Towards Anonymous Crowdsensing: A Smart Contract-Mediated Privacy Framework

Towards Anonymous Crowdsensing: A Smart Contract-Mediated Privacy Framework

Giuseppe Cacciapuoti*, Chiara Anna Cartarasa*, Delia Cavalca*, Luca Bedogni†, Stefano Ferretti*
*Department of Computer Science and Engineering, University of Bologna, Italy
†Department of Computer Science and Mathematics, University of Modena-Reggio Emilia, Italy
luca.bedogni@unimore.it, s.ferretti@unibo.it

Abstract—Crowdsensing platforms face a fundamental trade-off between data utility and participant privacy, where traditional approaches require users to expose sensitive identity information, creating barriers to widespread adoption. This paper presents a blockchain-based protocol that addresses these privacy concerns through pseudonymous participation and secure data exchange mechanisms. Our approach exploits smart contracts as trusted intermediaries to eliminate direct communication between data initiators and contributors, while employing asymmetric cryptography to enable secure key exchange without pre-established channels. The protocol enhances privacy through campaign-specific key pairs that remain unlinkable to participants' persistent identities, contextual separation of cryptographic identities, and anonymization sets that obscure individual actions within larger groups. We developed a prototype implementation to verify the correctness of the protocol and to evaluate gas consumption. Through simulation, we assess system performance under varying user dynamics and activity rates. Results confirm the viability of the proposal.

Index Terms—Crowdsensing, Blockchain, Smart Contracts

I. Introduction

Crowdsensing has emerged as a powerful paradigm for large-scale data collection, leveraging the ubiquity of mobile devices and the willingness of users to contribute sensor data for collective intelligence applications [1]. From environmental monitoring and traffic analysis to urban planning and social sensing, crowdsensing platforms enable the aggregation of distributed sensing capabilities to address complex challenges. However, the trade-off between data utility and participant privacy remains one of the most significant barriers to widespread adoption of crowdsensing systems. Traditional approaches often require participants to reveal sensitive information about their identity, location, or behavior patterns, creating privacy vulnerabilities that discourage participation and limit the effectiveness of data collection campaigns [2].

The integration of blockchain technology with crowdsensing presents a promising avenue to address these privacy concerns while maintaining the trustless and decentralized nature essential for large-scale participation [3]. Blockchain's immutable ledger and smart contract capabilities offer new mechanisms for coordinating data collection without relying on centralized authorities, while cryptographic techniques can provide strong privacy guarantees for participants. The challenge lies in designing protocols that not only preserve participant anonymity, but also ensure data integrity, enable fair compensation mechanisms, and maintain acceptable performance characteristics under realistic operating conditions.

The problem of achieving anonymization in crowdsensing can be mitigated by resorting to the use of anonymization sets, i.e., groups of users whose data or actions are aggregated together to make individual contributions indistinguishable from one another. The use of anonymization sets is a well-established practice across various domains in computer security and privacy. This is particularly true when individual actions or data points need to remain unlinkable while maintaining system functionality. For example, in cryptocurrency systems, mixing services and protocols, such as CoinJoin, aggregate multiple transactions, making it harder to trace individual payment flows [4]. Similarly, the Tor network relies on routing traffic through multiple nodes in large anonymity sets to obscure the origins of users [5]. In blockchain systems, ring signatures, as implemented in privacy-focused cryptocurrencies such as Monero, hide the real sender among a set of potential signers [6]. Even in more traditional security applications, techniques like k-anonymity in database privacy and mix networks in communication systems rely on the principle of hiding individual actions within larger groups [7].

The proposed protocol makes use of blockchain smart contracts to establish a secure data exchange mechanism for crowdsensing applications, while enhancing privacy through pseudonymous participation. Indeed, the motivation is to enable a secure data contribution while minimizing exposure of participants' identities. Traditional crowdsensing systems often require direct communication between data initiators and contributors, creating privacy vulnerabilities through exposed communication channels and identity information. This proposed protocol eliminates this direct interaction by using a blockchain-based smart contract as a trusted intermediary for all communication.

The key innovation of the protocol lies in its use of asymmetric cryptography paired with smart contract mediation. The initiator of the crowdsensing campaign and the contributors generate separate key pairs that are used exclusively for this specific crowdsensing campaign.

Neither party needs to reveal their true identity or persistent identifiers to participate. The smart contract serves as the communication medium, handling the exchange of encrypted public keys and cryptographic material. By implementing this approach, several privacy-enhancing properties emerge:

- Pseudonymous participation: Contributors can generate campaign-specific key pairs that are not linked to their persistent identities, which improves pseudonymity.
- Elimination of direct communication: All interactions occur through the smart contract, removing the need for direct channels that could expose identifying network information.
- Contextual Separation: The cryptographic identities used in the campaign can be distinct from participants' identities in other contexts, supporting privacy through separation.
- Secure Key Exchange: The protocol enables secure exchange of the data encryption key without requiring a pre-established secure channel between participants.

We developed a prototype system with the core components of the smart contract-based architecture. This implementation allowed us to verify the correctness of the protocol and to measure the gas consumption of smart contract calls. Based on this, we evaluated the protocol within a simulated environment. Simulation is a typical tool for evaluating blockchain-based solutions [8]–[10]. By creating controllable and repeatable environments, simulation modeling enables to study the impact of various parameters on system performance and reliability. In addition, simulations can model adversarial behaviors or rare edge cases that are difficult to reproduce in live deployments.

In the experimental scenario, we simulate user contribution behaviors using exponential, Gaussian, and lognormal distributions to capture a wide range of realistic activity patterns. For each distribution, we varied the number of users and their activity rates, modeling both users and verifiers as independent Poisson processes. This allowed us to assess the impact of different user dynamics on system performance metrics, such as gas consumption and time-to-threshold achievement. Results demonstrate the viability of the proposal.

The remainder of this paper is organized as follows. Section II presents the protocol, while in Section III we provide some details on the implementation of the smart contract that coordinates the crowdsensing activity. Section IV discusses the performance evaluation of the system- Finally, some concluding remarks are provided in Section V.

II. Protocol for Secure Data Exchange in Crowdsensing Campaigns

In this section, we present a secure protocol for data exchange in crowdsensing campaigns that leverages smart contracts and asymmetric encryption to ensure confidentiality and integrity of data. The protocol employs a multilayered encryption approach where data passing through the smart contract to and from the campaign initiator (in the rest of the section referred to as "B") are encrypted using B's private/public key pair, thus enhancing confidentiality, as only B possesses the ability to access the public keys of both data contributors (A) and verifiers (V). This approach is in some parts similar to previous work on the exchange of asymmetric keys through smart contracts [11], even if the application domain and final protocol are very different.

A. Protocol Actors

The protocol involves the following actors:

- A: End-users who contribute data to the crowdsensing campaign;
- B: Campaign initiator responsible for collecting and processing submitted data;
- V: Independent verifier who validates the submitted data;
- SC: Smart contract deployed on a blockchain that mediates interactions.

B. Cryptographic Components

Our protocol employs several pairs of cryptographic keys to establish secure communication channels between the actors:

1) Encryption/Decryption Keys (Generated by B):

- K_{enc}: Encryption key for data to be stored on IPFS;
- K_{dec}: Decryption key for retrieving data from IPFS.

These form an asymmetric cryptographic key pair that is used specifically to secure data stored in the distributed file system.

2) Actor-Specific Key Pairs:

- (PK_A, SK_A): Public and private key pair of contributor A;
- (PK_B, SK_B): Public and private key pair of the crowdsensing campaign initiator B;
- (PK_V, SK_V): Public and private key pair of verifier V.

Each actor generates these keys specifically for the campaign to isolate the security domain. The public key of B (PK_B) is stored in the smart contract during its initialization after deployment.

For the sake of clarity in the notation, we define:

- $Enc(message, public\_key)$: Encryption operation using the public key;
- $Dec(ciphered\_message, private\_key)$: Decryption operation using the private key.

C. Protocol Workflow

The protocol consists of three primary phases: initialization, data submission, and verification.

1) Smart Contract Deployment and Users Initialization Phase: The initialization phase is depicted in Figure 1, which shows the smart contract deployment by the crowdsensing campaign initiator, as well as the interactions that involve user enrollment.

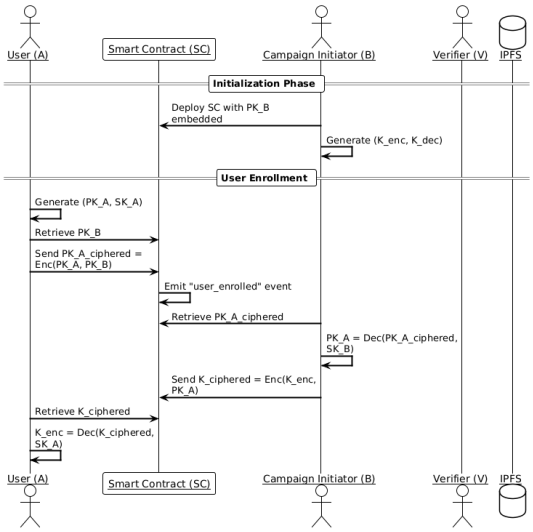

Fig. 1. Sequence diagram of the secure crowdsensing protocol illustrating the interactions between User (A), Smart Contract (SC), initiator (B), Verifier (V), and IPFS - Initialization phase and user enrollment.

In particular,

i. Initiator B generates the key pair (PK_B, SK_B) locally and deploys the smart contract SC, embedding PK_B within the contract state.
ii. Initiator B generates the cryptographic key pair (K_{enc}, K_{dec}) to secure data that will be stored in IPFS by contributors.
iii. Contributor A generates the key pair (PK_A, SK_A) locally.
iv. Contributor A retrieves PK_B from the smart contract.
v. A signals his intention to contribute data by sending to the smart contract $PK_{A_{ciphered}} = Enc(PK_A, PK_B)$, i.e., his public key ciphered with B's public key.
vi. The smart contract emits a user_enrolled event that is monitored by the initiator B.
vii. B retrieves the encrypted public key $PK_{A_{ciphered}}$ and decrypts it using his private key: $PK_A = Dec(PK_{A_{ciphered}}, SK_B)$.
viii. B encrypts the data encryption key with A's public key: $K_{ciphered} = Enc(K_{enc}, PK_A)$.
ix. B transmits $K_{ciphered}$ to the smart contract.
x. A retrieves $K_{ciphered}$ from the smart contract and decrypts it using their private key: $K_{enc} = Dec(K_{ciphered}, SK_A)$.

2) Verifiers Initialization Phase: Figure 2 shows the interactions to register a verifier to the crowdsensing campaign. In this case,

i. A verifier (V) generates the key pair (PK_V, SK_V) locally.
ii. V signals his intention to carry out the verification sending $PK_{V_{ciphered}} = Enc(PK_V, PK_B)$ to the smart contract.
iii. The smart contract validates V's authorization to act as a verifier through access control mechanisms. The implementation details of which are beyond the scope of this work, but can be achieved using Role-Based Access Control (RBAC) frameworks for smart contracts, e.g., [12].
iv. The smart contract emits a verifier_enrolled event monitored by initiator B.
v. B retrieves $PK_{V_{ciphered}}$ and decrypts it: $PK_V = Dec(PK_{V_{ciphered}}, SK_B)$.
vi. B encrypts the decryption key with V's public key: $K_{V_{ciphered}} = Enc(K_{dec}, PK_V)$.
vii. B transmits $K_{V_{ciphered}}$ to the smart contract.
viii. V retrieves both $K_{V_{ciphered}}$ and $CID_{ciphered}$ from the smart contract.
ix. V decrypts the decryption key: $K_{dec} = Dec(K_{V_{ciphered}}, SK_V)$.
x. V decrypts the CID: $CID = Dec(CID_{ciphered}, K_{dec})$.

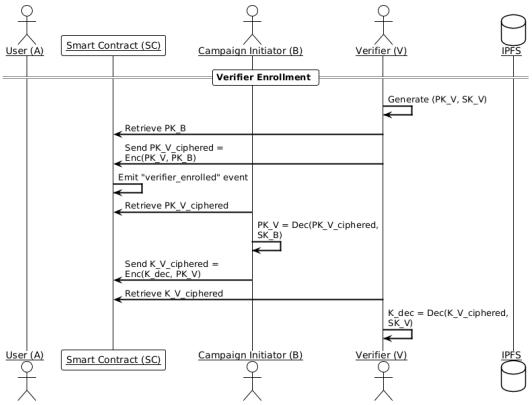

Fig. 2. Sequence diagram of the secure crowdsensing protocol illustrating the interactions between User (A), Smart Contract (SC), initiator (B), Verifier (V), and IPFS - Verifier enrollment.

3) Data Submission and Verification Phases: Figure 3 shows the interactions related to the data upload by a user A and the verification by V. In this case,

i. A encrypts his data using K_{enc} and uploads them to IPFS, obtaining a Content Identifier (CID).
ii. A communicates the location of the IPFS data to the smart contract by sending $CID_{ciphered} = Enc(CID, K_{enc})$.
iii. The smart contract generates a verification request event that is monitored by Verifier V.
iv. V retrieves the encrypted data from IPFS using the CID.
v. V decrypts the data using K_{dec}.

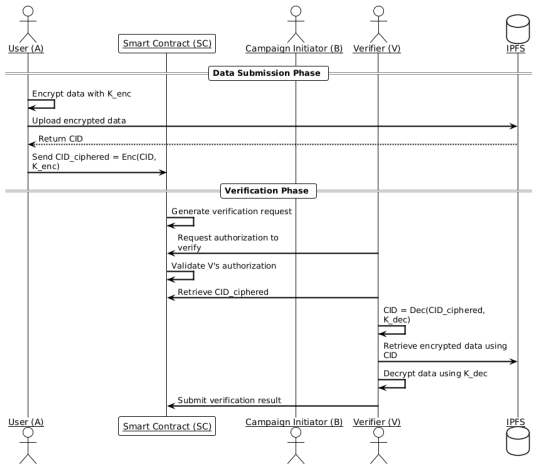

Fig. 3. Sequence diagram of the secure crowdsensing protocol illustrating the interactions between User (A), Smart Contract (SC), initiator (B), Verifier (V), and IPFS - Data upload and verification.

vi. V performs data validation and submits the verification result to the smart contract.

D. Security Considerations

This protocol architecture provides multiple security benefits:

1) Data Confidentiality: All sensitive data are encrypted before being stored on IPFS or transmitted through the blockchain.
2) Key Distribution Security: The distribution of cryptographic keys occurs through encrypted channels, preventing unauthorized access.
3) Separation of Concerns: The verification process is isolated from the data contribution, ensuring unbiased validation.
4) Key Management: Initiator B functions as a key management authority for this crowd-sensing campaign only, improving control over access to the data.

By implementing this multi-layered encryption approach, we ensure that only authorized parties can access the submitted data, while maintaining the transparency and immutability benefits of blockchain technology.

A feature of the protocol is that Users only know K_{enc} to encrypt data and the corresponding CIDs to retrieve them from the distributed file system. Conversely, Verifiers only know K_{dec} for decrypting data. This ensures that Users cannot gain access to information about data generated by other Users, even if they monitor transactions from other participants to the smart contract, as they do not possess the decryption key K_{dec}. Conversely, Verifiers can retrieve all Users data, and this is precisely why we assume the existence of an RBAC-like mechanism for Verifier enrollment. In any case, Verifiers do not possess the encryption key K_{enc}, which would otherwise enable them to, for example, retrieve data generated by a user, claim it as their own, and maliciously pretend to have generated the data themselves (thereby fraudulently obtaining undeserved rewards). Clearly, the campaign initiator B is able to retrieve all the data, but this is a reasonable assumption that complies with the requirements of the specific domain of the crowdsensing application.

III. Smart Contract Implementation

In this paper, we specifically focus on the implementation of the smart contract that manages the entire lifecycle of the crowdsensing campaign.

When Users intend to participate, they register their public keys (uploadPublicKey()) which enables the crowdsensing campaign Admininstrator to securely share encryption keys (sendEncryptedKey()) for data protection.

The data submission process is initiated when Users upload data references via IPFS hashes, through uploadData(), paying a small fee that serves as both a spam prevention measure and part of the economic incentive structure. Verifiers, who have previously registered their public keys (by calling uploadVerifierPublicKey()), can access unverified data (getUnverifiedData()) and perform validation checks.

Upon validation (verifyData()), the smart contract automatically manages the reward distribution process. Valid data submissions result in the original fee being returned to the User, while Verifiers receive compensation for their validation work (rewardVerifier()), creating a sustainable economic model. Invalid submissions are marked accordingly without rewards.

The campaign operates with configurable parameters, including a minimum participation threshold (minimumParticipants). When this threshold is reached, the campaign automatically concludes (closeCampaign()), demonstrating how smart contracts can autonomously manage campaign lifecycle events.

Our implementation leverages blockchain's inherent qualities to provide transparency through events, i.e., DataUploaded, VerificationRequested, DataVerified.

IV. Performance Evaluation

To assess the viability of our proposed protocol, we analyzed the Solidity based implementation of the smart contract, in order to obtain estimations of the costs of the operations to be accomplished through the smart contract. Gas consumption can be directly translated into monetary costs for deploying and interacting with smart contracts, making it a crucial metric for the practical evaluation of implementation. Based on the gas consumption estimates, we conducted a series of simulations to understand the viability and scalability of our approach in potential crowdsensing campaigns with varying numbers of participants.

A. Gas Usage Analysis

An analysis of gas consumption was performed on the blockchain. Gas is a unit that measures the computational

effort required to execute operations on an Ethereum-based blockchain, such as deploying smart contracts, executing functions, or storing data [13]. Each operation in a transaction has an associated gas cost based on its complexity and the resources it consumes, including processing power and storage space. In the case of the smart contract we implemented, gas consumption occurs during various stages of the transaction process. By carefully analyzing gas consumption, we can understand if the platform is efficient and economically viable while maintaining transparency and privacy in the whole process related to the crowdsourcing campaign. Moreover, understanding these costs allows for the fine-tuning of the system's fee structure to ensure users are not overcharged and verifiers are adequately rewarded, ultimately leading to a balanced and sustainable DApp ecosystem.

For our decentralized crowdsensing platform, gas consumption occurs across various operational stages. When users contribute data, gas is consumed for several cryptographic operations: the exchange of generated key pairs, the communication through the smart contract related to files encryption, upload to IPFS, and the recording of IPFS hashes on the blockchain. Similarly, the verification workflow incurs gas costs when verifiers retrieve encrypted data, validate its integrity, and process it using appropriate cryptographic keys.

The implementation of security features, i.e., the public-private key infrastructure, introduces additional gas requirements. Operations such as transmitting public keys and the encryption/decryption keys (KEnc, KDec) to the Smart Contract, significantly impact overall transaction costs. The Smart Contract must also generate and handle security-related events (e.g., UserEnrolled and VerifierEnrolled) monitored by the Admin node and by Verifiers.

Table I summarizes the gas consumption for key operations within our crowdsensing platform. In our measumerements, we noticed a difference on gas consumption related to the first call of each function and subsequent calls. For this reason, we report both first-time and subsequents transaction costs. All operations show considerable gas savings in subsequent calls, with reductions ranging from approximately 10% to 73%. The reason might be related to the fact that the first call of each operation typically involves initializing storage variables, which is gas-intensive on Ethereum. Subsequent calls modify existing variables, which costs substantially less.

The smart contract deployment requires 4,351,407 gas units, representing as expected the highest single gas consumption. The operation referred as "Send Encrypted KDec" consumes the most gas after deployment, probably due to the inizialization of the hash mapping, since subsequent calls have highly reduced costs. An interesting aspect is that this operation uses the same smart contract function that is used for sending KEnc, i.e., sendEncryptedKey(). In the case of sending KDec, it is the Administrator who encrypts KDec with the public key of the Verifier, and then uploads the encrypted key to the smart contract. Conversely, the "Send Encrypted KEnc" operation involves the Administrator encrypting KEnc with the public key of the User who wants to upload a file, followed by uploading the encrypted key to the smart contract.

Data Upload is relatively efficient, given the fact that only hashes are sent to the smart contract, with only a modest reduction for subsequent uploads. Data Verification requires less gas than other functions, as expected for a reading/verification operation.

User and Verifier Public Key Uploads consume similar gas amounts, indicating a standardized key registration process. Verification Request has the lowest gas consumption and no reduction in subsequent calls, since it is a simple function.

Figure 4 shows how average gas consumption varies based on the number of users in a setting where we maintain a fixed number of verifiers and each user generates a constant amount of data. Therefore, the higher the number of users, the larger the amount of data sensed. The chart demonstrates that average consumption decreases as the number of users increases. This aligns with the results already presented, since as discussed, the initial operations required by the smart contracts for the first user are more expensive than subsequent operations. Consequently, as the number of contributions increases, the initial costs have a diminishing impact on the total.

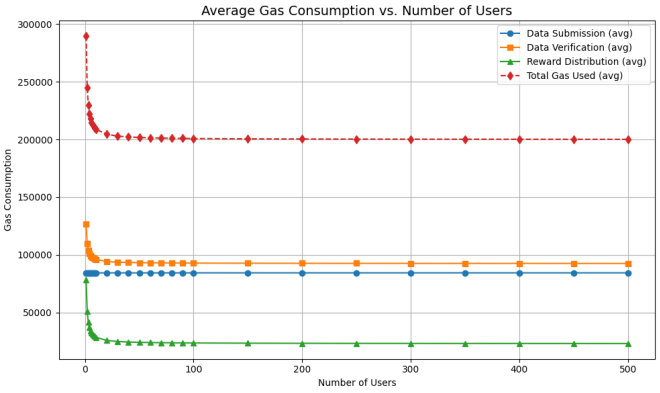

Fig. 4. Average gas consumption vs number of users.

B. Simulation Results

Given the gas usage obtained from the implementation and testing of the smart contract, in order to evaluate the scalability and viability of this system, we analyzed via simulation two specific key metrics, i.e., the average gas consumption per user and time to threshold. The average gas consumption per user measures the mean amount of gas expended by each participant in the crowdsensing protocol. It is important to notice that in these calculations we avoided counting the initialization

TABLE I
Gas Consumption Analysis for Crowdsensing Smart Contract Operations

Process	Gas Consumption - First Call	Gas Consumption . Subsequent Calls
Smart Contract Deployment	4,351,407	
User: Upload Public Key	162,124	57,032
Admin: Send Encrypted KEnc	462,492	135,100
Data Upload	168,222	151,122
Verifier: Verification Request	41,790	41,790
Verifier: Upload Public Key	162,211	57,119
Admin: Send Encrypted KDec	1,638,008	438,516
Data Verification	99,332	82,232

costs for the smart contract deployment, as well as the users and verifiers enrollment, thus analyzing only the cost associated to the data contribution and verification phases. The time to threshold measures the total simulated time required for the system to collectively reach a predefined number of confirmed data contributions, thus reflecting the efficiency and responsiveness of the protocol under different load conditions.

To capture a broad spectrum of real-world behaviors, we have modeled the number of data contributions per user event using three distinct probability distributions:

- Exponential Distribution: This distribution is memoryless and characterized by a single parameter (i.e., the mean). It is commonly used to model the time between independent events that occur at a constant average rate. The exponential distribution allows us to simulate scenarios where most users contribute infrequently, but occasional bursts of activity are possible. The mean values are chosen to represent typical and moderately active user behaviors.
- Gaussian Distribution: The Gaussian distribution is suitable for modeling phenomena where contributions are expected to cluster around an average value, with deviations in both directions. We varied the mean values, while keeping constant the standard deviations equal to 1. Needless to say, negative values were truncated to zero to maintain physical plausibility. In this case, the mean values varied while the standard deviation was kept equal to 2.
- Lognormal Distribution: The lognormal distribution is positively skewed, producing mostly small values with occasional large outliers. This is particularly relevant for crowdsensing scenarios where a few users may contribute disproportionately more than the average. The parameters for the lognormal distribution (mean and standard deviation in log space) are chosen to explore both moderate and highly skewed contribution patterns, highlighting the impact of heavy-tailed behaviors on system performance. In particular, we selected a set of mean values to explore a different range of user contribution behaviors. In fact, the lower mean value (0.5) models scenarios where users are generally less active, resulting in a majority of small contribution events. The intermediate mean (1) represents a typical or baseline level of user activity, while the higher mean (5) allows us to capture cases where users are, on average, significantly more active, increasing the likelihood of larger contribution bursts.

For each setup, we varied the number of users in a range from 10 to 200, while keeping constant the number of verifiers (20). In our simulations, both users and verifiers were modeled as independent Poisson processes. This means that the times between consecutive actions (such as data contributions by users or verification steps by verifiers) are generated according to exponential distributions with configurable rate parameters. We varied the rates (λ) at which users and verifiers, respectively, generate events in the simulation, thus changing the inter-arrival times of actions for each participant type. In these results, we show a scenario where, on average, users wait longer between consecutive actions w.r.t. verifiers, resulting in less frequent user activity and a more dispersed (variable) pattern of contributions over time. In particular, we set $\lambda_{\text{users}} = 0.001$, while $\lambda_{\text{verifiers}} = 0.1$. In substance, in this configuration, the verifiers are modeled as significantly more prompt in performing verification actions compared to the 'lazy' or sporadic behavior of the users.

Each simulation scenario was repeated 50 times. For each setting, we show the mean values and standard deviation for both metrics. Results for the average gas consumption per user and the time to reach the threshold, when adopting the exponential, Gaussian, and lognormal distributions are shown in Figures 5, 6 and 7, respectively.

As expected, our results show that as the number of users increases, the average gas cost per user decreases. What changes among the figures is the scale of the values obtained when varying the distributions. This trend is intuitive: with more users participating concurrently, each individual user is responsible for a smaller share of the total contributions required to reach the global threshold. The increased competition among users to contribute means that the workload, and thus the associated gas cost, is distributed more evenly and thinly across the population.

A similar effect is observed for the time to threshold. With a larger number of concurrent users, the system reaches the required number of contributions more quickly, as multiple users are able to act in parallel. The aggregate

rate of contributions increases with the number of participants, leading to a reduction in the overall time needed to achieve the target.

These results, while expected, highlight the critical importance of active participation in crowdsensing scenarios. High levels of user engagement not only reduce the individual burden in terms of gas costs (whose related activity is, however, rewarded) but also significantly improve the responsiveness and efficiency of the system as a whole.

V. Conclusions

This paper has presented a blockchain-based protocol that addresses the fundamental privacy challenges in crowdsensing systems through pseudonymous participation and smart contract mediation. By eliminating direct communication between participants and using campaign-specific cryptographic identities, our approach shows that strong privacy guarantees can be achieved without compromising system functionality or data integrity.

The experimental evaluation through simulation modeling validates the effectiveness of the protocol across diverse user behavior patterns and system configurations. Our results show that the gas consumption overhead remains acceptable while achieving the desired privacy properties, making the solution practically viable for real-world deployment. The use of anonymization sets, combined with contextual separation of identities, provides data protection against various privacy attacks, while maintaining the transparency and trustless nature essential for crowdsensing applications.

There are several areas for improvement. First, we plan to explore more sophisticated anonymization techniques to further protect user privacy [14]. Moreover, it could be interesting to integrate machine learning algorithms to detect and mitigate fraudulent data submissions. Finally, we plan to conduct real-world experiments to validate the system's performance.

Acknowledgments

This work is partially supported by the European Union - NextGenerationEU within the framework of PNRR Mission 4 - Component 2 - Investment 1.1 under the Italian Ministry of University and Research (MUR) programme "PRIN 2022" - grant number 2022N2NH42 SmartShires – CUP: H53D23003570006.

This work was funded by the European Union under the NextGeneration EU Programme within the Plan "Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 – C2. Dalla Ricerca all'Impresa - Investimento 1.1 Fondo per il Programma Nazionale della Ricerca (PNR) e Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN)" by the Italian Ministry of University and Research (MUR), Project title: "Tight control of treatment adherence and efficacy by tElemedicine for an improved personalized Management of Patients with hemOphilia (TEMPO)", Project code: 2022PKTW2B – CUP E53D23012700006, MUR D.D. financing decree n. 1065 of 18 July 2023.

References

[1] L. Gigli, F. Montori, M. Zichichi, L. Bedogni, S. Ferretti, and M. Di Felice, "On the decentralization of mobile crowdsensing in distributed ledgers: an architectural vision," in Proc. of the Proc. of the 2024 IEEE 21th Annual Consumer Communications and Networking Conference (CCNC 2024). IEEE, 2024, pp. 1–7.

[2] M. Hijjawi, F. Jamil, H. Jamil, T. Alsboui, R. Hill, and I. A. Hameed, "Optimal smart contracts for controlling the environment in electric vehicles based on an internet of things network," Computer Communications, vol. 224, pp. 192–212, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366424002081

[3] L. Bedogni and S. Ferretti, "Incentivizing decentralized privacy-preserving crowd-sensing with smart contracts," in Proceedings of the Workshop on Sensing, Algorithms, and Intelligent Applications (SAIA-2025) - 2025 IEEE Symposium on Computers and Communications (ISCC). IEEE, July 2025.

[4] H. Rosenquist, D. Hasselquist, M. Arlitt, and N. Carlsson, "On the dark side of the coin: Characterizing bitcoin use for illicit activities," in Passive and Active Measurement: 25th International Conference, PAM 2024, Virtual Event, March 11–13, 2024, Proceedings, Part II. Berlin, Heidelberg: Springer-Verlag, 2024, p. 37–66. [Online]. Available: https://doi.org/10.1007/978-3-031-56252-5_3

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, ser. SSYM'04. USA: USENIX Association, 2004, p. 21.

[6] A. Hinteregger and B. Haslhofer, "Short paper: An empirical analysis of monero cross-chain traceability," in Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers. Berlin, Heidelberg: Springer-Verlag, 2019, p. 150–157. [Online]. Available: https://doi.org/10.1007/978-3-030-32101-7_10

[7] P. Samarati, "Protecting respondents' identities in microdata release," IEEE Trans. on Knowl. and Data Eng., vol. 13, no. 6, p. 1010–1027, Nov. 2001. [Online]. Available: https://doi.org/10.1109/69.971193

[8] L. Serena, G. D'Angelo, and S. Ferretti, "Security analysis of distributed ledgers and blockchains through agent-based simulation," Simulation Modelling Practice and Theory, vol. 114, p. 102413, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X21001131

[9] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: A blockchain network simulator," in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2019, pp. 325–329.

[10] G. Diamantopoulos, R. Bahsoon, N. Tziritas, and G. Theodoropoulos, "Symbchainsim: A novel simulation tool for dynamic and adaptive blockchain management and its trilemma tradeoff," in Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, ser. SIGSIM-PADS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 118–127. [Online]. Available: https://doi.org/10.1145/3573900.3591121

[11] R. Muth and F. Tschorsch, "Smartdhx: Diffie-hellman key exchange with smart contracts," in 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), 2020, pp. 164–168.

[12] J. P. Cruz, Y. Kaji, and N. Yanai, "Rbac-sc: Role-based access control using smart contract," IEEE Access, vol. 6, pp. 12 240–12 251, 2018.

[13] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, vol. 151, no. 2014, pp. 1–32, 2014.

[14] F. Barbàra, M. Zichichi, S. Ferretti, and C. Schifanella, "Dlt-based personal data access control with key-redistribution," Cluster Computing, vol. 28, no. 6, p. 404, 2025. [Online]. Available: https://doi.org/10.1007/s10586-024-05016-y

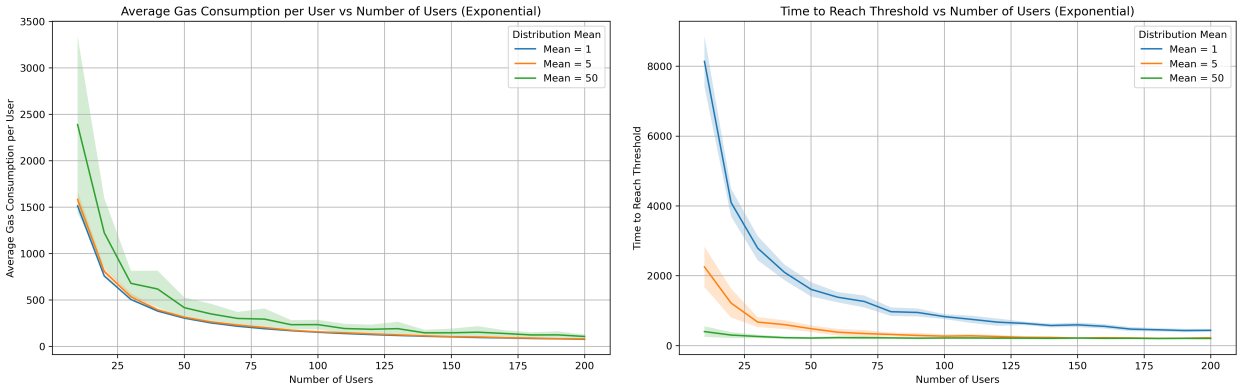

Fig. 5. Exponential Distribution for User Data Contribution: Average Gas consumption per user and Time to Threshold.

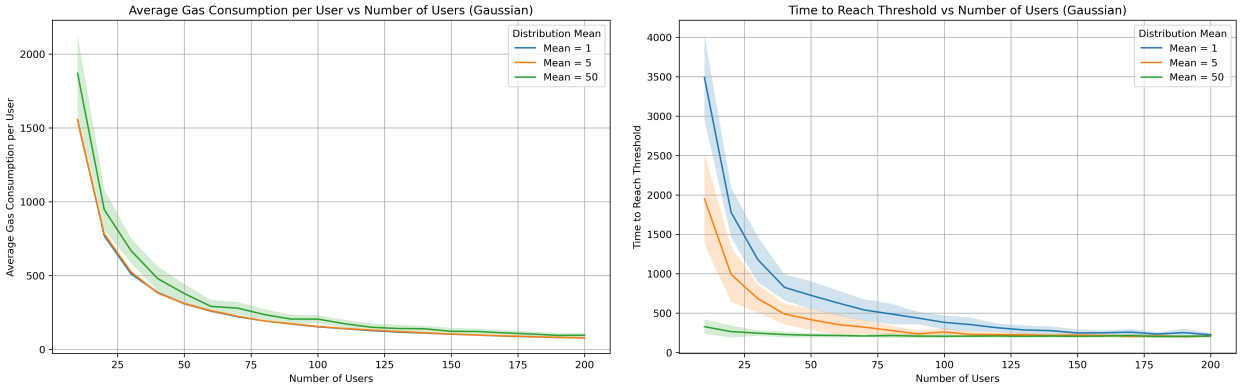

Fig. 6. Gaussian Distribution for User Data Contribution: Average Gas consumption per user and Time to Threshold.

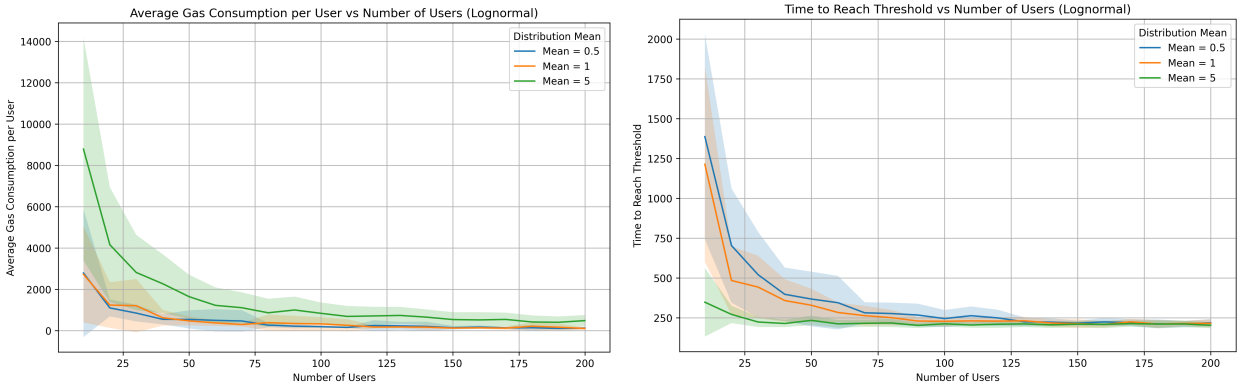

Fig. 7. Lognormal Distribution for User Data Contribution: Average Gas consumption per user and Time to Threshold.