Dehez-Clementi, Marina and Deneuville, Jean-Christophe and Lacan, Jérôme and Asghar, Hassan and Kaafar, Mohamed Ali Who let the DOGS out: Anonymous but Auditable communications using Group Signature schemes with Distributed Opening. (2020) In: 4th International Workshop on Cryptocurrencies and Blockchain Technology - CBT 2020, 17 September 2020 - 18 September 2020 (Guildford, United Kingdom).

Who let the $\mathcal{DOGS}$ out: Anonymous but Auditable communications using Group Signature schemes with Distributed Opening

Marina Dehez-Clementi[1,2], Jean-Christophe Deneuville[3], Jérôme Lacan[1],
Hassan Asghar[2,4], and Dali Kaafar[2,4]

[1] ISAE-SUPAERO, Toulouse, France
[2] Macquarie University, Sydney, Australia
[3] ENAC, Toulouse, France
[4] Data61/CSIRO, Sydney, Australia

Abstract. Over the past two decades, group signature schemes have been developed and used to enable authenticated and anonymous peer-to-peer communications. Initial protocols rely on two main authorities, Issuer and Opener, which are given substantial capabilities compared to (regular) participants, such as the ability to arbitrarily identify users. Building efficient, fast, and short group signature schemes has been the focus of a large number of research contributions. However, only a few dealt with the major privacy-preservation challenge of group signatures; this consists in providing user anonymity and action traceability while not necessarily relying on a central and fully trusted authority. In this paper, we present $\mathcal{DOGS}$, a privacy-preserving Blockchain-supported group signature scheme with a distributed Opening functionality. In $\mathcal{DOGS}$, participants no longer depend on the Opener entity to identify the signer of a potentially fraudulent message; they instead collaborate and perform this auditing process themselves. We provide a high-level description of the $\mathcal{DOGS}$ scheme and show that it provides both user anonymity and action traceability. Additionally, we prove how $\mathcal{DOGS}$ is secure against message forgery and anonymity attacks.

1 Introduction

Developed in the 70's, digital signatures are one of the most important primitives in public key cryptography and provide *authentication*, *integrity* and *non-repudiation* to various applications. However, they do not provide privacy of the signer [12]. Our focus is on signature schemes that provide both ***user anonymity*** and ***action traceability***, which fall in the realm of *group signature schemes.*

Introduced by Chaum and van Heyst [5], group signature schemes enable members of a group to sign messages on behalf of the group without revealing their identity. The signature can be verified by the recipient as a valid signature from the group, but cannot identify the member who generated the signature. However, signing members remain accountable for their messages as the *group manager*, a third-party managing the group, can open their signatures and hence identify them.

In [2], Bellare *et al.* presented some fundamental security notions for dynamic group signatures. A key feature of their scheme is that the group manager is split into two distinct entities: the *Issuer* which interacts with users to authenticate their credentials, and the *Opener* which is called when a signature needs to be opened.

Role of the Opener. The opener acts as the *tracing authority* in charge of linking a signed message to its origin, namely the signer. By definition, the role of the Opener entails no privacy preservation. Existing literature [9, 10] usually considers a unique entity to be playing the role of the Opener, inducing strong assumptions of the level of trust in such an entity as well as its resilience. In Vehicular Ad-hoc Networks (VANETs) for instance, vehicles use group signatures to sign the (location and time-dependent) road-safety messages they broadcast. A vehicle's identity is protected by the scheme unless the Opener is compromised, in which case signatures may be opened which then breaches user privacy. An alternative way to relax these strong assumptions in the Opener is to implement a ***distributed tracing authority*** instead.

Related Work on Distributed Traceability. Studies that propose a distributed traceability functionality for group signature schemes (e.g., [3, 6]) often leverage Shamir's Secret Sharing (SSS) scheme to generate shares of the Opener secret key and then distribute one share per user. This approach however presents a clear limitation: the computation of these shares is again centralized which still represents a single point of failure from an adversary perspective.

Our Contribution. In this paper, we propose the Distributed Opening Group Signature $\mathcal{DOGS}$ scheme, a BSZ group signature scheme [2] enhanced with the Distributed Key Generation (DKG) protocol from [11] (later denoted ETHDKG) for the distribution of the Opener secret key. The advantage of our DKG-enhanced group signature over SSS-based approaches is that the key generation is not entrusted to a third party. Instead, it is the result of the collaboration of a group of users. Hence, none of them knows all the shares, making the scheme strong against a more powerful adversary.

Structure of the paper. Section 2 provides a description of the system model, desirable security features and the primitives used. Section 3 gives a generic presentation of the $\mathcal{DOGS}$ construction. Section 4 presents the security analysis of $\mathcal{DOGS}$. Section 5 concludes the paper.

2 Preliminaries

In this section, we present the adversary model and corresponding security features of $\mathcal{DOGS}$. We describe the system, giving a real-world use case, and introduce our solution for the distribution of the Opener.

2.1 Adversary Model and Desirable Features

In $\mathcal{DOGS}$, we consider an adversary that wants to break the underlying group signature scheme. Therefore it can perform de-anonymization attacks (i.e. intercept a message and identify the signer), key recovery attacks or forgery attacks (i.e. create a signature that would defeat the opening procedure or frame a legitimate user). All these scenarios are captured in Bellare *et al.* [2].

Desirable features. Our main objectives are for our $\mathcal{DOGS}$ scheme to be correct and to provide anonymity, traceability and non-frameability. Let λ denotes the security parameter.

Correctness. The correctness property guarantees that signatures issued by honest users:

i) should pass the verification,
ii) should trace to the correct issuer if opened with the Opening key, and
iii) the proof output by the *Open* process should verify the *Judge* algorithm. □

Anonymity. Let $\mathcal{U}_0$ and $\mathcal{U}_1$ be two honest registered users, and σ a valid signature issued by $\mathcal{U}_b$ for some $b \in \{0,1\}$. The anonymity property requires that no probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ can guess b with non negligible (in λ) advantage. □

Traceability. An adversary breaks the traceability property if she succeeds in creating a valid signature σ such that either:

i) no registered (or revoked) user can be identified when σ is legitimately opened, or
ii) the proof, produced by a honest opening, revealing that σ belongs to user $\mathcal{U}$, does not convince the *Judge* algorithm. □

Non-frameability. Finally, the non-frameability property requires that no PPT adversary $\mathcal{A}$ can create a valid signature that would trace to an honest user if opened, unless this user has effectively issued it. □

We prove that $\mathcal{DOGS}$ is compliant with these security goals in Sec. 4.

2.2 System model

Involved in our $\mathcal{DOGS}$ scheme are: an authority called the Issuer Iss for the generation of initial cryptographic information, and a body of users, each with a unique identity i (e.g. $i \in \mathbb{N}$). We assume over half of the participants are honest. Iss is public and has its own secret key. It interacts with users to issue them an authenticated signing key. We consider a Region of Interest (RoI), which users can join and leave and in which they can communicate. One Issuer is responsible for one RoI.

Use Case. In VANETs, we can assimilate the RoI to a neighbourhood. Vehicles are the users. They can enter, leave and move in the area, communicate with each other and broadcast road information. The Roadside unit acts as the Issuer.

2.3 Distributed Key Generation for $\mathcal{DOGS}$

Distributed Key Generation (DKG) protocols are fundamental building blocks for a variety of cryptographic schemes [8]. They enable a group of users to collaborate in order to obtain a common public key. The strength of DKG lies in that the public key can be computed by any honest participant, while the corresponding secret key cannot. Instead, it requires a threshold number of collaborating honest parties to derive it.

In this paper, we draw from Schindler *et al.*'s DKG implementation [11] (ETHDKG). The authors present a fully functioning DKG protocol for threshold signature generation, based on Ethereum smart contracts. In the following section, we explain how we combine this DKG proposition with BSZ group signature to produce a new Group Signature scheme with Distributed Opener called $\mathcal{DOGS}$.

3 Protocol Description

There are three phases in $\mathcal{DOGS}$:

phase 1 *"Distributed Generation of an Opening Keys"*,
phase 2 *"Inter Communications and App-related event logging"* and
phase 3 *"Auditing and Identification"*.

We transpose them into five distinct modules (Fig. 1) namely **Bootstrapping**, **Registration** and **Opening Keys (OK) Generation** (phase 1), **Application** (phase 2) and **Audit** (phase 3).

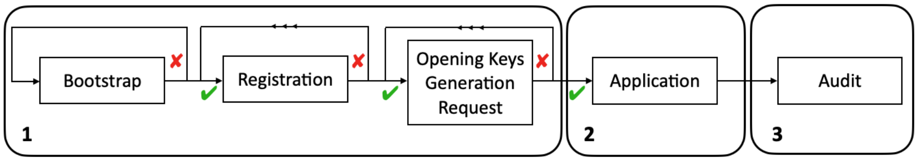

Fig. 1. $\mathcal{DOGS}$ Workflow diagram

Algorithms and their usage. Users can become group members by joining the Region of Interest and interacting with the Issuer. They become sub-opener authorities and participate to the opening process if they own a share of an Opening secret key. The group of sub-openers therefore replaces the Opener. The scheme is specified as a tuple $\mathcal{DOGS}$ =(`GKg, dOKg, UKg, Join, Iss, GSig, GVf, Request, Collaborate, Open, Judge`) of PPT algorithms, whose intended usage and functionalities are presented in this section. This protocol heavily relies on previous works [2, 11] and we advice the readers to briefly review them before they proceed. Indeed, `(GKg, UKg, Join, Iss, GSig, GVf, Open, Judge)` are inherited from [2], and `dOKg` invokes `(sharing, share_verification, claim_verification, key_derivation)` formalized from [11].

Usage of the Public Ledger. Throughout these phases, we assume the existence of a trusted message delivery service. To do so, we make use of a public ledger to record relevant information, i.e. any public data that can be used to assert the validity and integrity of the shared cryptographic parameters.

We leverage a classic blockchain implementation with the usual functionalities related to block addition, linkage and forking issues. In $\mathcal{DOGS}$ protocol, the

Module	Algorithm	Type	Comments
Bootstrap	`GKg`	Bootstrap	publication of the group public key gpk, and the Issuer public key ipk
Registration	`UKg`	none	U_i generates local identity $\mathbf{upk}[i], \mathbf{usk}[i]$
	`Join`	Joining Request	by U_i, sends personal data to Issuer
	`Iss`	Issuing Reply	2 issues: accept/reject. If accept, the Issuer records the link between $\mathbf{upk}[i], \mathbf{usk}[i]$ and (i, pk_i, sig_i)
Opening Keys Generation	`sharing`	Opening Keys (OK) Generation Request	U_i generates and distributes commitments (C_{ij}), encrypted shares $\overline{s_{ij}}$ and value h^{s_i}, for $k = 0..t$, $j = 1..n$
	`share_verification`	Claim	a user U issues a claim $=< status, data, key, proof >$
	`claim_verification`	Claim Verification	all U check the validity of the claim; 2 issues: accept/reject
	`key_derivation`	Key Publication	the Issuer determines and publishes the resulting Opening public key opk
Application	`GSig`	Signature	user U produces a group signature σ under opk
	`GVf`	Verification	user U verifies a group signature σ under opk. 2 issues: accept/reject.
Audit	`Request`	Open Request	user U requests the opening of a group signature σ under opk
	`Collaborate`	Secret Key Publication	one of the authorized users publishes the re-constructed Opening secret key osk
	`Open`	Open	the signature σ is opened and the result is published by the requester
	`Judge`	Judge	a user - not the requester - asserts and publishes the validity of the requester's opening

Table 1. Description of the transaction (Tx) types and corresponding published data.

transaction types and corresponding published data are summarized in Table 1, which includes the definition of: seven transaction types for Phase 1 (e.g. the *bootstrap transaction* corresponds to the publication of bootstrapping information); two transaction types for Phase 2, which illustrate the signing action of a user and verifying process of other members of the group; and finally, four transaction types for Phase 3.

Since the information published does not hold private or sensitive information, the public ledger is readable by anyone. However, we restrict the writing permissions to authenticated users only.

3.1 Phase 1: Distributed Generation of the Opening Keys

In the following paragraphs, we will explain each step of Phase 1.

Bootstrap. The scheme starts with the bootstrapping of the system. It consists in the Issuer executing the `GKg` algorithm and results in the publication of both the group public key gpk and the the Issuer's public key ipk (Bootstrap transaction).

Registration. User U_i enters the RoI. First, U_i executes the `UKg` algorithm to obtain a personal public/private key pair, referring to its local identity $(\mathbf{usk}[i], \mathbf{upk}[i])$. Then, U_i starts the `Join,Iss` interactive protocols. It executes the `Join` function and triggers the JOIN REQUEST transaction. The generation and publication of related information (Tab. 1) is compliant with Bellare *et al.*'s $\mathcal{GS}$ `Join` algorithm, but enhanced with logging functionalities for traceability purposes. The Issuer receives encrypted data from U_i, including identifying information such as the public key pk_i used to verify U_i's signed messages. It compares it to the public record for data integrity checks; then executes the `Iss` function, triggering the publication of resulting data and the ISSUING REPLY transaction. The **Registration** step has two issues: the first ends up in U_i's request being rejected due to faulty data; the second grants U_i with a certificate $cert_i$ and allows it to perform subsequent actions such as OPENING KEYS (OK) GENERATION REQUEST, make a CLAIM, or a CLAIM VERIFICATION. If so, the Issuer records, in its local database, the relationship between the user's local identity given by `UKg`, and the authenticated identity (i, pk_i, sig_i) used in the RoI.

Opening Keys (OK) Generation. Since the Opener is no longer a single entity, from now on, we will use the wording "Opening" to designate the functionality it was in charge of. We now consider an authenticated user U_i and explain how it can request the generation of Opening keys to protect its future communications (see `GSig` [2]).

U_i executes the `sharing` function inherited from [11]. This function triggers the OPENING KEYS (OK) GENERATION REQUEST transaction. Subsequently, U_i generates and publishes the required information (commitments (C_{ik}), shares (s_{ij}) and value h^{s_i} according to [11]) for the computation of an Opening public key.

The broadcasting of this new transaction triggers an update of the local ledgers of the users. They individually execute the `share_verification` function to check the correctness of the share that has been sent to them by U_i (note: each share s_{ij} is encrypted with a symmetric encryption scheme of key $k_{ij} = g^{sk_i sk_j} = pk_i^{sk_j} = pk_j^{sk_i}$). The execution of this function triggers a CLAIM transaction and publishes the result as $< status, data, key, proof >$. Depending on *status*, there are two different results: if *status* contains the value "*no_claim*" or U_j does not reveal k_{ij} or its proof $\pi(k_{ij})$, then the share is accepted and U_j in turn has to broadcast its own shares (see **Case 1**). Else if, *status* value is equal to "*claim*", then the protocol holds as others check the claim (see **Case 2**).

Case 1. Therefore, U_j in turn executes the `sharing` function, distributes the resulting shares and publishes related data. Again, `share_verification` function is used to check the validity of the shares but this time either the peers $\{U_k\}_{k \neq j \in N}$ accepts U_j's share and it ends there, or it rejects it.

Case 2. In this case, a share has been rejected and a claim has been broadcast. Hence, users execute `claim_verification` function to check its validity. Doing so, they trigger a CLAIM VERIFICATION transaction which results in either the claim being denied or accepted.

Once all the shares and claim have been verified, the Issuer can execute `key_derivation` function. It browses the public records and determines which shares are usable to compute the final Opening public key. It finally publishes this key *opk* and $\mathcal{R}$ the list of authenticated local identities which participated to the establishment of this Opening key.

In the following sub-section, we explain how anyone in the RoI can use *opk* in `GSig` to encrypt its signature, ensuring its anonymity in communications while still providing action traceability.

3.2 Phase 2: Inter Communications and Application-related event logging

Application. Our $\mathcal{DOGS}$ scheme has been initially thought to be applied in the context of local logging of events, especially for VANETs. Let us consider U_i has a road-safety information to share with vehicles in the neighbourhood, for instance an alert about a car accident. However, U_i does not want to reveal its identity nor its position at this particular time (to prevent location-based identity inference [7]). If U_i applies `GSig` function on the alert message, it is able to produce a signature that refers to the group but protects its identity. It additionally triggers the SIGNATURE transaction. The use of [2]'s `GSig` along with the DKG-computed Opening public key ensures that no single sub-opener can open this signature, hence identify U_i. However, users in the neighbourood can still individually execute `GVf` [2] to assert the correctness of the received signature and trust U_i's alert. The result gets published along with VERIFICATION transaction.

That is how user anonymity is provided in $\mathcal{DOGS}$. In the following subsection, we consequently explain how $\mathcal{DOGS}$ also provides action traceability.

3.3 Phase 3: Auditing and de-anonymization

Audit. This module regroups the required functionalities implemented in $\mathcal{DOGS}$ for providing action traceability.

With `Request`, the requester U_i triggers the OPEN REQUEST transaction, hence summoning users in the RoI to collaborate in order to reconstruct an Opening secret key. U_i therefore communicates the signature σ it wants to open, the message m it signs and the corresponding Opening public key *opk*. For traceability guarantees, this request gets published on to the public ledger.

Then, the `Collaborate` function is executed. All U_j in the set $\mathcal{R}$ related to *opk* will collaborate to reconstruct the Opening secret key *osk* (by consecutively summing their shares to previous partial results). The last peer to add its own secret s_{t+1} also publishes the result while triggering the SECRET KEY PUBLICATION transaction. It includes the initial data $< m, opk, \sigma >$, the set $\mathcal{R}$, and the result of their work *osk*.

Finally, by retrieving *osk*, the requester U_i can identify the origin of σ and publishes the result via the `Open` function (OPEN transaction). Other peers in the RoI can consequently check this identification by executing the `Judge` function (JUDGE transaction).

4 Security Analysis

In this section, we show that the use of ETHDKG functionalities is compatible with the security environment related to group signature schemes as presented in [2], hence that $\mathcal{DOGS}$ presents all the security properties we aimed for. Due to space restrictions, the security proofs are only sketched, but most of them are directly inherited from the BSZ construction and the ETHDKG protocol. Only the anonymity property requires a careful treatment. We refer the reader to BSZ [2] for a complete description of the experiments for each security feature.

Let λ be $\lambda = 100$ bits of security provided by the instantiation of the ETHDKG with the elliptic curve BN254 [1].

Correctness. Properties i) and iii) are directly satisfied using BSZ. Assuming the Opening key is correctly reconstructed, which is the case with overwhelming probability (in λ) thanks to ETHDKG, then ii) is also satisfied. □

Anonymity. In the original experiment [2], $\mathcal{A}$ does *not* have access to the opening oracle. In $\mathcal{DOGS}$, we weaken this assumption to "$\mathcal{A}$ has access to at most t shares of the Opening secret key (including her own, if she is a registered user)". Then $\mathcal{A}$ has negligible advantage in λ in recovering osk [11] and our anonymity feature boils down to the original one, which is fulfilled by using BSZ. □

Traceability. Similarly, in [2], $\mathcal{A}$ is granted access to the Opening secret key osk. Therefore, $\mathcal{A}$ learns nothing more by corrupting users and $\mathcal{DOGS}$ traceability is inherited from BSZ. □

Non-frameability. Here again, since $\mathcal{A}$ has already access to osk in the original experiment, she obtains no additional advantage by exploiting the shares of the Opening secret key, and $\mathcal{DOGS}$ satisfies non-frameability as BSZ. □

Discussion. Most of the security features are directly inherited from the BSZ and ETHDKG constructions. However, the anonymity experiment as described in [2] needs to be slightly modified. Indeed, an adversary $\mathcal{A}$ against the original property could create and corrupt sufficiently many (more than t) users to obtain their respective shares of the Opening secret key osk, and hence reconstruct it. Using osk, $\mathcal{A}$ could trivially break the original anonymity property. Therefore, an upper bound has to be integrated to compensate for the extra knowledge $\mathcal{A}$ gets by corrupting users.

5 Conclusion

In this work, we present $\mathcal{DOGS}$, a Blockchain-supported group signature scheme which implements a distributed Opening functionality.

It would be meaningful, in future works, to include additional materials notably regarding the algorithms and their explicit definitions, ways to contact users from the group of sub-openers that are no longer in the Region of Interest or potential solutions to redistribute their shares to newcomers, an implementation of the scheme with performance analysis to compare to real-world scenarios (e.g. VANETs), and the establishment of more formal security proofs. Also, it would be interesting to complete the current scheme with the addition of a distributed Issuing functionality hence proposing a fully distributed Blockchain-supported group signature scheme. Camenisch *et al.* [4] recently came up with their own proposition and it might be interesting to analyse and compare both to some extent.

Nonetheless, being the combination of a BSZ group signature scheme and the ETHDKG protocol, we already succeed in formulating DOGS and show that, only by distributing the Opener among a group of users, we preserve the traceability feature of group signatures while enhancing their anonymity feature. Indeed, our scheme is proven stronger than BSZ's in terms of anonymity due to the use of DKG. Furthermore, the scheme brings an additional novelty as it enables the dynamic definition of the set of sub-opener authorities which makes it more practical to applications with variable topologies such as VANETs.

Acknowledgements

We would like to express our great appreciation to E. Lochin for his valuable and constructive suggestions during the planning and development of this research work.

This work was partly supported by the French government through the Toulouse graduate School of Aerospace Engineering (TSAE). Contract ANR-17-EURE-0005. This work has also been supported by the Optus Macquarie University Cyber Securiy Hub.

References

1. Barreto, P.S., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: International workshop on selected areas in cryptography. pp. 319–331. Springer (2005)
2. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Cryptographers' Track at the RSA Conference. pp. 136–153. Springer (2005)
3. Blömer, J., Juhnke, J., Löken, N.: Short group signatures with distributed traceability. In: International Conference on Mathematical Aspects of Computer and Information Sciences. pp. 166–180. Springer (2015)
4. Camenisch, J., Drijvers, M., Lehmann, A., Neven, G., Towa, P.: Short threshold dynamic group signatures. IACR Cryptol. ePrint Arch. **2020**, 16 (2020)
5. Chaum, D., Van Heyst, E.: Group signatures. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 257–265. Springer (1991)
6. Ghadafi, E.: Efficient distributed tag-based encryption and its application to group signatures with efficient distributed traceability. In: International Conference on Cryptology and Information Security in Latin America. pp. 327–347. Springer (2014)
7. Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preventing location-based identity inference in anonymous spatial queries. IEEE transactions on knowledge and data engineering **19**(12), 1719–1733 (2007)
8. Neji, W., Blibech, K., Rajeb, N.B.: A survey on e-voting protocols based on secret sharing techniques. Proceedings of CARI 2018 p. 142 (2018)
9. Perera, M.N.S., Koshiba, T.: Fully dynamic group signature scheme with member registration and verifier-local revocation. In: International conference on mathematics and computing. pp. 399–415. Springer (2018)

10. Sakai, Y., Schuldt, J.C., Emura, K., Hanaoka, G., Ohta, K.: On the security of dynamic group signatures: Preventing signature hijacking. In: International Workshop on Public Key Cryptography. pp. 715–732. Springer (2012)
11. Schindler, P., Judmayer, A., Stifter, N., Weippl, E.: Ethdkg: Distributed key generation with ethereum smart contracts. Tech. rep., Cryptology ePrint Archive, Report 2019/985, 2019, https://eprint.iacr.org (2019)
12. Yang, G., Wong, D.S., Deng, X., Wang, H.: Anonymous signature schemes. In: International Workshop on Public Key Cryptography. pp. 347–363. Springer (2006)