the p-SDH problem. If for any t-time algorithm the advantage $Adv_B^{p\text{-}SDH}$ is negligible ($\leq \epsilon$), we say that the (p, t, ϵ)-SDH assumption holds.

See [9] for more details on p-SDH assumption.

B. BILINEAR PAIRING

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order q. A bilinear map $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ must satisfy:

- **bilinearity**: $\mathbf{e}(g^x, g_2^y) = \mathbf{e}(g, g_2)^{xy}$ for all $x, y \in \mathbb{Z}_q$;
- **non-degeneracy**: for all generators $g \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, $\mathbf{e}(g, g_2)$ generates $\mathbb{G}_T$;
- **computability**: there exists an efficient algorithm $\mathcal{G}(1^k)$ to compute $\mathbf{e}(g, g_2)$ for all $g \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

By definition $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g, g_2)$ is a bilinear group if it satisfies all above properties. In this article, we consider the case $\mathbb{G}_1 \neq \mathbb{G}_2$ that is when $\mathbf{e}$ is an asymmetric bilinear map and DDH assumption hold. Moreover, having $\mathbb{G}_1 \neq \mathbb{G}_2$ permits to obtain the shortest possible signature (check [9] for more details).

C. WEAK BONEH-BOYEN SIGNATURE

The wBB signature scheme is a pairing-based short signature scheme. This signature was proven existentially unforgeable against a weak (non-adaptive) chosen message attack under the Strong Diffie-Hellman assumption [9]. The scheme can be used to efficiently sign messages and can be also integrated with the zero-knowledge proofs [13]. In this way, the knowledge of signed messages can be proven anonymously, and unlinkably. The wBB signature is briefly depicted below:

- $(pk_s, sk, par) \leftarrow \texttt{KeyGen}(1^\kappa)$: on the input of the system security parameter κ, the algorithm generates a bilinear group $par = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g, g_2)$, computes $pk_s = g_2^{sk}$ where $sk \xleftarrow{\$} \mathbb{Z}_q$, and outputs sk as the private key and (pk_s, par) as the public key.
- $(\sigma) \leftarrow \texttt{Sign}(m, par, sk)$: on the input of the message $m \in \mathbb{Z}_q$, the system security parameters par and the secret key sk, the algorithm outputs the signature of the message $\sigma = g^{\frac{1}{sk+m}}$.
- $(1/0) \leftarrow \texttt{Verify}(\sigma, m, pk_s, par)$: on the input of the system security parameters par, the public key pk_s, a signature σ and a message m, the algorithm returns 1 if and only if $\mathbf{e}(\sigma, pk_s) \cdot \mathbf{e}(\sigma^m, g_2) = \mathbf{e}(g, g_2)$ holds, i.e. the signature is valid, and 0 otherwise.

D. LIGHTWEIGHT GROUP SIGNATURE

In our previous article [19], we develop a fast group signature based on wBB proposal. Our signature allows a signer to generate an anonymous signature $\sigma(sk_i, m)$ on a message m, where sk_i is the signer's private key. The protocol works as follows:

- $(pk, sk_m, par) \leftarrow \texttt{Setup}(1^\kappa)$: on the input of the security parameter κ, the algorithm generates the bilinear group with parameters $par = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g \in \mathbb{G}_1, g_2 \in \mathbb{G}_2)$ satisfying $|q| = \kappa$. It also generates the manager's private key $sk_m \xleftarrow{\$} \mathbb{Z}_q$ and computes the public key $pk = g_2^{sk_m}$. It outputs the (pk, par) as a public output and the sk_m as the manager's private output.
- $(sk_i, rd) \leftarrow \texttt{KeyGen}(id_i, sk_m)$: on the input of manager's private key sk_m and signer's private identifier id_i. The protocol outputs the wBB signature $sk_i = g^{\frac{1}{sk_m + id_i}}$ to the signer and updates the manager's revocation database rd by storing id_i.
- $\sigma(sk_i, m) \leftarrow \texttt{Sign}(m, id_i, sk_i)$: on the inputs the signer's private identifier id_i, signer's private key sk_i, and the message m, the algorithm outputs the signature $\sigma(sk_i, m) = (g', sk_i', \bar{sk_i}, \pi)$, where:
  - $g' = g^r$: the generator raised to a randomly chosen randomizer $r \xleftarrow{\$} \mathbb{Z}_q$.
  - $sk_i' = sk_i^r$: the signers' private key raised to the randomizer.
  - $\bar{sk_i} = sk_i'^{-id_i}$: the randomized private key raised to the signer identifier.
  - $\pi = PK\{(id_i, r) : \bar{sk_i} = sk_i'^{-id_i} \wedge g' = g^r\}(m)$: proof of knowledge of r and id_i signing the message m.
- $0/1 \leftarrow \texttt{Verify}(\sigma(sk_i, m), m, pk, bl)$: on the input of the message m, its signature $\sigma(sk_i, m)$, a blacklist bl, and the public key pk, the algorithm checks the proof of knowledge signature π and checks that the signature is valid with respect to the manager's public key using the equation $\mathbf{e}(\bar{sk_i} g', g_2) \stackrel{?}{=} \mathbf{e}(sk_i', pk)$. The collector also performs the revocation check $sk_i' \stackrel{?}{=} \bar{sk_i}^{id_i}$ for all id_i values stored on the blacklist bl. If the revocation check equation holds for any value on the blacklist, the signature is rejected. Otherwise, the signature is accepted if all other checks pass.

In the above algorithm, the manager knows the signer's private key sk_i. In our signcryption protocol, we overcome this issue and we achieve the exculpability feature of the signature. Therefore, the manager is not able to signcrypt a message on behalf of any group signer. Note that traceability of malicious signers remains possible.

E. NON-INTERACTIVE ZERO-KNOWLEDGE PROOF OF KNOWLEDGE (NIZKPK) OF AN AUTHENTICATOR

The following NIZKPK [7] allows two entities, namely the manager and the sender, to jointly compute a Boneh-Boyen signature $\sigma = g^{1/(K+m)}$ of a sender's private message m and the manager's secret key K. Let C_m be a commitment on the message m created by the sender. Let $\texttt{Keygen}$, $\texttt{Enc}$, and $\texttt{Dec}$ be an additively homomorphic semantically secure encryption scheme. Let $\oplus$ denote the homomorphic operation on ciphertexts and $e \otimes r$ denote "adding" a ciphertext e to itself r times, where r is an integer. The NIZKPK scheme is briefly depicted below:

- On the input of the system security parameter κ, the manager generates $(pk_h, sk_h) \leftarrow \texttt{KeyGen}(1^\kappa)$ in such a way that the message space is of size at least $2^\kappa q^2$, where $|q| = \kappa$.
- The manager computes $e_1 = \texttt{Enc}(pk_h, K)$ and sends e_1, pk_h to the sender.
- The manager and the sender engage in an interactive zero-knowledge proof that e_1 encrypts to a message $m \in [0, q]$.
- The sender chooses $r_1 \xleftarrow{\$} \mathbb{Z}_q$ and $r_2 \xleftarrow{\$} \{0, \ldots, 2^\kappa q\}$ and computes

$$e_2 = ((e_1 \oplus \texttt{Enc}(pk_h, m)) \otimes r_1) \oplus Enc(pk_h, r_2 q),$$

  and sends e_2 to the manager.
- The manager and the sender perform an interactive zero-knowledge proof in which the sender shows that e_2 has been correctly computed using the message in the commitment C_m, and that r_1, r_2 are in the appropriate ranges.
- The manager decrypts $x = \texttt{Dec}(sk_h, e_2)$ and sends $\sigma^* = g^{1/x}$ to the sender.
- The sender computes $\sigma = (\sigma^*)^{r_1}$ and verifies that it is a correct wBB signature on m. Note that the manager obtains no information on m.

Belenkiy *et al.* [7] prove that this construction is a secure two-party computation of Boneh-Boyen signature. Moreover, they show how NIZKPK can be efficiently implemented using Paillier cryptosystem [36] for their delegatable anonymous credentials scheme. We refer to [7] for more details. Note that NIZKPK can be easily adapted to work with our signcryption scheme.

F. ELLIPTIC CURVE INTEGRATED ENCRYPTION SCHEME (ECIES)

ECIES [18] is an efficient and provable-secure encryption scheme based on the elliptic curve discrete logarithm problem. Let $\mathbb{G}$ denote a group of prime order q with generator g. Then the public system parameters are $par = (\mathbb{G}, g, q)$. The scheme needs a symmetric encryption scheme $SYM = (E_k, D_K)$, a message authentication code MAC_k, and a key derivation function KDF. The ECIES scheme is briefly depicted below:

- $(pk, sk) \leftarrow \texttt{KeyGen}(par)$: on the input of the system parameters par, the protocol randomly chooses the secret key $v \xleftarrow{\$} \mathbb{Z}_q$ and computes the public key $pk = g^v$.
- $(e) \leftarrow \texttt{Enc}(par, pk, m)$: on the input of the public key pk and a message m, the protocol randomly chooses $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $u = g^x$ and $t = pk^x$. Then it

computes the keys $(k_1, k_2) = KDF(t)$ which are used for encrypting the message $c = E_{k_1}(m)$ and for generating the message authentication code $r = MAC_{k_2}(c)$ of ciphertext c. The algorithm outputs $e = u||r||c$.

- $(\perp / m) \leftarrow \texttt{Dec}(par, sk, e)$: on the input of the secret key sk and the ciphertext c, the protocol parses e as $u||r||c$, and computes $t = u^{sk}$ and $(k_1, k_2) = KDF(t)$. If $r = MAC_{k_2}(c)$, then the algorithm returns $m = D_{k_1}(c)$, otherwise invalid $\perp$.

In addition to proving that the algorithm is secure, Smart [41] provides several specifications on the choice of *SYM* and *KDF*. Our signcryption scheme builds on the ECIES scheme and takes into consideration Smart's recommendations.

G. BCEP GROUP KEY AGREEMENT PROTOCOL

The BCEP group key agreement protocol [11] is an efficient and provable-secure group key agreement protocol. The scheme security is based Computational Diffie–Hellman (CDH) problem. Furthermore, the scheme requires the employment of a secure signature scheme. Let $\mathbb{G}$ denote a group of prime order q with generator g. The BCEP algorithm is run between a User (U_i) (in user group $\mathcal{G}_U$) and a Server S which will be renamed in our protocol as a Receiver and the Receiver Group Manager, respectively. The BCEP scheme is briefly depicted below:

- The user U_i generates $x_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $y_i = g^{x_i}$. Then the user generates the signature σ_i on the value y_i and sends (σ_i, y_i) to the server.
- The server generates a random $x_s \xleftarrow{\$} \mathbb{Z}_q$ and computes $y_s = g^{x_s}$. The server verifies the signature (σ_i, y_i) for each user U_i and computes $\alpha_i = y_i^{x_s}$. Then the server initializes the counter $c = 0$, as a bit-string of length ℓ_1, and computes the shared secret value $k = \mathcal{H}_0(c||\alpha_1||\ldots||\alpha_n)$, where the $\mathcal{H}_0 : \{0,1\}^* \rightarrow \{0,1\}^{\ell_0}$ is a secure hash function with output length ℓ_0 and n is the number of users.
- Finally, the server computes $k_i = k \oplus \mathcal{H}_1(c||\alpha_i)$, where $\mathcal{H}_1 : \{0,1\}^{\ell_1} \times \mathbb{G} \rightarrow \{0,1\}^{\ell_0}$ is a secure hash function, where ℓ_1 is the maximal bit-length of a counter c used to prevent replay attacks. The server signs the message $m_i = c||k_i||y_s$ and sends (m_i, σ_s) to each user.
- Each user U_i verifies the signature (m_i, σ_s) and computes $\alpha_i = y_s^{x_i}$ in order to recover the shared secret key k and the session key sk as depicted below:

$$k = k_i \oplus \mathcal{H}_1(c||\alpha_i),$$
$$sk = \mathcal{H}_2(k||\mathcal{G}_U||S),$$

where $\mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^{\ell_2}$ is a secure hash function with output length ℓ_2 that need not be equal to ℓ_0.

This algorithm is integrated without modifications in our multi-receiver group signcryption protocol. See Section VII for more details.

H. SIGNCRYPTION SCHEME ARCHITECTURE

In this section, we briefly refresh the structure of a signcryption protocol. A traditional signcryption protocol consists of at least four basic algorithms: `Setup`, `KeyGen`, `Signcrypt`, `Unsigncrypt`. In particular, for a fixed security parameter, these algorithms work as follows:

$(pk_s, par) \leftarrow \texttt{Setup}(1^\kappa)$: on the input of the security parameter κ, the algorithm outputs the public system security parameters par and the group public key pk_s.
$(sk_s, pk_s, pk_r, sk_r) \leftarrow \texttt{KeyGen}(par)$: on the input of par, generates sender's secret and public keys (sk_s, pk_s), and receiver's key pair (pk_r, sk_r).
$(c, \sigma) \leftarrow \texttt{Signcrypt}(par, sk_s, pk_r, m)$: on the input of par, sk_s and pk_r and a message m, outputs a ciphertext c and a signature σ.
$(1/0, m) \leftarrow \texttt{Unsigncrypt}(par, c, \sigma, pk_s, sk_r)$: on the input of par, c, σ, pk_s and sk_r, verifies the signature σ and decrypts the ciphertext c. It returns 1 and m iff the signature is valid and 0 otherwise.

IV. SECURITY MODEL AND REQUIREMENTS

In this section, the signcryption security model and security requirements are presented. At first, basic and privacy-enhancing properties of a group sygncryption scheme are listed and delineated. Then Strong Existential Unforgeability (sUF), Indistinguishability (IND), and Ciphertext anonymity (ANON) are described in detail.

A. SECURITY REQUIREMENTS

In general, a group signcryption protocol should have the following security properties:

- **Correctness:** Valid signcryptions generated by group members are always accepted via a verification process, while invalid signcryptions always fail verification.
- **Unforgeability:** Only valid group members are able to signcrypt a message on behalf of the group.
- **Confidentiality:** No one can recover the signcrypted message, except for either the receiver or the members belonging to the receiving group.
- **Sender's Anonymity:** Identifying the sender of a valid unsigncrypted message is computationally hard for anyone except the group manager.
- **Ciphertext Anonymity:** The ciphertext reveals no information about who created it nor about whom it is intended to, i.e., the sender's identity is hidden not only to an outsider but also to the receiver.
- **Unlinkability:** No one can tell if two signcryptions were from the same signer or not.
- **Traceability:** The group manager can find the true signer, for any valid verified message.
- **Exculpability:** No one, even the group manager, can signcrypt on the behalf of other group members.
- **Coalition-resistance:** A colluding subset of group members cannot generate valid signcryptions in such a

way that the group manager is unable to link to one of the colluding group members.
- **Unforgeable tracing verification:** The group manager cannot falsely accuse a signer of creating signcryptions he/she did not create.

We refer to [15], [45] for more details.

B. SECURITY MODEL

We mainly focus on sUF, IND and ANON proofs since it is known that the notion of security for a signcryption protocol combines unforgeability of the signature and indistinguishability of the encryption scheme [28], [37], [39], [45]. Moreover, the notion of ciphertext anonymity [45] is also considered since it is an important privacy-enhancing property characterizing our proposal.

1) STRONG EXISTENTIAL UNFORGEABILITY (sUF)

We consider the notion of Strong Existential Unforgeability under adaptive Chosen Message Attack (sUF-CMA) [9], [45]. In an asymmetric settings, the sender and the receiver do not share the same secret key, therefore, the system needs to be protected not only from an outsider but also from an insider. In case of sUF-CMA, the attacker is given the private key of the receiver [45]. This proves that a receiver cannot forge a signcryption ciphertext that should be from the sender.

Therefore, sUF-CMA is defined by using the following game between a Challenger $\mathcal{C}$ and an Adversary $\mathcal{A}$:

Setup: $\mathcal{C}$ runs algorithms `Setup` and `KeyGen` to generate the public system security parameters par, sender's key pair (pk_s, sk_s) and receiver's key pair (pk_r, sk_r). $\mathcal{A}$ is given (par, pk_s, pk_r, sk_r).
Signcryption-Queries: $\mathcal{A}$ requests signcryption of at most q_s messages of its choice $m_1, \ldots, m_{q_s} \in \{0,1\}^*$. $\mathcal{C}$ responds to each query with a ciphertext and a signature $(c_i, \sigma_i) \leftarrow$ `Signcrypt`(par, sk_s, pk_r, m_i) (note that $\mathcal{A}$ does not need to have access to an unsigncryption oracle as it can compute the unsigncryption algorithm itself using sk_r).
Output: $\mathcal{A}$ eventually outputs a pair (c, σ) and wins the game if:

1. $(1, m) \leftarrow$ `Unsigncrypt` $(par, c, \sigma, pk_s, sk_r)$ is a valid signature.
2. (c, σ) was not the output of a signcryption query `Signcrypt` (par, sk_s, pk_r, m_i) during the game.

We define $Adv_{\mathcal{A}}^{sUF}$ to be the probability that the adversary $\mathcal{A}$ wins in the above game, taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{C}$.

Definition 3: A forger $\mathcal{A}$ is said to (t, q_s, ϵ)-break a signcryption scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_s signcryption queries and q_s unsigncryption queries, and $Adv_{\mathcal{A}}^{sUF}$ is at least ϵ. A signcryption scheme is (t, q_s, ϵ)-secure against strongly existentially unforgeable under adaptive chosen message attack if there exists no forger that (t, q_s, ϵ)-breaks it.

Definition 3 follows Boneh proposal (see Definition 1 in [9]) and is modified to work in a signcryption environment [45]. Note that the proposed group signcryption protocol is based on wBB signature [9].

2) INDISTINGUISHABILITY (IND)

We consider the notion of INDistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2) [41], [45]. In an asymmetric settings, the sender and the receiver do not share the same secret keys and, therefore, the system need to be protected not only from an outsider but also from an insider. In case of IND-CCA2, the private key of the sender is given to the attacker [45]. In this way, it is proven that the signcryption scheme protects the confidentiality of the messages even if the sender's secret key is leaked to an attacker.

IND-CCA2 is defined by using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

Setup: $\mathcal{C}$ runs algorithms `Setup` and `KeyGen` to generates the public system security parameters par, sender's key pair (pk_s, sk_s) and receiver's key pair (pk_r, sk_r). $\mathcal{A}$ is given (par, pk_s, sk_s, pk_r).
Queries-1: $\mathcal{A}$ requests unsigncryption of at most $q_{s'}$ ciphertexts $c_1, \ldots, c_{q_{s'}}$, under pk_s and pk_r. $\mathcal{C}$ responds to each query with 1 and a signed message $(1, m_i) \leftarrow$ `Unsigncrypt` $(par, pk_s, sk_r, c_i, \sigma_i)$ if the obtained signed plaintext is valid and with 0 otherwise (note that $\mathcal{A}$ does not need to have access to a signcryption oracle as it can compute the signcyption algorithm using sk_s).
Challenge: $\mathcal{A}$ outputs two equal-length messages m_0' and $m_1' \in \{0,1\}^*$ on which it wishes to be challenged. Then, hidden from $\mathcal{A}$ view, $\mathcal{C}$ chooses $b \leftarrow \{0, 1\}$ and computes the challenge ciphertext $(c_*', \sigma_*) \leftarrow$ `Signcrypt`(par, sk_s, pk_r, m_b')
Queries-2: $\mathcal{A}$ may request at most $q_{s''}$ signcryption and unsigncryption queries as in Queries-1 phase but with the restriction that $\mathcal{A}$ cannot query for c_*'.
Guess: $\mathcal{A}$ produces its guess b' of b. $\mathcal{A}$ is successful if $b' = b$, i.e., the guess is correct.

We define $Adv_{\mathcal{A}}^{IND}$ to be the probability that the adversary $\mathcal{A}$ wins in the above game, and it is defined as

$$Adv_{\mathcal{A}}^{IND} = |2\mathbf{Pr}[b' = b] - 1|$$

Definition 4: An adversary $\mathcal{A}$ is said to $(t, q_s, \mu, m, \epsilon)$-break a signcryption scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most $q_s = q_{s'} + q_{s''}$ signcryption queries and q_s unsigncryption queries, the size of the decryption queries is at most μ bits, the size of the challenge messages m_0' and m_1' is at most m bits, and $Adv_{\mathcal{A}}^{IND}$ is at least ϵ. A signcryption scheme is $(t, q_s, \mu, m, \epsilon)$-secure against indistinguishability under adaptive chosen ciphertext attack if there exists no adversary that $(t, q_s, \mu, m, \epsilon)$-breaks it.

Definition 3 uses 1) the same notation proposed in [9] for consistence purposes, 2) Smart indistinguishability definitions (see Section 3 in [41]), and 3) is slightly modified to work in a signcryption environment [45]. Note that

the proposed group signcryption protocol is based on Gayoso *et al.* encryption scheme [18] which was proven to be secure by Smart [41].

3) CIPHERTEXT ANONYMITY (ANON)

We consider the notion of ciphertext ANONymity under adaptive Chosen Ciphertext Attack (ANON-CCA) [45]. This property is satisfied if ciphertexts reveal no information about who created them nor about whom they are intended to. Therefore, the system needs to be protected from an outsider and ANON-CCA is defined by using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

Setup: $\mathcal{C}$ runs algorithms `Setup` and `KeyGen` to generates the public system security parameters par, sender's key pair (pk_s, sk_s), and two distinct receiver's key pair (pk_{r_0}, sk_{r_0}) and (pk_{r_1}, sk_{r_1}). $\mathcal{A}$ is given par, pk_{r_0} and pk_{r_1}.
Queries-1: $\mathcal{A}$ requests signcryption of at most $q_{s'}$ messages of its choice $m_1, \ldots, m_{q_{s'}} \in \{0,1\}^*$ for the key pairs (pk_{r_0}, sk_{r_0}) and (pk_{r_1}, sk_{r_1}). $\mathcal{C}$ responds to each query with a ciphertext and a signature $(c_{i_j}, \sigma_i) \leftarrow \texttt{Signcrypt}(par, sk_s, pk_{r_j}, m_i)$, where $j = 0, 1$. Then, proceeding adaptively, $\mathcal{A}$ requests unsigncryption of at most $q_{s'}$ ciphertexts $c_1, \ldots, c_{q_{s'}}$, under pk_s and pk_{r_j} with $j = 0, 1$. $\mathcal{C}$ responds to each query with 1 and a signed message $(1, m_i) \leftarrow \texttt{Unsigncrypt}(par, pk_s, sk_{r_j}, c_{i_j}, \sigma_i)$ if the obtained signed plaintext is valid and with 0 otherwise.
Challenge: $\mathcal{A}$ eventually outputs two sender's private keys sk_{s_0} and sk_{s_1}, and a message $m \in \{0,1\}^*$ on which it wishes to be challenged. Then, hidden from $\mathcal{A}$ view, $\mathcal{C}$ chooses $b, d \leftarrow \{0,1\}$ and computes the challenge ciphertext $(c'_*, \sigma_*) \leftarrow \texttt{Signcrypt}(par, sk_{s_b}, pk_{r_d}, m)$.
Queries-2: $\mathcal{A}$ may request at most $q_{s''}$ signcryption and unsigncryption quieries as in Queries-1 phase but with the restriction that $\mathcal{A}$ cannot query for (c'_*, pk_{s_j}), where $j = 0, 1$.
Guess: $\mathcal{A}$ produces its guess b' of b and d' of d. $\mathcal{A}$ is successful if $b' = b$ and $d = d'$, i.e. the guess is correct.

We define $Adv_{\mathcal{A}}^{ANON}$ to be the probability that the adversary $\mathcal{A}$ wins in the above game, and it is defined as

$$Adv_{\mathcal{A}}^{IND} = |4\mathbf{Pr}[(b', d') = (b, d)] - 1|$$

Definition 5: An adversary $\mathcal{A}$ is said to $(t, q_s, \mu, m, \epsilon)$-break a signcryption scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most $q_s = q_{s'} + q_{s''}$ signcryption queries and q_s unsigncryption queries, the size of the decryption queries is at most μ bits, the size of the challenge messages m'_0 and m'_1 is at most m bits, and $Adv_{\mathcal{A}}^{ANON}$ is at least ϵ. A signcryption scheme is $(t, q_s, \mu, m, \epsilon)$-secure against anonymity under adaptive chosen ciphertext attack if there exists no adversary that $(t, q_s, \mu, m, \epsilon)$-breaks it.

Definition 5 uses the same notation proposed in [9] for consistence purposes and 3) is slightly modified to work in a signcryption environment [45].

TABLE 3. Definition of the variables.

Variable	Description
par	Public system parameters
m	Message
sk_m	Manager secret key
sk_i	Group Sender i secret key
δ_i	Group Sender i credential
pk_s	Group senders public key
(pk_r, sk_r)	Receiver public and secret keys
KDF	Key derivation function
SYM	Symmetric encryption scheme
$\mathcal{H}$	Hash functions

V. ARCHITECTURE

Three types of entities interact in our signcryption scheme: a Sender Group Manager, a Sender, and a Receiver. Moreover, a Receiver Group Manager is involved in the multi-receiver scenario.

- **Sender Group Manager (SGM)**, shortly **Manager**: the Sender Group Manager generates system security parameters and cryptographic keys, enrolls new senders and traces malicious ones.
- **Group Sender**, shortly **Sender**: the Sender signcrypts the data and sends them to the receiver.
- **Receiver Group Manager (RGM)**: the Receiver Group Manager generates group public and secret keys, enrolls new receivers, and distributes the decryption keys between them all. RGM is needed only in the multi-receiver scenario.
- **Receiver**: the Receiver receives the signcrypted data, and decrypts and checks the validity of the signature of the plaintext.

Table 3 shows the main variables with their definition used throughout the our scheme. The signcryption scheme consists of the following five algorithms (which are sketched in Figure 1):

- $(par, (pk_s, sk_m), (pk_r, sk_r)) \leftarrow \texttt{Setup}(1^\kappa)$: this algorithm works in two phases. At first, on the input of security parameter κ, the Manager generates and publishes the public system parameters $par = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g, g_2, \mathcal{H}, SYM)$, chooses and publishes the public key shared by all senders pk_s, and chooses the manager's private key sk_m which is kept secret. In particular, $\mathcal{H}$ is a predefined hash function and SYM is a predefined secure symmetric encryption scheme. At second, on the input of the public system parameters par, the Receiver generates a secret key sk_r and publishes the receiver's public key pk_r.
- $(\delta_i, rd) \leftarrow \texttt{Join}(par, sk_i, sk_m)$: on the input of the public system parameters par, the manager's private key sk_m and the sender's secret key sk_i, this protocol outputs the sender group member credential δ_i and the revocation database rd. The `Join` algorithm is run as an interactive protocol between the Manager and the Sender.
- $(\sigma, c) \leftarrow \texttt{Signcrypt}(par, m, sk_i, \delta_i, pk_r)$: on the input of the public system parameters par, the message m,

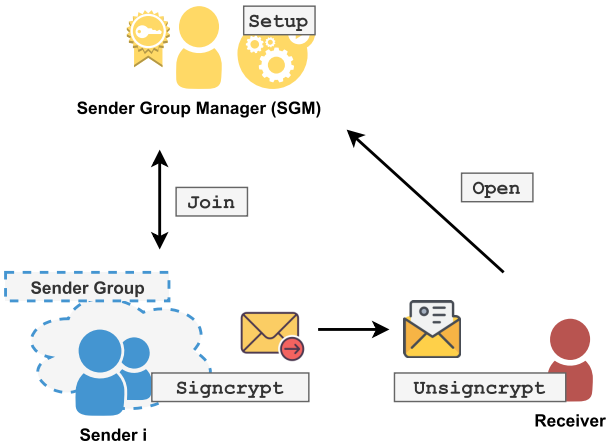


FIGURE 1. Group signcryption scheme architecture.

the receiver's public key pk_r, the sender's key sk_i and the credential δ_i, the `Signcrypt` algorithm outputs the signature σ on the message m, and the ciphertext c of m. This algorithm is run by the Sender.

- $(m, 0/1) \leftarrow$ `Unsigncrypt`$(par, sk_r, pk_s, c, \sigma)$: on the input of the public system parameters par, receiver's private key sk_r, the public key pk_s, the ciphertext c and the signature σ, the `Unsigncrypt` algorithm decrypts the ciphertext c and returns the message m, then verifies the signature σ and returns 1 and the message m iff the signature is valid and 0 otherwise.
  This algorithm is run by the Receiver.
- $(pk_i) \leftarrow$ `Open`(rd, σ): on the input of the manager's revocation database rd and a signature σ, the algorithm outputs the sender's public key pk_i which is linkable with sender's identity. The `Open` algorithm is run by the Manager.

A. CRYPTOGRAPHIC PRIMITIVES INTEGRATION

We use several cryptographic primitives in the following parts of the scheme:

- **Group signature (GS):** It allows a signer to generate anonymous signatures on messages. In particular, we use the lightweight group signature presented by Hajny *et al.* [19] which is based on the weak Boneh-Boyen (wBB) signature [9].
- **Encryption scheme:** Integration of ECIES scheme [18] allows us to establish a session key and encrypt the signer's data.
- **Proofs of Knowledge (PK):** Thanks to PK, the Sender can prove the possession of its secret key and therefore generate the group signature within the `Signcrypt` algorithm. Furthermore, PK is also used to prove the possession of secret keys of both Manager and Sender within the `Join` phase. To do so, we use the Schnorr protocol [42].
- **Homomorphic encryption (HE):** We use the Paillier encryption scheme [36] to securely compute the group sender credential as shown in [7]. HE is run in the `Join` phase between the Manager and the Sender. HE ensures that no secret values of both parties, which are needed for forging the sender credential, are shown to the counterparty.
- **Group key agreement (GKA):** The BCEP group key agreement protocol [11] is used in the `Join` phase to generate and distribute the decryption key in the receiver group. This protocol is applied in the multi-receiver scenario.

VI. PROPOSED SCHEME

In this section, our group-to-one signcryption scheme is presented in detail. This scheme allows any sender from a group to signcrypt a message in the group's behalf and send it to one receiver. Regarding the group signature scheme, we slightly modify the original group signature scheme proposed by Hajny *et al.* [19]. In our variant, we employ the Paillier encryption [36] to provide exculpability property as shown by Belenkiy *et al.* [7]. This property was not provided in the original scheme and guarantees that the group manager cannot sign on the behalf of other group members. Moreover, the group signature scheme [19] uses Weak Boneh-Boyen signature [9] and its efficient proof of knowledge [13] to sign messages. The wBB signatures were proven to be existentially unforgeable against a weak (non-adaptive) chosen message attack under the p-SDH assumption [9]. For the encryption, we take inspiration from the ECIES scheme proposed by Gayoso *et al.* [18]. In our proposal, a Key Derivation Function (KDF) is needed. In particular, KDF is defined as $KDF : \mathbb{Z}_q \times \mathbb{Z}_q \rightarrow \{0,1\}^\lambda$, where λ is the bitlength of a SYM key. The concrete algorithms can be found below.

A. `SETUP` ALGORITHM

The `Setup` algorithm consists of two phases:

`Setup_SGM`: The Manager performs the following steps:

1) Choose a bilinear map $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of the same prime order q, g a generator of $\mathbb{G}_1$, and g_2 a generator of $\mathbb{G}_2$.
2) Define a secure hash function $\mathcal{H} : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^{|m|} \rightarrow \mathbb{Z}_q$, where $|m|$ is the length of the plaintext message.
3) Choose a symmetric encryption scheme $SYM = (Enc_{SYM}, Dec_{SYM})$.
4) Choose $sk_m \xleftarrow{s} \mathbb{Z}_q$ as the manager's private key, and set $pk_s = g_2^{sk_m}$ as the sender's group public key.
5) Generate an RSA-modulus $\mathbf{n}$ of size at least $2^{3\kappa}q^2$, where κ is a security parameter. Furthermore, let $\mathbf{h} = \mathbf{n} + 1$ and $\mathbf{g}$ be an element of the order $\phi(\mathbf{n})$ mod $\mathbf{n}^2$.
6) For simplicity of this exposition, we assume the existence of an RSA modulus $\mathfrak{n}$ such that neither the Sender nor the Manager knows its factors. This modulus can be provided by a Trusted Third Party (TTP). Alternatively, the Sender and the Manager can generate their own modules and use them in the protocol as proposed

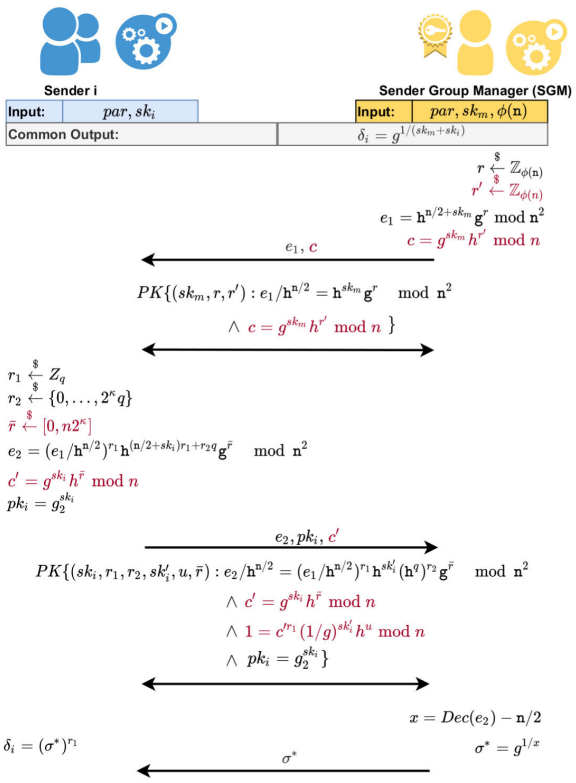


FIGURE 2. CS notation of the `Join` algorithm.

in [4]. Furthermore, let $\mathfrak{h}$ and $\mathfrak{g}$ be two elements in $\mathbb{Z}_{\mathfrak{n}}^*$ such that $\log_{\mathfrak{g}} \mathfrak{h}$ is unknown and $\mathfrak{g} \in \langle \mathfrak{h} \rangle$.

7) Publish the public system security parameters $par = (pk_s, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g, g_2, \mathcal{H}, SYM, \mathbf{n}, \mathbf{h}, \mathbf{g}, \mathfrak{n}, \mathfrak{g}, \mathfrak{h})$ and keep $(sk_m, \phi(\mathbf{n}))$ secret.

`Setup_R`: This algorithm is run by the Receiver. With public system parameters par, the Receiver performs the following steps:

1) Randomly choose a private key $sk_r \xleftarrow{\$} \mathbb{Z}_q$.
2) Compute and publish its public key $pk_r = g^{sk_r}$.

B. `JOIN` ALGORITHM

This algorithm is run by the Sender and the Manager. Figure 2 shows the `Join` algorithm in Camenisch and Stadler (CS) notation, where the secure two-party computation of the Sender i credential δ_i takes place.

This algorithm allows computing $\delta_i = g^{1/(sk_m+sk_i)}$ without that the Manager reveals it private key sk_m and the Sender its secret key sk_i. With public system security parameters par, Manager's secret key sk_m and Sender's secret key sk_i as input, the Manager and the Sender perform the following steps (see in Appendix E.4 of [7] for more details):

1) the Manager computes

$$e_1 = \mathbf{h}^{\mathbf{n}/2+sk_m} \mathbf{g}^r \mod \mathbf{n}^2,$$

where $r \xleftarrow{\$} Z_{\phi(\mathbf{n})}$,

$$\mathfrak{c} = \mathfrak{g}^{sk_m} \mathfrak{h}^{r'} \mod \mathfrak{n},$$

where $r' \xleftarrow{\$} Z_{\phi(\mathfrak{n})}$, and sends $(e_1, \mathfrak{c})$ to the Sender,

2) the Manager and the Sender run the following PK protocol with each other:

$$PK\{(sk_m, r, r') : e_1/\mathbf{h}^{\mathbf{n}/2} = \mathbf{h}^{sk_m} \mathbf{g}^r \mod \mathbf{n}^2 \wedge \mathfrak{c} = \mathfrak{g}^{sk_m} \mathfrak{h}^{r'} \mod \mathfrak{n}\}$$

3) the Sender chooses $r_1 \xleftarrow{\$} \mathbb{Z}_q$ and $r_2 \xleftarrow{\$} \{0, \ldots, 2^\kappa q\}$, computes

$$e_2 = (e_1/\mathbf{h}^{\mathbf{n}/2})^{r_1} \mathbf{h}^{(\mathbf{n}/2+sk_i)r_1+r_2 q} \bar{\mathbf{g}}^{\bar{r}} \mod \mathbf{n}^2$$

and the commitment

$$\mathfrak{c}' = \mathfrak{g}^{sk_i} \mathfrak{h}^{\bar{r}} \mod \mathfrak{n},$$

with $\bar{r} \xleftarrow{\$} [0, \mathfrak{n}2^\kappa]$, his/her public key $pk_i = g_2^{sk_i}$, and sends $(e_2, pk_i, \mathfrak{c}')$ to the Manager,

4) the Manager and the Sender run the following protocol with each other:

$$PK\{(sk_i, r_1, r_2, sk_i', u, \bar{r}) : e_2/\mathbf{h}^{\mathbf{n}/2} = (e_1/\mathbf{h}^{\mathbf{n}/2})^{r_1} \mathbf{h}^{sk_i'} (\mathbf{h}^q)^{r_2} \mathbf{g}^{\bar{r}} \mod \mathbf{n}^2 \wedge \mathfrak{c}' = \mathfrak{g}^{sk_i} \mathfrak{h}^{\bar{r}} \mod \mathfrak{n} \wedge 1 = \mathfrak{c}'^{r_1} (1/\mathfrak{g})^{sk_i'} \mathfrak{h}^u \mod \mathfrak{n} \wedge pk_i = g_2^{sk_i}\},$$

where $sk_i' = sk_i r_1$ and $u = -\bar{r} r_1$.

5) The Manager decrypts $x = Dec(e_2) - \mathbf{n}/2$, computes $\sigma^* = g^{1/x}$ and sends it to the sender.
6) The sender computes $\delta_i = (\sigma^*)^{r_1}$ and verifies that it is a correct signature on sk_i, i.e. $\delta_i = g^{\frac{1}{sk_m+sk_i}}$ holds.

C. `SIGNCRYPT` ALGORITHM

With the public system security parameters par, the message m, the receiver's public key pk_r, the sender's secret key sk_i and the credential δ_i, the Sender i generates the ciphertext c and the signature σ of m as follows:

1) Randomly choose randomizers $r, \rho_r, \rho_{sk_i} \xleftarrow{\$} \mathbb{Z}_q$, and compute $g' = g^r$ and $j = pk_r^r$.
2) Generate a symmetric key $k_{\text{enc}} = KDF(j)$.
3) Encrypt the message $c = Enc_{SYM}(m, k_{\text{enc}})$ by the symmetric encryption scheme.
4) Compute the values $\delta_i' = \delta_i^r$, $\bar{\delta}_i' = \delta_i'^{-sk_i}$, $t = \delta_i'^{\rho_{sk_i}} g^{\rho_r}$, $e = \mathcal{H}(g', \delta_i', \bar{\delta}_i, t, m)$, $s_r = \rho_r - er$, and $s_{sk_i} = \rho_{sk_i} - esk_i$ necessary to generate the signature proof of knowledge $\pi = \{(sk_i, r) : \bar{\delta}_i' = \delta_i'^{-sk_i} \wedge g' = g^r\}$.
5) Send (σ, c) to the Receiver, where $\sigma = (g', \delta_i', \bar{\delta}_i, \pi)$.

D. `UNSIGNCRYPT` ALGORITHM

When receiving (σ, c), the Receiver decrypts the message and, then, verifies the signature as follows:

1) DECRYPT

1) Compute $j' = g'^{sk_r}$ and $k_{\text{enc}}' = KDF(j')$.
2) Recover the message $m = Dec_{SYM}(c, k_{\text{enc}}')$.

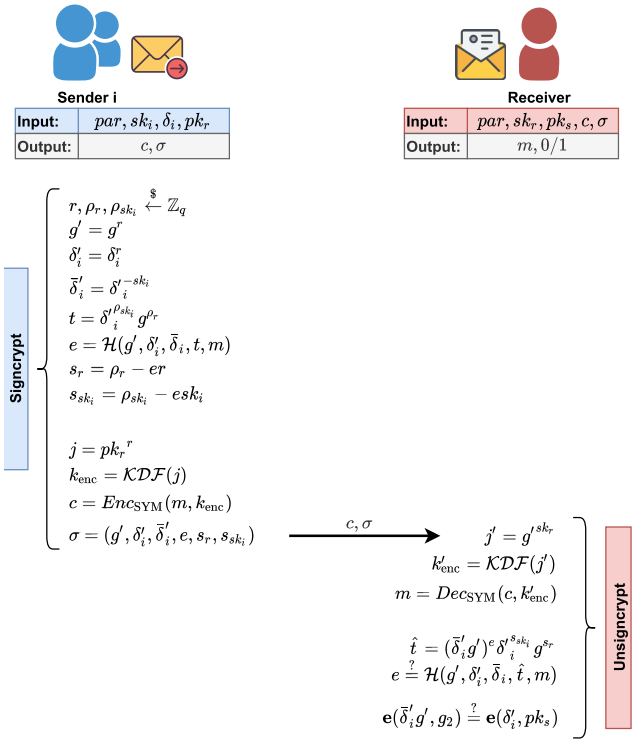


FIGURE 3. Full notation of `Signcrypt` and `Unsigncrypt` algorithms.

2) VERIFY

The receiver computes $\hat{t} = (\bar{\delta}'_i g')^e \delta'^{s_{sk_i}}_i g^{s_r}$, and checks if the equations $e \stackrel{?}{=} \mathcal{H}(g', \delta'_i, \bar{\delta}_i, \hat{t}, m)$ and $\mathbf{e}(\bar{\delta}'_i g', g_2) \stackrel{?}{=} \mathbf{e}(\delta'_i, pk_s)$ hold.

The full notation of the `Signcrypt` and the `Unsigncrypt` algorithms is depicted in Figure 3.

E. OPENING ALGORITHM

This algorithm allows the Manager to open the signature and track the Signer. With the manager's revocation database rd and the signature σ, the Manager checks if the equation $\mathbf{e}(\delta', pk_j) \stackrel{?}{=} \mathbf{e}(\bar{\delta}_i, g_2)$ holds for any of pk_j in its database, where j in $\{0, \ldots n\}$ and n is the number of sender group members. If there exists an pk_j for which this equation holds, pk_j is linked with the sender's real identity.

F. REVOKE ALGORITHM

Our scheme is compatible with standard revocation algorithms for randomized proofs, see [13] for more details.

VII. MULTI-RECEIVER SCENARIO

The proposed signcryption scheme can be easily adapted to a multi-receiver scenario. A sketch of the multi-receiver scenario is depicted in Figure 4. In this case, the Sender signcrypts the message and sends it to a group of receivers instead of one receiver. Therefore, we need to create a group of authorized receivers and a way to securely distribute the group secret key (unsigncryption key) to all group members.

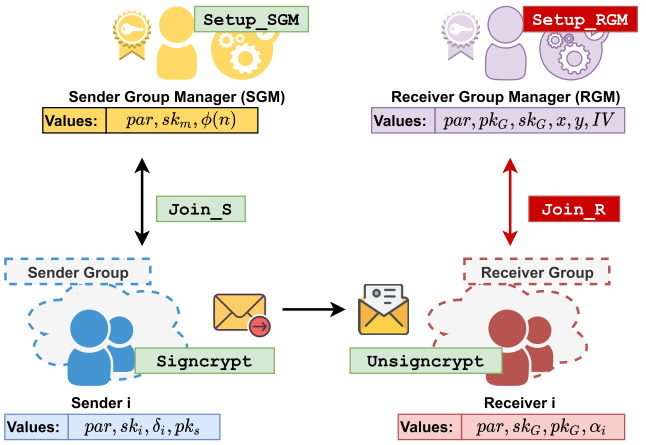


FIGURE 4. Multi-receiver group signcryption scenario.

To do so, we adopt the solution of Kwak *et al.* [23] which involves the BCEP [11] protocol to distribute the unsigncryption key to the targeted group of receivers.

In particular, `Setup_SGM`, `Join`, `Signcrypt`, `Unsigncrypt`, and `Open` algorithms remain unchanged. In fact, these algorithms either belong to the group of senders or receive the same input as in the group-to-one scenario. On the contrary, the group of receivers requires the addition of `Setup-RGM` and `Join-R` algorithms to `Setup` and `Join` algorithms, respectively. The main task of these new protocols is to distribute the group secret key between the members of the receiver group.

The concrete algorithms of our multi-receiver scheme can be found below.

A. SETUP ALGORITHM

The `Setup` algorithm consists of two phases:

`Setup_SGM`: This algorithm is run by the Manager. The algorithm is equal to Algorithm `Setup_SGM` in Section VI.

`Setup_RGM`: RGM performs the following steps:

- on the input of the system public parameters par, RGM chooses random $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $y = g^x$,
- then computes $sk_G = \mathcal{H}(IV, y)$, where IV is an initial vector. The value x is the manager's secret key, sk_G is the group secret key, while $pk_G = g^{sk_G}$ is the group public key.

B. JOIN_S ALGORITHM

This algorithm is equal to Algorithm `Join` in Section VI.

C. JOIN_R ALGORITHM

A Receiver belonging to the authorized group computes $y_i = g^{x_i}$, where $x_i \xleftarrow{\$} \mathbb{Z}_q$. Then it sends y_i and the signature σ_i on y_i to RGM. Note that σ_i is generated by a secure signature scheme such as either RSA or Elliptic Curve Digital Signature Algorithm (ECDSA). If the Receiver belongs also to the sender group, and if it is permitted by the system, then the Receiver can signcrypt the value y_i and send it to RGM.

The RGM checks whether the signature is valid or not. If it is valid, then RGM computes the member's key $\alpha_i = y_i^x$ and regenerates the group secret key $sk_G = \mathcal{H}(IV, y, \alpha_1, \cdots \alpha_n)$, where n is the number of group members. The RGM sends $(sk_{Gi}, IV, y, \sigma_{RGM})$ to all members, where $sk_{Gi} = sk_G \oplus \mathcal{H}(IV, \alpha_i)$ and σ_{RGM} is a signature on the triplet (sk_{Gi}, IV, y). Each group member then can verify the signature σ_{RGM}, compute $\alpha_i = y^{x_i}$ and recover the shared group secret key sk_G. In this way, the RGM can securely share the group secret key sk_G with all group members, while the $pk_G = g^{sk_G}$ is made public.

VIII. SECURITY ANALYSIS

In this section, we prove that the proposed scheme satisfies all group signcryption security features listed in Section IV-A. Firstly, we focus on proving that our scheme satisfies correctness, confidentiality (IND-CCA2), unforgeability (sUF-CMA) and ciphertext anonymity (ANON-CCA). These are the main features of any signcryption protocol as shown in [45]. Then we remark that our group signcryption scheme also guarantees sender anonymity, unlinkability, traceability, and coalition-resistance. Finally, we show that our scheme provides exculpability and unforgeable tracing verification properties.

A. CORRECTNESS

Theorem 1: The decryption process in Section VI-D is correct.

Proof: Since a symmetric cryptographic scheme is used to encrypt the message, at first we show that the receiver can reconstruct the sender's key. In fact,

$$j' = g'^{sk_r} = (g^r)^{sk_r} = pk_r^r = (g^{sk_r})^r = j$$
$$k'_{\text{enc}} = KDF(j') = KDF(j) = k_{\text{enc}}$$

and, therefore, $Dec_{SYM}(c, k'_{\text{enc}}) = Dec_{SYM}(c, k_{\text{enc}}) = m$. Accordingly, the decryption process is correct. □

Theorem 2: The verification process in Section VI-D is correct.

Proof: See Appendix A for proof. □

B. STRONG EXISTENTIAL UNFORGEABILITY (sUF)

Boneh and Boyen [9] prove that the wBB signature scheme is strong existentially unforgeable against an adaptive chosen message attack under the p-SDH assumption. The sUF-CMA of our scheme follows from the unforgeability of wBB signature (see Lemma 9 in [9]) and uses the same proof technique. We consider an attacker who makes up to q_s adaptive signcryption and unsigncryption queries, and reduce the forgery to the resolution of a random p-SDH instance for $p = q_s$.

Theorem 3: Suppose the (p, t', ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then the signcryption scheme proposed in Section VI is (t, q_s, ϵ)-secure against existential forgery under adaptive chosen message attack with

$$q_s \leq p \text{ and } t \leq t' - \Theta(pT)$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{Z}_q$.

Proof: See Appendix B for proof. □

C. INDISTINGUISHABILITY (IND)

Smart [41] analyzes the security of a generic ECIES scheme, in particular, he focuses on the indistinguishability under adaptive chosen ciphertext attacks. The IND-CCA2 of our scheme follows the same proof technique of ECIES indistinguishability (see Section 4 in [41]). We consider an attacker who makes up to q_s adaptive signcryption and unsigncryption queries.

Lemma 4: For any adversary $\mathcal{A}$ running in time t and making at most $q_s = q_{s'} + q_{s''}$ unsigncryption queries, the advantage of winning the IND-CCA2 game is

$$Adv_{\mathcal{A}}^{IND}(t, q_s) \leq 2Adv_{\mathcal{B}}^{DDH}(t', q) + 2Adv_{\mathcal{B}}^{SDH}(t'', p) + Adv_{\mathcal{B}}^{SYM}(t''', |\kappa|)$$

where

- *$Adv_{\mathcal{B}}^{DDH}(t', q)$ is the maximal probability of solving the DDH assumption in time t'.*
- *$Adv_{\mathcal{B}}^{SDH}(t'', p)$ is the maximal probability of solving the SDH assumption in time t''.*
- *$Adv_{\mathcal{B}}^{SYM}(t''', |\kappa|) = 2\boldsymbol{Pr}[b' = b] - 1$ is the maximal advantage of any adversary mounting a chosen plaintext attack on SYM in time t''' with key size $|\kappa|$.*

Proof: See Appendix C for proof. □

Theorem 5: Suppose the (q, t', ϵ')-DDH and (p, t'', ϵ'')-SDH assumptions hold in $\mathbb{G}_1$ and $(\mathbb{G}_1, \mathbb{G}_2)$, respectively. Then the signcryption scheme proposed in Section VI is (t, q_s, ϵ)-secure against indistinguishability under adaptive chosen ciphertext attacks with

$$q_s \leq p \text{ and } t \leq 2t''' + t'' + \frac{2q_s^2}{q} - \Theta(q_{s''}T)$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{Z}_q$.

Proof: We prove this theorem using Lemma 4 which allows bounding the advantage of winning the IND-CCA2 game. Since $Adv_{\mathcal{B}}^{DDH} = \frac{q_s^2}{p}$, where q_s is the number of queries that $\mathcal{A}$ makes (as proven by Shoup [43], Theorem 4), the claimed bound is obvious by construction.

It is important to notice that our proof theoretically works for any *SYM* and *KDF* schemes which are separately proven to be secure. In fact, the security of our sygncryption scheme relies on the security of chosen *SYM* and *KDF* schemes. For instance, Smart [41] suggests using SHA-1 as the *KDF* function. □

D. CIPHERTEXT ANONYMITY (ANON)

Ciphertext anonymity property is satisfied if ciphertexts reveal no information about who created them nor about whom they are intended to [45]. In particular, this exactly covers that sender's and receiver's identities are hidden to outsiders.

We consider an attacker who makes up to q_s adaptive signcryption and unsigncryption queries.

Lemma 6: For any adversary $\mathcal{A}$ running in time t and making at most $q_s = q_{s'} + q_{s''}$ signcryption and unsigncryption queries, the advantage of winning the ANON-CCA game is

$$Adv_{\mathcal{A}}^{ANON}(t, q_s) \leq 4Adv_{\mathcal{B}}^{DDH}(t', q) + 4Adv_{\mathcal{B}}^{SDH}(t'', p) + 2Adv_{\mathcal{B}}^{SYM}(t''', |\kappa|) + \frac{1}{4}$$

where $Adv_{\mathcal{B}}^{DDH}(t', q)$, $Adv_{\mathcal{B}}^{SDH}(t'', p)$ and $Adv_{\mathcal{B}}^{SYM}(t''', |\kappa|)$ are defined as in Lemma 4.

Proof: See Appendix D for proof. □

Theorem 7: Suppose the (q, t', ϵ')-DDH and (p, t'', ϵ'')-SDH assumptions hold in $\mathbb{G}_1$ and $(\mathbb{G}_1, \mathbb{G}_2)$, respectively. Then the signcryption scheme proposed in Section VI is (t, q_s, ϵ)-secure against anonymity under adaptive chosen ciphertext attacks with

$$q_s \leq p \text{ and } t \leq 4t''' + 2t'' + \frac{4q_s^2}{q} + \frac{1}{4} - \Theta(q_{s''}T)$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{Z}_q$.

Proof: We prove this theorem using Lemma 6 which allows bounding the advantage of winning the ANON-CCA game. Since $Adv_{\mathcal{B}}^{DDH} = \frac{q_s^2}{p}$, where q_s is the number of queries that $\mathcal{A}$ makes (as proven by Shoup [43], Theorem 4), the claimed bound is obvious by construction. □

E. SENDER'S ANONYMITY, UNLINKABILITY, TRACEABILITY, AND COALITION-RESISTANCE

It is important to notice that sender's anonymity, unlinkability, traceability, and coalition-resistance are privacy-enhancing features achieved thanks to the usage of our previously proposed group signature [19].

This group signature is integrated with the zero-knowledge proofs, i.e., the Sender i proves the knowledge of its secret key sk_i and the credential δ_i. In particular, without the knowledge of the secret key sk_i and a randomizer r, these proofs are provably *unlinkable*. Moreover, *traceability* is guaranteed since the Manager knows the senders' public keys $pk_j = g_2^{-sk_j}$, for $j \in \{1, \ldots n\}$ where n is the number of senders. Therefore, the Manager is able to efficiently link all proofs by computing $\mathbf{e}(\delta_i', pk_j) \stackrel{?}{=} \mathbf{e}(\bar{\delta_i}, g_2)$. Regarding *sender's anonymity*, any sender can sign the message on behalf of a group, therefore, its identity is hidden inside the group. In order to break the *coalition-resistance* property, a subset of senders needs to generate a new valid group sender credential $\delta_i = g^{1/(sk_m + new)}$ for a secret key *new* without the knowledge of Manager secret key sk_m and with *new* different from sk_i for any Sender i in the colluding group. This is equivalent to solve p-SDH problem. We refer to [13] for more details.

F. EXCULPABILITY AND UNFORGEABLE TRACING VERIFICATION

The *exculpability* is guaranteed by NIZKPK scheme [7]. The NIZKPK allows to generate the secret group member credential $\delta_i = g^{1/(sk_m + sk_i)}$ for a Sender i without disclosing the sender's secret key sk_i and its credential δ_i. In particular, without the knowledge of sk_i and δ_i, no one, neither the Manager, can generate signcrypted messages on the behalf of any Sender i. In case of *unforgeable tracing verification*, `Opening` algorithm guarantees that the Manager cannot falsely accuse a signer of creating signcryption that it did not create. On the input of the signer's proof $(\delta_i', \bar{\delta_i})$, public system parameter g_2, and sender's public key from Manager's revocation database $pk_j \leftarrow rd$, everyone can verify whether the following equation holds:

$$e(\delta_i', pk_j) \stackrel{?}{=} e(\bar{\delta_i}, g_2).$$

IX. APPLICATION

In this section, we present two use cases: (A) deduplication of big data in cloud computing and (B) anonymous statistical survey of attributes. Note that our many-to-one group signcryption scheme is suitable for Use case (A) while our multi-receiver group signature for Use case (B). See Sections VI and VII, respectively, for more details.

A. DEDUPLICATION OF BIG DATA IN CLOUD COMPUTING

The cloud is fast becoming a suitable strategy in the big data context. The 2021 State of the Cloud Survey [17] estimated that 92 percent of enterprises had either a multi-cloud strategy or a hybrid strategy. Data deduplication is a process that allows controlling the growth of data on the cloud by eliminating duplicate copies. Cho and Toshiba [15] propose a verifiable hash convergent group signcryption which requires the involvement of a group signcryption scheme in the data deduplication process. In their proposal, a group of users is able to eliminate redundant encrypted data owned by different users.

Our scheme can be also adapted to work in this scenario and allows any user to anonymously upload and download encrypted data. Whereas Cho and Toshiba considered a multi-receiver signcryption scheme, we think that a many-to-one group signcryption (presented in Section VI) is more suitable for this application. Our scheme needs the involvement of a Hash Convergent Encryption (HCE). HCE allows data encrypted by different users to generate the same ciphertext. We consider Bellare-Keelveedhi-Ristenpart HCE algorithm [8] following Cho and Toshiba proposal [15]. In an HCE, the message is encrypted with a message-derived key k. This key is the hash of the message m and a public parameter p. The message m is then encrypted $\gamma = \text{Enc}(p, k, m)$ and a *tag* is created from a tag generation algorithm $t = T(\gamma)$. The tag is used to check whether the deduplicated file is fake or not. The message m can be recovered through the decryption process $m = \text{Dec}(k, c)$.

The participants of this system are the Group Manager, the User, and the Server. Note that the Group Manager, the User, and the Server take the role of the Manager, the Group Sender, and the Receiver in our scheme. In this case, the Server can verify the users' ownership of the ciphertext, i.e. it can partially unsigncrypt the ciphertext.

- **Setup**: the Group Manager of group G_a initiates `Seput_SGM` algorithm and establishes the public parameters par, the group public key pk_s, and its secret key sk_m. Then the Server initiates `Seput_R` and establishes its public pk_r and secret sk_r keys. See Section VI for more details.
- **Join**: the User i with the Group Manager runs `Join_S` algorithm to join group G_a.
- **Upload protocol**: given a file f, the User i runs HCE scheme which generates a ciphertext γ and a tag $t = T(\gamma)$. On the input message γ, the User i runs the `Signcrypt` algorithm that outputs a ciphertext c and a signature σ. The user then uploads (c, t, σ) to the Server which checks the validity of the file and the signature by running the `Unsigncrypt` protocol and, if σ and t are valid, obtains the ciphertext γ. If γ is already stored in the cloud, it adds σ to the existing file, otherwise it stores (γ, t, σ).
- **Download protocol**: when the User i wants to download ciphertext γ from the Server, it sends a download request to the Server. The request consists in the file name, σ and t. The Server checks the validity and ownership of the file and if the verification is valid, return γ to the User i which decrypts it and recover the file f.

Due to the confidentiality, anonymity, and unlinkability of the group signcryption scheme, the Server obtains no information beyond the stored ciphertext γ.

B. ANONYMOUS STATISTICAL SURVEY OF ATTRIBUTES

A group signcryption protocol is a suitable candidate to perform an anonymous statistical survey of attributes [23]. In this kind of surveys [34], [35], a service provider wants to collect users' personal information attributes such as gender, age, and job. In particular, the service provider has interest in running statistics on these sensitive data for marketing purposes. On the other hand, users desire to use the service anonymously. In fact, disclosing their personal information may enable the service provider to recover their identity.

The participants in this system are an attribute authority, users, a service provider and trustees. It is assumed that the attribute authority is a TTP that can assure the validity of the users' encrypted attributes. Note that the attribute authority, users, and trustees are respectively SGM, Senders, and Receivers in our group signcryption scheme. Therefore, the survey runs as follows:

- **Setup**: the parameters of the group signcryption scheme are set up through the `Setup` algorithm of Section VII by the attribute authority.
- **Registration**: to join the system, a user conducts the `Join_S` protocol with the attribute authority, where the user joins the group based on a corresponding attribute value. Then the trustees run the `Join_R` algorithm.
- **Offer**: during the service, the user sends their signcrypted attribute (i.e., its encrypted group ID) to the service provider for decryption by a certain trustee. The `Sincrypt` algorithm is used in this step. Users select one trustee and warn the service provider with which trustee is designated.
- **Generate**: the service provider gives the trustees the collected signcryptions. The trustees decrypt the ciphertexts to reveal the group IDs and then verify the signatures. The revealed groups indicate the statistics of the attributes. The `Unsincrypt` algorithm is used in this step.

Due to the confidentiality, anonymity, and unlinkability of the group signcryption scheme, the service provider obtains no information beyond the statistics. The correctness of the statistics is guaranteed by the unforgeability of the scheme.

TABLE 4. Complexity comparison of current group signcryption schemes (`Signcrypt` and `Unsigncrypt` algorithms).

	Kwak [23]	Our Scheme
Security Assumption	RSA	ECDL
`Signcrypt`	15 × EXP	6 × EXP
`Unsigncrypt`	11 × EXP	4 × EXP 2 × PAIR

Note: EXP - exponentiation, PAIR - pairing

X. COMPARISON

In this section, we compare the efficiency of our scheme with Kwak *et al.* proposal [23]. As shown in Tables 1 and 2, Kwak *et al.* scheme is the only provable-secure scheme in addition to our achieving sender and ciphertext anonymity. In Table 4, the number of exponentiations and pairings are depicted. Our scheme is more efficient than Kwak's scheme since their scheme performs ca. 3× more exponentiations than our scheme. This is due to the fact that Kwak's scheme is 1) based on the sign-then-encrypt approach, and 2) the underlying operations are run over RSA group, which is significantly larger than the EC group. Furthermore, the RSA construction of Kwak's scheme is less efficient and less practical on constrained devices in the IoT environment. These devices have limited memory and computational power, and therefore, multiplicative groups, such as RSA, are practically ineffective on these devices. On the contrary, the additive groups over elliptic curves are currently dominant. In fact, many of these constrained devices support only 3072-bit RSA which is equivalently strong to 256-bit EC while others do not support RSA at all. In contrast to Kwak's scheme, our scheme requires two operations of bilinear pairing in `Unsigncrypt` protocol. However, considering the higher computational power of the Receiver, the impact on efficiency is minimal, see Section XI for more details.

XI. EXPERIMENTAL RESULTS

This section provides the whole protocol implementation and the implementation aspects discussion. Current IoT net-

TABLE 5. Technical specification of tested smart cards.

	J3D081	Sm@rtCafe6	ZC7.6	ML4	ML3
MCU	P5CD081	P5CD081	–	SC23Z018	SLE78CLXPM
OS	Java Card	Java Card	Basic Card	MultOS	MultOS
Version	3.0.1	3.0.1	ZC7	4.3.1	4.3.1
ROM	264 KB	264 KB	–	252 KB	280 KB
EEPROM	80 KB	80 KB	72 KB	18 KB	96 KB
RAM	6 KB	6 KB	4.3 KB	1.75 KB	2 KB

TABLE 6. Cryptographic support on tested smart cards.

Algorithm	J3D081	Sm@rtCafe6	ZC7.6	ML4	ML3
SHA1	✓	✓	✓	✓	✓
SHA256	✓	✓	✓	✓	✓
3DES	✓	✓	✓	✓	✓
AES256	✓	✓	✓	✗	✓
mod	✗	✗	✓	✓	✓
modMul	✗	✗	✓	✓	✓
modExp	✓	✓	✓	✓	✓
ecAdd	✓	✗	✓	✓	✗
ecMul	✓	✗	✓	✓	✗
Pair	✗	✗	✗	✗	✗

Note: ✓– algorithm is supported, ✓– algorithm is supported through a special API (e.g. NXP JCOP) or another function (e.g. RSA encryption), ✗ – algorithm is not supported.

works consist of many resource-constrained devices with limited computational and storage capabilities. In order to cover the vast majority of possible use cases, we decided to employ these devices in our testing scenario. The main purpose is to demonstrate the efficiency and the practical potential of our scheme. In particular, we consider ARM-platform (Raspberry Pi) and smart card platforms (Java Card & MultOS). Their specifications are described in Sections XI-A and XI-B, while the testing scenario and evaluations are presented in Section XI-C.

A. SMART CARD SELECTION

Smart cards (SCs) are closed platforms. This means that it is not usually possible to upgrade cryptographic libraries on the card. SC cryptographic support differs according to: 1) the SC platform (e.g., Java Card, MultOS and Basic Card), 2) the version of the operating system, and 3) the SC implementation itself.

For our tests, the newest cards in the market (for each card platform one representative) were selected and their HW/SW properties and cryptographic support were compared. The technical specification of tested SCs is shown in Table 5. Current SCs usually have only 8-bit, 16-bit (or 32-bit in really special cases) processors, and small Random Access Memory (RAM) and Electrically Erasable Programmable Read-Only Memory (EEPROM). These limited resources make the development of novel cryptographic protocols very difficult. On the other hand, SCs are equipped with a co-processor, which allows developers to accelerate specific cryptographic operations and algorithms.

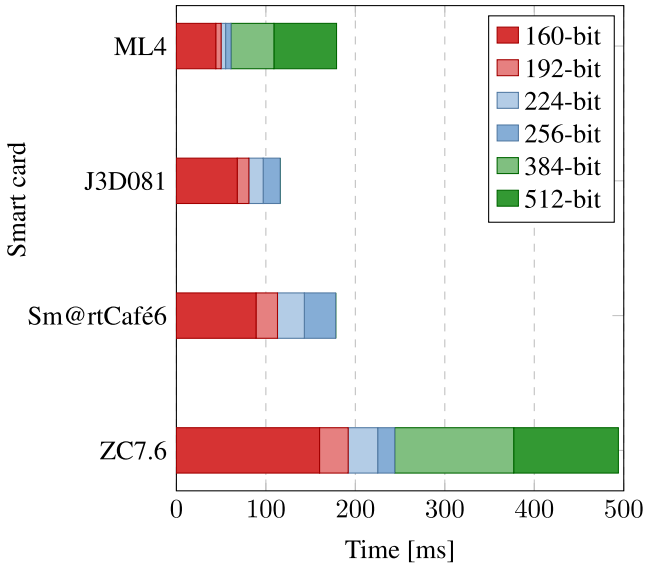


FIGURE 5. Efficiency of `ecMul` operation on different smart card platforms.

Note that our proposal requires 1) a symmetric encryption algorithm to encrypt data, and 2) algebraic operations over finite field and a secure hash algorithm to generate a signature. These simple requirements are not of easy support for current SCs. The cryptographic support in accordance with our signcryption scheme requirements is shown in Table 6. It is important to note that there is no one smart card platform that supports bilinear pairing operations nowadays. In particular, MultOS and Basic Cards are the only platforms which allow accessing to modular and elliptic curve operations.

EC support and speed are crucial for our implementation, and therefore we compared the speed of individual SC platforms. Figure 5 depicts the EC scalar multiplication `ecMul` (which is the most computationally demanding operation of `Signcrypt` protocol) cost for Brainpool curves for different elliptic curve sizes. MultOS (ML4) card is 75% faster than Basic card (ZC7.6) and 35% faster than the fastest Java Card (J3D081). Sm@rtCafe implementation shows a bit worse results than JCOP SC implementation.

Furthermore, we also provided benchmarks of the employed cryptographic algorithms. The SHA-1 algorithm is used for creating non-interactive proof of knowledge (signing part) and as a part of the key derivation function *KDF* for key establishment (encryption part). We use Triple Data Encryption Standard (3DES) algorithm to provide data

TABLE 7. Technical specification of tested Raspberry Pi devices.

	ISA	CPU	SDRAM	OS (32-bit)
RPi Model B+	ARMv6Z (32-bit)	ARM1176JZF-S 700 MHz	512 MB	Raspbian 9.3
RPi Zero W	ARMv6Z (32-bit)	ARM1176JZF-S 1 GHz	512 MB	Raspbian 9.3
RPi 3 Model B+	ARMv8-A (64/32-bit)	Cortex-A53 1.4 GHz	1 GB	Raspbian 9.3
RPi 4 Model B	ARMv8-A (64/32-bit)	Cortex-A72 1.5 GHz	2 GB	Raspbian 9.3

Note: RPi – Raspberry Pi, ISA – Instruction Set Architecture, CPU – Central Processing Unit, SDRAM – Synchronous Dynamic Random Access Memory, OS – Operating System.

confidentiality. The reason for this choice is the missing support of a more secure Advanced Encryption Standard (AES) algorithm on MultOS cards. Figure 6 shows the speed of SHA-1 and 3DES algorithms across platforms. The Java Card reports a bit better results than MultOS cards. However, we can assume that our data will not exceed 200 B, and therefore the difference between SCs is minimal (except for the ML3 card, which reports much worse results in encryption), i.e. around 20 ms for SHA-1 and 40 ms for 3DES.

B. ARM PLATFORM AND SOFTWARE SELECTION

ARM processors are widely used in smartphone, tablet, smartwatch and other IoT mobile devices. Raspberry Pi is an ARM-based single-board computer that runs Linux and has various communication interfaces, e.g., General Purpose Input/Output (GPIO) pins, Ethernet, HDMI, USB ports and Bluetooth and WiFi adapters. These features allow a Raspberry Pi to be a part of many services in the IoT ecosystems. The technical specification of tested Raspberry Pi devices is shown in Table 7.

In public repositories, e.g., GitHub, there are several libraries with pairing-based cryptography support. The choice of the cryptographic library is crucial during the application development on resource-constrained devices. Since we are interested in the best performance, and therefore, the fastest pairing calculation, we focused on libraries implemented in C/C++ programming language. The selected libraries (Pairing Based Cryptography (PBC) [29], Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL) [31], University of Tsukuba Elliptic Curve and Pairing Library (TEPLA) [21], Efficient LIbrary for Cryptography (RELIC) [1] and MCL [40]) were installed on an embedded device, i.e., ARM-based microcomputer (Raspberry Pi 3 Model B). The benchmarks were run by using the 256-bit Barreto-Naehrig (BN) paring-friendly curve and averaged over 10-runs. The results are presented in Figure 7. We choose the MCL library, since it has support for the ARM architecture (32-bit and also 64-bit version) and has the best computational speed results among the compared libraries.

Furthermore, Table 8 shows the comparison of the most time consuming operations for our protocol which are performed on the tested ARM devices.

TABLE 8. Computational capability of tested ARM devices for most demanding operations of our scheme.

	ecMul G1 [ms]	ecMul G2 [ms]	expGT [ms]	Pairing [ms]
Raspberry Pi Model B+	6.2	12.9	19.0	38.7
Raspberry Pi Zero W	4.2	8.9	13.1	26.4
Raspberry Pi 3 Model B+	2.2	5.1	7.6	11.9
Raspberry Pi 4 Model B	0.8	1.6	2.5	5.1

C. TESTING SCENARIO AND SYSTEM PARAMETERS

In our testing scenario, receivers are represented by Raspberry Pi devices, and senders by SCs or Raspberry Pi devices. Normally, senders are represented by very resource-restricted devices (i.e., with processing and memory restrictions). For instance, a sender can be a user who owns a smartphone, a smart meter, an on-board unit built in cars (each of these devices can be represented by Raspberry Pis which are using the same ARM processors) or an access card (which is a SC). Accordingly, we choose a smart card platform that follows these constrained assumptions. Furthermore, the SC is a tamper-resistant device which securely allows the storage and the processing of sensitive data such as cryptographic keys. In case of SC application development, we use only standard MultOS Application Programming Interface (API) and free public development environment (Eclipse IDE for C/C++ Developers, SmartDeck 3.0.1, MUtil 2.8). The application is written in MultOS assembly code and C language.

Conversely, receivers can be servers, PCs, or embedded devices that are less constrained and, therefore, can be represented by a more powerful device. Tested SCs and Raspberry Pi hardware and software specifications are depicted in Tables 5 and 7, respectively. The Raspberries run Raspbian 9.0.3 operating system and C/C++ application. The application provides the communication with sender's smart card through Personal Computer/Smart Card (PC/SC) interface and executes `Unsigncrypt` (and `Signcrypt`) protocols. We use OpenSSL 1.1.1c library to perform cryptographic operations (i.e., hash and cipher), and MCL [40] library to perform operations over elliptic curves (i.e., EC point addition, EC scalar point multiplication and bilinear pairing). The application for Raspberry Pi was developed in NetBeans IDE 8.2 development environment. The code was remotely built and executed on the targeted devices, i.e. Raspberry Pi B+/ZeroW/3B+/4B.

The signcryption scheme implementation follows the restrictions of current smart cards (see Table 6), and the most recent security requirements defined by National Institute of Standards and Technology (NIST), see [5] and [6] for more details. The security level of our implementation is 112 bits. This restriction is due to the use of the 3DES cipher algorithm since the more secure AES-128 algorithm is not supported by our MultOS smart card. However, replacing

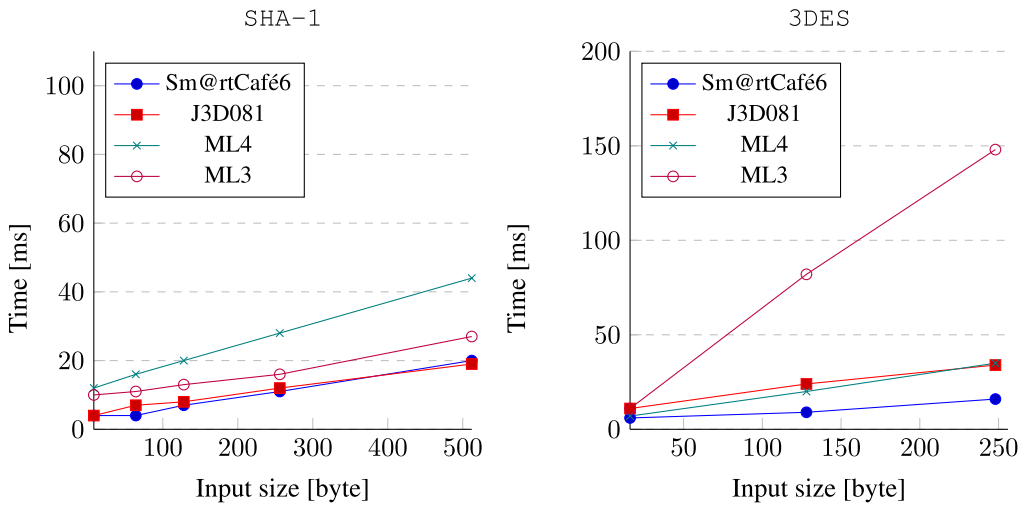


FIGURE 6. Message digest based on hash function SHA-1 and 3DES encryption on different smart card platforms.

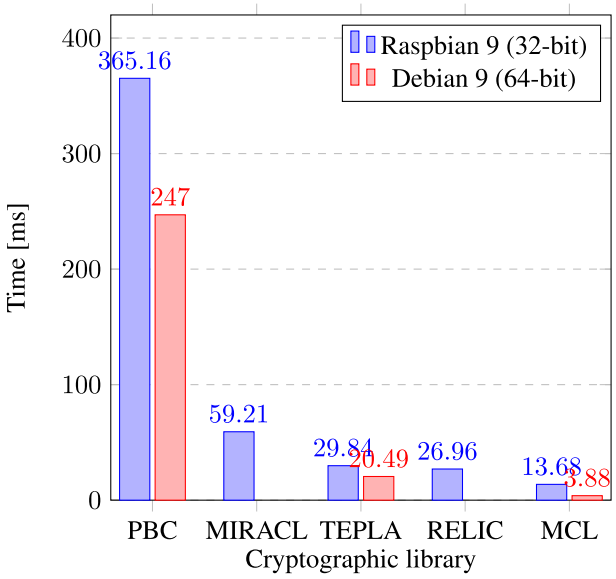


FIGURE 7. The comparison of different cryptographic libraries from the point of view of bilinear pairing performance over BN 256-bit elliptic curve on the ARMv8 processor (the Raspberry Pi 3, 32-bit and 64-bit OS).

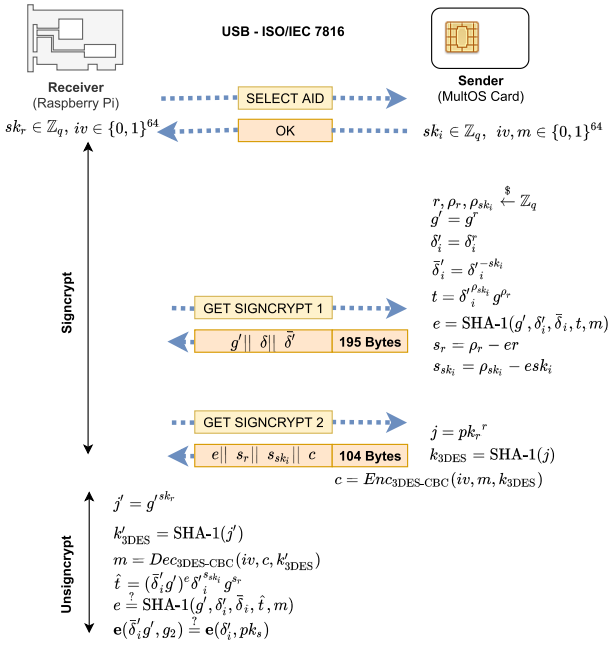


FIGURE 8. Implementation of `Signcrypt` and `Unsigncrypt` algorithms.

3DES with AES-128 algorithm directly increases the scheme security to 128 bits, since our signcryption scheme already uses 256-bits elliptic curves with embedding degree 12 (i.e. Barreto–Naehrig curve) and SHA-1 hash algorithm. Table 9 shows the system parameters set in details.

Our implementation considers only the single-receiver (i.e., many-to-one) scenario with messages of 64 bites (8 bytes), where MultOS card acts as a Sender and Raspberry Pi acts as both a Sender and a Receiver. A sketch of our implementation with the involved smart card is depicted in Figure 8. MultOS ML4 smart card supports only T=0 transport protocol. Since we need to transfer 299 bytes in total and T=0 protocol allows us to transfer data payload of a maximum of 255 bytes, we need to use two Application Protocol Data Unit (APDU) commands (`GET SIGNCRYPT 1` and `GET SIGNCRYPT 2`). While `GET SIGNCRYPT 1` performs group signature generation, `GET SIGNCRYPT 2` derives encryption key and encrypts data.

Figures 9 and 10 show the final computational times for `Signcrypt` and `Unsigncrypt` algoritms performed on Raspberries and the MultOS card. In case of Raspberries, the times are negligible and under 200 ms for both `Signcrypt` and `Unsigncrypt` protocols. In case of Raspberry Pi 4, the whole signcryption process takes less than 60 ms (without communication overhead). Generally, SCs are much slower

TABLE 9. Cryptographic algorithms and elliptic curve domain parameters.

	Algorithm	Size [Byte]	Description
Crypto	3DES-CBC	8 (block) 24 (key)	Data encryption
	SHA-1	20 (output)	Key Derivation Function (KDF) and Fiat–Shamir (FS) heuristic
	BN-curve	32	Signature Proof of Knowledge (SPK)
	Parameter	**Size [Bites]**	**Hexadecimal Value**
EC Parameters	p (characteristic)	254/[256]*	0x2523648240000001ba344d8000000086121000000000013a700000000000013
	a (constant)	0/[256]*	0x0000000000000000000000000000000000000000000000000000000000000000
	b (constant)	2/[256]*	0x0000000000000000000000000000000000000000000000000000000000000002
	G[x,y] (generator)	255/[513]*	0x04252364824000001BA344D8000000086121000000000013A700000000000012 0000000000000000000000000000000000000000000000000000000000000001
	q (order of G)	254/[256]*	0x2523648240000001ba344d8000000007ff9f800000000010a10000000000000d
	h (cofactor)	1/[8]*	0x01

Note: [Size]* – real allocated space on smart card.

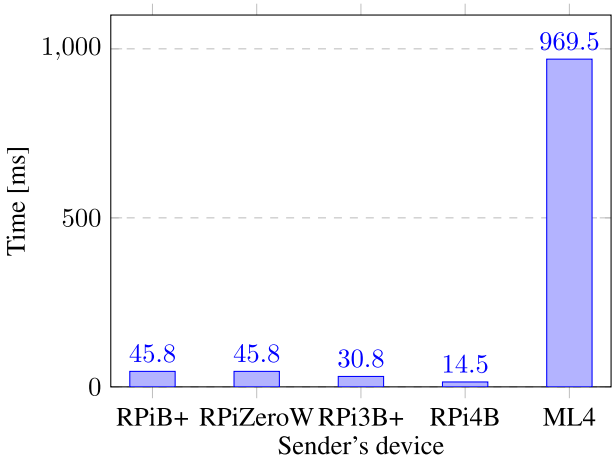


FIGURE 9. The performance comparison of `Signcrypt` algorithm performed on devices with different computing power.

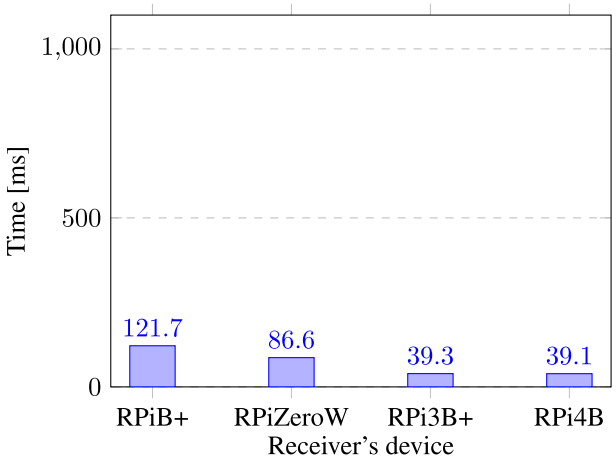


FIGURE 10. The performance comparison of `Unsigncrypt` algorithm performed on devices with different computing power.

to process `Signcrypt` algorithm compared to Raspberries. However, in our implementation the SC is fast enough (under 1 s including communication overhead) to be used in a real scenario.

XII. CONCLUSION

In this article, we presented a new privacy-enhancing group signcryption scheme that provides: unforgeability, confidentiality, ciphertext and sender anonymity, traceability, unlinkability, exculpability, coalition-resistance, and unforgeable tracing verification. The scheme is also compatible with current revocation techniques such as [13]. This is achieved by deploying our group signature scheme combined with an elliptic curve integrated encryption scheme. Our scheme is then extended to work in a multi-receiver scenario. In this case, a group of senders can send a signcrypted message to a group of receivers instead of only one receiver.

Moreover, the security analysis of the scheme is also provided. Our proposal is proven to be strongly existentially unforgeable under an adaptive chosen message attack, indistinguishable under an adaptive chosen ciphertext attack, and to provide ciphertext anonymity under an adaptive chosen ciphertext attack. The used signature has also sender's anonymity, traceability, unlinkability, and coalition-resistance privacy features. Moreover, the integration of NIZKPK in the key generation process (i.e., `Join` algorithm) allows achieving exculpability and unforgeable tracing verification properties.

The experimental results show that our scheme is efficient even on computationally restricted devices and can be therefore used in many IoT applications. The `Signcrypt` protocol on SCs takes less than 1 s (including communication overhead). The `Unsigncrypt` protocol complexity time on current ARM devices is negligible (less than 40 ms).

APPENDIX A
THEOREM 2 PROOF - CORRECTNESS

Once the message is correctly decrypted, we need to show that $\hat{t}$ is equal to t. This can be proven as follows:

$$\begin{aligned}\hat{t} &= (\bar{\delta}_i' g')^e \delta_i'^{s_{sk_i}} g^{s_r} \\ &= (\delta_i'^{-sk_i} g^r)^e \delta_i'^{s_{sk_i}} g^{s_r} \\ &= \delta_i'^{-esk_i} g^{er} \delta_i'^{s_{sk_i}} g^{s_r}\end{aligned}$$

$$
\begin{aligned}
&= {\delta'_i}^{-esk_i} g^{er} {\delta'_i}^{\rho_{sk_i} - esk_i} g^{s_r} \\
&= g^{er} {\delta'_i}^{\rho_{sk_i}} g^{s_r} \\
&= g^{er} {\delta'_i}^{\rho_{sk_i}} g^{\rho_r - er} \\
&= {\delta'_i}^{\rho_{sk_i}} g^{\rho_r} = t.
\end{aligned}
$$

Therefore, $e = \mathcal{H}(g', \delta'_i, \bar{\delta}_i, t, m) = \mathcal{H}(g', \delta'_i, \bar{\delta}_i, \hat{t}, m)$. In order to accept the signature, the receiver also needs that $\mathbf{e}(\bar{\delta}'_i g', g_2) \stackrel{?}{=} \mathbf{e}(\delta'_i, pk_s)$ holds. For a valid signature, we have that

$$
\begin{aligned}
\mathbf{e}(\bar{\delta}_i g', g_2) &= \mathbf{e}(\delta'_i, pk_s) \\
\mathbf{e}(\delta_i^{-sk_i r} g^r, g_2) &= \mathbf{e}(\delta_i^r, g_2^{sk_m}) \\
\mathbf{e}(g^{\frac{-sk_i r}{sk_m + sk_i}} g^r, g_2) &= \mathbf{e}(\delta_i^r, g_2^{sk_m}) \\
\mathbf{e}(g^{\frac{sk_m r + sk_i r - sk_i r}{sk_m + sk_i}}, g_2) &= \mathbf{e}(\delta_i^r, g_2^{sk_m}) \\
\mathbf{e}(\delta_i^{sk_m r}, g_2) &= \mathbf{e}(\delta_i^r, g_2^{sk_m}) \\
\mathbf{e}(\delta_i, g_2)^{sk_m r} &= \mathbf{e}(\delta_i, g_2)^{sk_m r}.
\end{aligned}
$$

Therefore, the correctness of the message and the signature is proven.

APPENDIX B
THEOREM 3 PROOF - STRONG EXISTENTIAL UNFORGEABILITY

We prove that if $\mathcal{A}$ can (t, q_s, ϵ)-break the signcryption scheme, then there exists an algorithm $\mathcal{B}$ such that, by interacting with $\mathcal{A}$, solves the p-SDH problem in time t' with advantage ϵ. Let $(g, d_1, d_2, \ldots, d_p, g_2, h)$ be a random instance of the p-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$, where $d_i = g^{x^i} \in \mathbb{G}_1$ for $i = 1, \ldots, p$ and $h = g_2^x \in \mathbb{G}_2$ for some unkown $x \in \mathbb{Z}_q$. Let $g = d_0$ and $x = sk_m$ for convenience. The goal of $\mathcal{B}$ is to compute the pair $(c, g^{1/(x+c)}) \in \mathbb{Z}_q \times \mathbb{G}_1$ for some value $c \in \mathbb{Z}_q \setminus \{x\}$ of its choice.

$\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

A. QUERY

$\mathcal{A}$ outputs a list of $q_s \leq p$ messages $m_1, \ldots, m_{q_s} \in \mathbb{Z}_q$. We can suppose that $q_s = p$ for simplicity. If less queries are made, we can always reduce the value of p to $p' = q_s$.

B. RESPONSE

$\mathcal{B}$ responds with p pairs

$$(c_i, \sigma_i) \leftarrow \texttt{Signcrypt}(par, sk_s, pk_r, m_i)$$

and p "signed" messages

$$(m_i, 0/1) \leftarrow \texttt{Unsigncrypt}(par, pk_s, sk_r, c_i, \sigma_i).$$

Therefore, $\mathcal{A}$ obtains p signature proofs of knowledge on its input messages.

Let f be the univariate polynomial defined as $f(X) = X + sk_i$. $\mathcal{B}$ chooses $\theta \in \mathbb{Z}_q$ and computes

$$g_1 = g^{\theta f(x)}$$

Therefore, $\mathcal{A}$ receives key sk_i, parameters $\widehat{par} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, \mathcal{H}, SYM)$ and public key $pk_s = h$.

If $f(x) = 0$, then $x = -sk_i$ and $\mathcal{B}$ can easily recover the secret key x and solve the p-SDH problem. If $f(x) \neq 0$, then g_1 and g_2 are independently and uniformly distributed random generators for the respective groups due to the action of ϕ. In this case, $\mathcal{B}$ has to apply both `Signcrypt` and `Unsigncrypt` algorithms and generate a valid signature σ_j on each message m_j, for $j = 1, \ldots, p$. To do so, by following `Signcrypt` algorithm, $\mathcal{B}$ chooses at random r, ρ_r, ρ_{sk_i}, encrypts m_j and creates the signature: $\sigma_j = (g', \bar{\delta}'_i, \delta'_i, e, s_r, s_{sk_i})$ where all the computations are made using g_1 instead of g. In fact, if g_1 is used, then $\delta_i = g_1^{1/f(x)} = g^{\theta}$ and $\mathcal{B}$ can compute each other component of the signature easily. This is repeated for each message m_j, where $j = 1, \ldots, p$.

Observe that σ_j is a valid signature on m_j under $\widehat{par}$, since

$$
\begin{aligned}
\mathbf{e}(\bar{\delta}'_i g', g_2) &= \mathbf{e}(g_1^{-sk_i r/f(x)} g_1^r, g_2) \\
&= \mathbf{e}(g^{-\theta sk_i r f(x)/f(x)} g^{r\theta f(x)}, g_2) \\
&= \mathbf{e}(g^{\theta r(f(x) - sk_i)}, g_2) \\
&= \mathbf{e}(g^{\theta r x}, g_2) = \mathbf{e}(g^{\theta r x}, g_2) \\
&= \mathbf{e}(g^{\theta f(x) r/f(x)}, g_2^x) = \mathbf{e}(\delta'_i, pk_s)
\end{aligned}
$$

The fact that $e = \mathcal{H}(g', \delta'_i, \bar{\delta}_i, \hat{t}, m_j)$ follows straightforward from the correctness of our scheme, see Theorem 2. These are exactly the verification steps performed by $\mathcal{B}$ when it applies `Unsigncrypt` algorithm and, therefore, links each message m_j to its signature σ_j. Since each message admits only a unique signature proof of knowledge, the output distribution is trivially correct.

C. OUTPUT

$\mathcal{A}$ returns for a user's identity sk_* a forgery (c_*, σ_*) such that σ_* is a valid signature and $c_* \notin \{c_1, \ldots, c_p\}$. The signature σ_* is a vector $\sigma_* = (g', \bar{\delta}'_i, \delta'_i, e, s_r, s_{sk_i})$ computed using the parameters $\widehat{par}$. We suppose that $sk_* \neq sk_i$ since $\mathcal{A}$ can choose sk_* knowing sk_i. By construction and uniqueness of the proof, we know that the component δ_i is equal to $\delta_* := g_1^{\frac{1}{x+sk_*}} = g^{\frac{\theta f(x)}{x+sk_*}}$ where $f(x) = x + sk_i$. If $x = -sk_*$, then $\mathcal{B}$ can easily recover the secret key x and solve the p-SDH problem. Otherwise, note that the polynomial f can be rewritten as $f(x) = x + sk_* + \gamma_*$ where $\gamma_* = sk_i - sk_* \in \mathbb{Z}_q$. Therefore, the ratio $f(x)/(x + sk_*)$ can be written as $f(x)/(x + sk_*) = 1 + \frac{\gamma_*}{x+sk_*}$ and the expression of δ^* becomes

$$\delta_* = g^{\theta(1 + \frac{\gamma_*}{x+sk_*})}$$

Taking roots of order θ and $\gamma_* \mod q$, $\mathcal{B}$ can compute

$$\omega = (\delta_*^{1/\theta} g^{-\theta})^{1/\gamma^*} = g^{1/x+sk_*} \in \mathbb{G}_1 \tag{1}$$

and obtain the pair (sk_*, ω) as the solution to the submitted instance of the p-SDH problem.

The claimed bound is obvious by the construction of the reduction.

APPENDIX C
LEMMA 4 PROOF - INDISTINGUISHABILITY

We wish to use $\mathcal{A}$ to attack the security of the DDH problem, the underlying *SYM* and proposed group signature (GS) schemes. During the proof the bitlength of the messages is bounded by μ.

Game 1: Following the definition of IND-CCA2 game (Section IV-B2), the below game is used to break the encryption scheme. $\mathcal{C}$ and $\mathcal{A}$ do as follows:

> **Setup**: $\mathcal{C}$ runs algorithms `Setup` and `KeyGen` to generate the public system security parameters par, the senders' public key pk_s, the manager's private key sk_m, the sender's i private key sk_s ($:= sk_i$) and the receiver's key pair (sk_r, pk_r). $\mathcal{A}$ is given (par, pk_s, sk_s, pk_r).
> **Queries-1**: $\mathcal{A}$ requests unsigncryption of at most $q_{s'}$ ciphertexts $c_1, \ldots, c_{q_{s'}}$, under pk_s and pk_r. $\mathcal{C}$ responds to each query with 1 and a message $(m_i, 0/1) \leftarrow$ `Unsigncrypt`$(par, pk_s, sk_r, c_i, \pi_i)$ if the obtained plaintext is valid and with 0 otherwise.
> **Challenge**: $\mathcal{A}$ outputs two equal-length messages m'_0 and $m'_1 \in \{0,1\}^*$ on which it wishes to be challenged. Then, hidden from $\mathcal{A}$ view, $\mathcal{C}$ chooses $b \leftarrow \{0,1\}$ and computes the challenge signcryption $(c'_*, \sigma'_*) \leftarrow$ `Signcrypt`(sk_s, pk_r, m'_b).
> **Queries-2**: $\mathcal{A}$ may request at most $q_{s''}$ signcryption and unsigncryption queries as in Queries-1 phase but with the restriction that $\mathcal{A}$ cannot query for c'_*.
> **Guess**: $\mathcal{A}$ produces its guess b' of b. $\mathcal{A}$ is successful if $b' = b$, i.e. the guess is correct.

Therefore, $Adv^{IND}_{\mathcal{A}}(t, q_s) = 2\mathbf{Pr}[b' = b] - 1$ represents the probability that $\mathcal{A}$ wins in the above game in time t with at most q_s signcryption and unsigncryption queries.

Since $\mathcal{A}$ is not allowed querying the unsigncryption protocol for the target cipher c'_*, namely **Type** $Q_\perp$ query, $\mathcal{A}$ cannot have access to Dec_{SYM} for the key k_{enc} corresponding to c'_*. In case a **Type** $Q_\perp$ query is made, Dec_{SYM} will output $\gamma \in \{0,1\}$. Let **Type** Q_v be any valid query different from **Type** $Q_\perp$.

Game 2: In this game, we prove that if $\mathcal{A}$ can $(t, q_s, \mu, m, \epsilon)$-break the signcryption scheme, then there exists an algorithm $\mathcal{B}$ such that, by interacting with $\mathcal{A}$, solves the DDH problem in time t' with advantage ϵ'. Let $\langle g, g^a, g^b, g^c \rangle$ be a random instance of DDH problem in $\mathbb{G}_2$, where $a, b, c \in \mathbb{Z}_q$. The goal is to determine whether $c \equiv ab \mod q$.

Therefore, *Game 2* is the same as *Game 1* but $\mathcal{B}$ has as input the following values: $(sk_r = b, pk_r = g^b)$, $r = a$, i.e. $g' = g^a$, and $j = g^c$ (see Figure 3 for more details on the protocol). In this way, k_{enc} and k'_{enc} are equal if and only if $c \equiv ab \mod q$. In other words, $\mathcal{A}$ believes that c is equal to $ab \mod q$ if $\mathcal{A}$ is successful in the game, i.e. $b = b'$.

We have three different situations depending on chosen DDH problem instance and query type.

1) When a valid DDH problem instance is given as input, $\mathcal{A}$ runs $\mathcal{B}$ as if one wants to mount an attack against the proposed signcryption protocol. Therefore,
$$Adv^{\text{DDH}}_{\mathcal{B}}(t', q) = \frac{1 + Adv^{IND}_{\mathcal{A}}(t, q_s)}{2} \tag{2}$$
2) If a non-valid DDH problem instance is given as input and $\mathcal{A}$ makes a **Type** $Q_\perp$ query, $\mathcal{A}$ runs $\mathcal{B}$ as if one wants to mount an attack against *SYM*. Therefore,
$$Adv^{\text{DDH}}_{\mathcal{B}}[(t', q) \wedge \textbf{Type } Q_\perp] \leq \frac{1 + Adv^{SYM}_{\mathcal{B}}(t''', |\kappa|)}{2} \tag{3}$$
where the inequality appears since $\mathcal{B}$ makes 0 signcryption queries in order to break *SYM*.
3) If a non-valid DDH problem instance is given as input and $\mathcal{A}$ makes a **Type** Q_v query, the game is the same of breaking GS and, therefore, breaking the p-SDH problem. The proof follows straightforward from the sUF-CMA (Theorem 3) of the signcryption scheme. Indeed, the fact that $k_{enc} \neq k'_{enc}$ does not affect the computation of σ'_* and `Verify` phase of `Unsigncrypt` algorithm since m'_0 and m'_1 are known. Therefore, we have
$$Adv^{\text{DDH}}_{B}[(t', q) \wedge \textbf{Type } Q_v] \leq Adv^{SDH}_{\mathcal{B}}(t'', p) \tag{4}$$
where the inequality appears since $\mathcal{B}$ only requires one round of signcryption and unsigncryption queries, i.e. $q'_s \leq q_s$ signcryption and unsigncryption queries in order to break GS.

Finally, combining Equations 2, 3 and 4, we obtain

$$Adv^{\text{DDH}}_{B}(t', q) \geq \frac{1 + Adv^{IND}_{\mathcal{A}}(t, q_s)}{2} - \frac{1 + Adv^{SYM}_{\mathcal{B}}(t''', |\kappa|)}{2} - Adv^{SDH}_{\mathcal{B}}(t'', p)$$

and the claimed bound directly follows from the last inequality.

APPENDIX D
LEMMA 6 PROOF - CIPHERTEXT ANONYMITY

The proof of this lemma follows the same structure of Lemma 4. Since it would be redundant to rewrite the same proof two times, we just sketch it emphasizing the main difference.

As in Lemma 4, we wish to use $\mathcal{A}$ to attack the security of DDH problem, the underlying *SYM* and the proposed GS schemes. During the proof, the bitlength of the messages is bounded by μ.

Game 1: In this case, we consider ANON-CCA game (Section IV-B3), where

$$Adv^{ANON}_{\mathcal{A}} = |4\mathbf{Pr}[(b', d') = (b, d)] - 1| \tag{5}$$

is the probability that the adversary $\mathcal{A}$ wins the ANON-CCA game for our proposed signcryption scheme.

As above, $\mathcal{A}$ can do two different queries: **Type** $Q_\perp$ query, which is $\mathcal{A}$ querying for (c'_*, σ_*), and **Type** Q_v query, that is any valid query.

Game 2: As above, if $\mathcal{A}$ can $(t, q_s, \mu, m, \epsilon)$-break the signcryption scheme, then there exists an algorithm $\mathcal{B}$ such that, by interacting with $\mathcal{A}$, solves the DDH problem in time t' with advantage ϵ'. Let $\langle g, g^a, g^b, g^c\rangle$ be a random instance of DDH problem in $\mathbb{G}_2$, where $a, b, c \in \mathbb{Z}_q$. The goal is to determine whether $c \equiv ab \mod q$. As in Lemma 4, we have three different situations, where only the first one is slightly different from the previous proof:

1) When a valid DDH problem instance is given as input, $\mathcal{A}$ runs $\mathcal{B}$ as if one wants to mount an attack against the proposed signcryption protocol, therefore,

$$Adv_{\mathcal{B}}^{\text{DDH}}(t', q) = \frac{1 + Adv_{\mathcal{A}}^{ANON}(t, q_s)}{4} \tag{6}$$

Observe that the denominator is 4, since this equality is derived from Equation 5.
2) If a non-valid DDH problem instance is given as input and $\mathcal{A}$ makes a **Type** $Q_\perp$ query, we have Equation 3.
3) If a non-valid DDH problem instance is given as input and $\mathcal{A}$ makes a **Type** Q_v query, the game is the same of breaking GS and, therefore, we have Equation 4.

Finally, combining Equations 3, 4 and 6, we obtain the claimed bound.

APPENDIX E
SECURITY ISSUES OF THE MOHANTY SCHEME [32]

Mohanty *et al.* [32] propose a signcryption scheme for secure electronic cashes. The authors claim that their scheme is secure such that neither the group manager nor any other member of the group can produce a valid signcrypted text. In this section, we show that the scheme presents security flows, in particular, we prove that it does not provide confidentiality, unforgeability, exculpability, and traceability properties.

Four entities are involved in the protocol: a Group Manager (GM), a Key Generation Center (KGC), users, and a verifier. Let briefly summarize the protocol (see [32] for more details):

- `Setup`: The KGC chooses two large primes p and q, a generator g of $\mathbb{Z}_p$ and computes $n = pq$. Then KGC sends n and g to GM.
- `Key Generation_KGC`: The KGC chooses its private key M_{sk}, its identity ID_{KGC} and computes its public key $M_{pk} = g^{M_{sk}}$. Then KGC sends (M_{pk}, ID_{KGC}) to GM.
- `Key Generation_GM`: The GM chooses V and ID_G and computes the group public and private key (G_{pbk}, G_{prk}). Then GM publishes $(n, g, M_{pk}, ID_{GM}, e, G_{pbk})$ and keeps private (d, V, G_{prk}), where $ed \equiv 1 \mod \phi(n)$.
- `Key Generation_User`: A user chooses its private parameter W and computes its public identity $ID_U = ID_{GM}^W$. The GM receives ID_U which is used to generate three values $\delta_1, \delta_2, \delta_3$ with $\delta_3 = (ID_{GM})^{\delta_1 \cdot d}$ These values are sent back to the user.
- `Signcryption`: The user signcrypts message M on behalf of the group. First the user chooses a private parameter $\beta \xleftarrow{\$} \mathbb{Z}_n^*$, then computes μ, key K and ciphertext σ as follows:

$$\begin{aligned} \mu &= \beta + (\delta_3)^{e \cdot \delta_1^{-1}} \mod n \\ K &= \mathcal{H}(\mu \cdot \beta) \mod n \\ \sigma &= (K \cdot M) + G_{pbk} \mod n \\ \Omega &= G_{pbk}^{\delta_3} \cdot (ID_{GM})^W \mod n \\ \Omega_1 &= g^{\delta_3} \mod n \\ \Omega_2 &= \Omega + \Omega_1^M \mod n \end{aligned} \tag{7}$$

Then the user sends the signcrypted text $(\mu, \sigma, \Omega, \Omega_1, \Omega_2)$ to the verifier.
- `Verification`: In order to find a message $M = M'$, the verifier computes the following steps:

$$\begin{aligned} \beta' &= (\mu - ID_{GM}) \mod n \\ K' &= \mathcal{H}(\mu \cdot \beta') \mod n \\ M' &= (\mu - G_{pbk}) \cdot K'^{-1} \mod n \end{aligned} \tag{8}$$

After finding M', the verifier checks the authenticity of the message as

$$\Omega_2 \stackrel{?}{=} \Omega + \Omega_1^{M'} \mod n$$

- `Opening`: In case of any legal dispute the group manager can open the signcryption and identify the sender by computing:

$$ID_U = \Omega / \Omega_1^{G_{prk}} \mod n \tag{9}$$

D. CONFIDENTIALITY

Confidentiality is achieved if no one can recover the signcrypted message, except for the receiver. This requirement does not holds since anyone can decrypt the message. The decryption process (Equation 8) works as follows:

$$\begin{aligned} \beta' &= (\mu - ID_{GM}) \mod n \\ K' &= \mathcal{H}(\mu \cdot \beta') \mod n \\ M' &= (\mu - G_{pbk}) \cdot K'^{-1} \mod n \end{aligned}$$

Note that ID_{GM}, μ and G_{pbk} are public values and therefore, the decryption process can be run by anyone.

E. UNFORGEABILITY

Unforgeability guarantees that only valid group members are able to signcrypt a message on behalf of the group. This requirement does not hold since anyone can generate a valid signcryption, since

$$(\delta_3)^{e \cdot \delta_1^{-1}} = (ID_{GM}^{\delta_1 \cdot d})^{e \cdot \delta_1^{-1}} = ID_{GM}.$$

Therefore, if anyone wants to sincrypt a message M, it can do as follows (Equation 7):

$$\begin{aligned} \mu' &= \beta + (\delta_3)^{e \cdot \delta_1^{-1}} = \beta + ID_{GM} \mod n \\ K &= \mathcal{H}(\mu' \cdot \beta) \mod n \\ \sigma' &= (K \cdot M) + G_{pbk} \mod n \end{aligned}$$

$$\Omega' = G_{pbk}^{\Delta} \cdot (ID_{GM})^{\Lambda} \bmod n$$
$$\Omega'_1 = g^{\Delta} \bmod n$$
$$\Omega'_2 = \Omega' + \Omega'^{M}_1 \bmod n$$

where $\Delta, \Lambda, \xleftarrow{\$} \mathbb{Z}_n^*$. Note that ID_{GM} and G_{pbk} are publicly available values, and Δ and Λ can be chosen at random by any entity that plays the role of the signer. Therefore, $(\mu', \sigma', \Omega', \Omega'_1, \Omega'_2)$ is a valid signature, which is untraceable by the GM. Since the unforgeability is broken, the coalition-resistance is broken as well.

F. EXCULPABILITY

Exculpability property provides that no one, even the group manager, can signcrypt on behalf of other group members. This requirement does not hold since the manager knows all secret values needed to generate signcrypted messages on behalf of the user. Namely, the manager knows values ID_{GM}, δ_3, ID_U. Therefore, it is easy to generate a signature equivalent to Equation 7:

$$\mu = \beta + ID_{GM} \bmod n$$
$$K = \mathcal{H}(\mu \cdot \beta) \bmod n$$
$$\sigma = (K \cdot M) + G_{pbk} \bmod n$$
$$\Omega = G_{pbk}^{\delta_3} \cdot ID_U \bmod n$$
$$\Omega_1 = g^{\delta_3} \bmod n$$
$$\Omega_2 = \Omega + \Omega_1^{M} \bmod n$$

G. TRACEABILITY

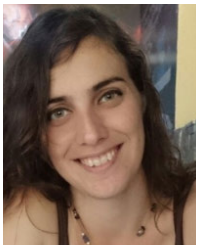
Traceability guarantees that the group manager can find the true signer, for any valid verified message. This requirement does not hold since any signer can compute value Ω of Equation 7 as $\Omega = G_{pbk}^{\delta_3} \cdot ID_{GM}^{\Lambda} \bmod n$, where $\Lambda \xleftarrow{\$} \mathbb{Z}_n^*$. This signature will be verified correctly, however, it will be untraceable by the GM (Equation 9):

$$ID_{\Lambda} = \Omega / \Omega_1^{G_{prk}} \bmod n$$

REFERENCES

[1] D. F. Aranha and C. P. L. Gouvêa. (2018). *RELIC is an Efficient LIbrary for Cryptography*. [Online]. Available: https://github.com/relic-toolkit/relic
[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2000, pp. 255–270.
[3] H. Bao, Z. Cao, and H. Qian, "On the security of a group signcryption scheme from distributed signcryption scheme," in *Proc. Int. Conf. Cryptol. Netw. Secur.* Berlin, Germany: Springer, 2005, pp. 26–34.
[4] E. Bangerter, J. Camenisch, and U. Maurer, "Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order," in *Proc. Int. Workshop Public Key Cryptogr.* Berlin, Germany: Springer, 2005, pp. 154–171.
[5] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management Part 1: General (revision 4)," Special Publication Part 1. General Revision, NIST, Gaithersburg, MD, USA, Tech. Rep. 800-57, 2016, p. 800.
[6] E. Barker, L. Chen, A. Roginsky, and M. Smid, "NIST special publication 800-56A revision 2: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography," NIST Special Publication, Gaithersburg, MD, USA, Tech. Rep. 800-56A, 2016.
[7] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Randomizable proofs and delegatable anonymous credentials," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2009, pp. 108–125.
[8] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2013, pp. 296–312.
[9] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *J. Cryptol.*, vol. 21, no. 2, pp. 77–149, 2008.
[10] A. Braeken and A. Touhafi, "AAA-autonomous anonymous user authentication and its application in V2G," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 12, p. e4303, 2018.
[11] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval, "Mutual authentication and group key agreement for low-power mobile devices," *Comput. Commun.*, vol. 27, no. 17, pp. 1730–1737, 2004.
[12] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Advances in Cryptology–(CRYPTO)* (Lecture Notes in Computer Science), vol. 1294, B. Kaliski, Ed. Berlin, Germany: Springer, 1997, pp. 410–424.
[13] J. Camenisch, M. Drijvers, and J. Hajny, "Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs," in *Proc. ACM Workshop Privacy Electron. Soc.*, Oct. 2016, pp. 123–133.
[14] P. Chaudhari and M. L. Das, "Privacy preserving signcryption scheme," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.* Cham, Switzerland: Springer, 2017, pp. 196–209.
[15] E. M. Cho and T. Koshiba, "Big data cloud deduplication based on verifiable hash convergent group signcryption," in *Proc. IEEE 3rd Int. Conf. Big Data Comput. Service Appl. (BigDataService)*, Apr. 2017, pp. 265–270.
[16] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. I-31, no. 4, pp. 469–472, Jul. 1985.
[17] Flexera. *2021 State of the Cloud Report*. Accessed: Jul. 2021. [Online]. Available: https://info.flexera.com/CM-REPORT-State-of-the-Cloud?lead_source=Website%20Visitor&id=Blog
[18] V. G. Martínez, E. L. Hernández, and Á. C. Sánchez, "A survey of the elliptic curve integrated encryption scheme," *J. Comput. Sci. Eng.*, vol. 2, no. 2, Aug. 2010.
[19] J. Hajny, P. Dzurenda, L. Malina, and S. Ricci, "Anonymous data collection scheme from short group signatures," in *Proc. 15th Int. Joint Conf. e-Business Telecommun.*, 2018, pp. 1–10.
[20] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "Identity-based ring signcryption schemes: Cryptographic primitives for preserving privacy and authenticity in the ubiquitous world," in *Proc. 19th Int. Conf. Adv. Inf. Netw. Appl.*, vol. 2, Mar. 2005, pp. 649–654.
[21] A. Kanaoka, "TEPLA elliptic curve and pairing library," Univ. Tsukuba, Tsukuba, Japan, Tech. Rep., 2018. [Online]. Available: https://github.com/TEPLA/tepla-library
[22] D. Kwak and S. Moon, "Efficient distributed signcryption scheme as group signcryption," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2003, pp. 403–417.
[23] D. Kwak, S. Moon, G. Wang, and R. H. Deng, "A secure extension of the Kwak–Moon group signcryption scheme," *Comput. Secur.*, vol. 6, pp. 435–444, Sep. 2006.
[24] S. Lal and P. Kushwah, "Anonymous ID based signcryption scheme for multiple receivers," IACR Cryptol. ePrint Arch., Dept. Math., Dr. Bhimrao Ambedkar Univ., Agra, India, Tech. Rep. 345, 2009.
[25] F. Li, Y. Han, and C. Jin, "Cost-effective and anonymous access control for wireless body area networks," *IEEE Syst. J.*, vol. 12, no. 1, pp. 747–758, Mar. 2018.
[26] H. Li and L. Pang, "Cryptanalysis of Wang et al.'s improved anonymous multi-receiver identity-based encryption scheme," *IET Inf. Secur.*, vol. 8, no. 1, pp. 8–11, 2014.
[27] F. Li, Z. Zheng, and C. Jin, "Secure and efficient data transmission in the Internet of Things," *Telecommun. Syst.*, vol. 62, no. 1, pp. 111–122, 2016.
[28] B. Libert and J. J. Quisquater, "Efficient signcryption with key privacy from gap Diffie-Hellman groups," in *Proc. Int. Workshop Public Key Cryptogr.* Berlin, Germany: Springer, 2004, pp. 187–200.
[29] B. Lynn. (2018). *The Pairing-Based Cryptography (PBC) Library*. [Online]. Available: https://crypto.stanford.edu/pbc/
[30] L. Malina, P. Dzurenda, and J. Hajny, "Evaluation of anonymous digital signatures for privacy-enhancing mobile applications," *Int. J. Secur. Netw.*, vol. 13, no. 1, pp. 27–41, 2018.
[31] MIRACL UK Ltd. (2018). *Multiprecision Integer and Rational Arithmetic Cryptographic Library–The MIRACL Crypto SDK*. [Online]. Available: https://github.com/miracl/MIRACL

[32] S. Mohanty, B. Majhi, and S. Das, "A secure electronic cash based on a certificateless group signcryption scheme," *Math. Comput. Model.*, vol. 58, nos. 1–2, pp. 186–195, 2013.
[33] Y. Mu and V. Varadharajan, "Distributed signcryption," in *Proc. Int. Conf. Cryptol. India*. Berlin, Germany: Springer, 2000, pp. 155–164.
[34] T. Nakanishi and Y. Sugiyama, "Anonymous statistical survey of attributes," in *Proc. Australas. Conf. Inf. Secur. Privacy* Berlin, Germany: Springer, 2001, pp. 460–473.
[35] T. Nakanishi, M. Tao, and Y. Sugiyama, "A group signature scheme committing the group," in *Proc. Int. Conf. Inf. Commun. Secur.* Berlin, Germany: Springer, 2002, pp. 73–84.
[36] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1999, pp. 223–238.
[37] L. Pang, L. Gao, H. Li, and Y. Wang, "Anonymous multi-receiver ID-based signcryption scheme," *IET Inf. Secur.*, vol. 9, no. 3, pp. 194–201, May 2015.
[38] L. Pang and H. Li, "nMIBAS: A novel multi-receiver ID-based anonymous signcryption with decryption fairness," *Comput. Inform.*, vol. 32, no. 3, pp. 441–460, 201.
[39] V. Saraswat, R. A. Sahu, and A. K. Awasthi, "A secure anonymous proxy signcryption scheme," *J. Math. Cryptol.*, vol. 11, no. 2, pp. 63–84, Jan. 2017.
[40] M. Shigeo. (2018). *MCL Library*. [Online]. Available: https://github.com/herumi/mcl
[41] N. P. Smart, "The exact security of ECIES in the generic group model," in *Proc. IMA Int. Conf. Cryptogr. Coding*, Berlin, Germany: Springer, 2001, pp. 73–84.
[42] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Conf. Theory Appl. Cryptol.* New York, NY, USA: Springer, 1989, pp. 239–252.
[43] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1997, pp. 256–266.
[44] H. Wang, Y. Zhang, H. Xiong, and B. Qin, "Cryptanalysis and improvements of an anonymous multi-receiver identity-based encryption scheme," *IET Inf. Secur.*, vol. 6, no. 1, pp. 20–27, 2012.
[45] M. Yung, *Practical Signcryption*. Berlin, Germany: Springer-Verlag, 2010.



SARA RICCI received the M.Sc. degree in mathematics from the University of Pisa, Italy, in 2015, and the Ph.D. degree in computer engineering and mathematics security from Universitat Rovira i Virgili, Spain, in 2018. She is currently a Postdoctoral Researcher with the Department of Telecommunications, Faculty of Electrical Engineering and Communication, Brno University of Technology, Czech Republic. She is also involved in the SPARTA H2020 and ERASMUS+ REWIRE projects. Her research interests include theoretical cryptography, in particular lattice-based and elliptic curve cryptography, and data privacy and security. She is also focused on the design of new privacy-preserving cryptographic protocols and their security analyses.

PETR DZURENDA received the Ph.D. degree from Brno University of Technology, Czech Republic, in 2019. He is currently a Postdoctoral Researcher with the Department of Telecommunications, Faculty of Electrical Engineering and Communication, Brno University of Technology. He is also involved in the SPARTA H2020 Project (task: Privacy-by-Design). He is the author or coauthor of several new privacy-friendly solutions and cryptographic schemes, such as group signatures and anonymous credentials, which are practically implementable on current smart cards. His research interests include privacy-enhancing technologies, cryptographic protocol design, and protocol implementation on constrained devices in the IoT area.

JAN HAJNY received the Ph.D. degree from Brno University of Technology, Czech Republic, in 2012. He currently works as an Associate Professor with the Faculty of Electrical Engineering and Communication, Brno University of Technology. He also deals with the research into cryptographic protocols for the privacy and digital identity protection. He is the Co-Founder and a Lead of the Cryptology Research Group (https://axe.utko.feec.vutbr.cz/) and responsible for managing the information security study program at the university. He is the author of more than 80 scientific publications and cooperates with renowned laboratories abroad.

LUKAS MALINA received the M.Sc. (Hons.) and Ph.D. degrees from Brno University of Technology (BUT), Czech Republic, in 2010 and 2014, respectively. He is currently an Associate Professor with the Department of Telecommunications, BUT. He is currently a Task Leader in the SPARTA H2020 Project (task: Privacy-by-Design) and a Senior Researcher in several Czech scientific projects focused on cybersecurity. He has published more than 80 papers in international journals and conferences. He has provided several invited research and teaching lectures at abroad, such as the University of Tampere, Finland, in 2013; URV Tarragona, Spain, in 2015; St. Petersburg ITMO, Russia, in 2017; and KU Leuven, Belgium, in 2017. His research interests include applied cryptography, privacy-preserving protocols, and authentication systems. He received the Dean's Prize for his master's thesis from BUT.