









Byzantine Fault Detection in Swarm-SLAM Using Blockchain and Geometric Constraints

Angelo Moroncelli^{1,2,3} , Alexandre Pacheco¹ , Volker Strobel¹ ,
Pierre-Yves Lajoie⁴ , Marco Dorigo¹ , and Andreagiovanni Reina^{1,5,6} 

¹ IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
alexandre.melo.pacheco@gmail.com, volker.strobel@ulb.be,
mdorigo@ulb.ac.be

² DEIB, Politecnico di Milano, Milan, Italy

³ IDSIA, USI-SUPSI, Lugano, Switzerland
angelo.moroncelli@idsia.ch

⁴ Department of Software and Computer Engineering, Polytechnique Montréal,
Montreal, Canada

pierre-yves.lajoie@polymtl.ca

⁵ CASCB, Universität Konstanz, Konstanz, Germany
andreagiovanni.reina@uni-konstanz.de

⁶ Department of Collective Behaviour, Max Planck Institute of Animal Behavior,
Konstanz, Germany

Abstract. Effective methods for Simultaneous Localisation And Mapping (SLAM) are key to enabling autonomous robots to navigate unknown environments. Multi-robot collaborative SLAM (C-SLAM) offers the opportunity for higher performance thanks to parallel execution of mapping and localisation by a distributed team of robots but it also introduces challenges in system scalability and consistent data aggregation, exposing the system to potential security risks. In particular, we show that the state-of-the-art decentralised C-SLAM framework for swarm robotics is vulnerable to Byzantine robots, which are robots that behave incorrectly, possibly due to malfunctioning or hacking. We propose a solution that uses a blockchain to achieve data consistency and a smart contract that manages robots' reputations to identify and neutralise Byzantine robots. Each robot's contribution to collaborative mapping is peer-reviewed by other robots by verifying its correctness through geometric constraints. Our multi-robot simulation results show the existence of a trade-off between fault tolerance and efficiency in terms of map generation speed. With this work, we also release open-source research software that interfaces a custom blockchain with the ROS 2 framework.

1 Introduction

Autonomous robotic systems face the challenge of navigating unknown environments without relying on external localisation systems. To address this chal-

lenge, robots employ Simultaneous Localisation And Mapping (SLAM) algorithms [12, 47]. Through SLAM, robots build a map of the environment and determine their positions within this map [40]. Several efficient solutions to single-robot SLAM have been proposed thanks to decades of research that focused on this crucial topic [3, 9, 14, 30, 33, 36, 54]. More recently, research has started to investigate multi-robot collaborative SLAM (C-SLAM) [28], where groups of robots collaborate to build maps. Thanks to parallelisation, C-SLAM offers opportunities for increased efficiency, localisation accuracy, and robustness to errors.

A particularly promising type of multi-robot system is a robot swarm, which comprises typically a large number of autonomous robots. A key characteristic of robot swarms is decentralisation as robots only interact with their near neighbours and lack a centralised controller that orchestrates the actions of every robot [11]. To allow robot swarms to perform their operations, they must be able to navigate their environment, therefore implementing C-SLAM algorithms for robot swarms can be particularly useful [22, 23].

However, robot swarms introduce new challenges to C-SLAM due to the large number of robots participating in the process, and the lack of a central server that aggregates potentially conflicting data generated by different robots. Additionally, although robustness is often indicated as an intrinsic characteristic of swarm robotics, recent research [49, 50] has shown that robot redundancy and parallelisation of operations are not sufficient to achieve system robustness against misbehaving robots. In fact, a small proportion of misbehaving robots—called Byzantine robots—is often sufficient to disrupt the entire swarm system [2, 48, 52, 55]. Because it is reasonable to assume that a subset of robots may misbehave—for example, due to internal errors or external malicious tampering—implementing robust and secure algorithms is of utmost importance [13, 19, 21, 29, 41, 43].

This paper studies the potential security vulnerabilities of Swarm-SLAM [25], the state-of-the-art framework for C-SLAM with decentralised robot swarms (Sect. 2). We first discuss and characterise security issues that Swarm-SLAM, in particular, and C-SLAM, in general, face (Sect. 3). We show that Swarm-SLAM is highly vulnerable to the presence of different types of Byzantine robots. Inspired by recent research successes on protecting robot swarms via blockchain technology [4, 10, 17, 38, 42, 48–50, 50, 55], we build a security layer for Swarm-SLAM through a blockchain-based smart contract, which is a distributed tamper-proof algorithm running on data stored in the blockchain (Sect. 4). We test our solution with physics-based simulations of groups of eight robots using ROS 2 and a custom blockchain framework (Sect. 5). The results show that the proposed blockchain-based solution makes Swarm-SLAM tolerant to a relatively large number of Byzantine robots. However, this comes at the cost of a decrease in the map construction speed (i.e., lower system efficiency). In Sect. 6, we conclude the paper by discussing such a robustness-efficiency trade-off and suggesting potential future research in blockchain-based swarm robotics.

2 Background and Related Work

Multi-robot Collaborative SLAM. Individual robots use SLAM to autonomously explore and map unknown environments, but through collabora-

tion, multiple robots can more effectively navigate large spaces to create comprehensive maps. This multi-robot collaborative SLAM (C-SLAM) approach may mitigate exploration costs, map error, computational load, and single-point failure risks; but achieving this coordination is a complex task [6, 23]. Initially, single-robot SLAM algorithms were adapted for multi-robot use (e.g., by employing Kalman filters [44], or cooperative localisation algorithms [35]). Methods that formulate C-SLAM as a mathematical optimisation problem have become prevalent due to their higher performance than traditional C-SLAM methods that use filters to estimate the robots' poses and the map [45]. While recent research has shown great progress in C-SLAM methods (e.g., through advanced multi-source data fusion and deep learning to enhance adaptability and reduce the likelihood of failures [6]), most applications remain limited to small robot teams, and addressing the problems of perceptual aliasing [51], heterogeneous robot teams [5], and real-time distributed multi-robot coordination remain open challenges. Existing open-source frameworks for C-SLAM [5, 7, 8, 15, 20, 25, 27, 46, 51, 56] produce accurate results in the tested configurations but still have limitations in efficient data management, scalability to larger robot teams, and robustness against single points of failure, either because they use a centralised component to aggregate the maps and coordinate robots' movements or because the decentralised approach requires onboard computation by robots with computation and communication limits.

When we consider swarm robotics systems, Swarm-SLAM [25] stands out as a unique framework (based on the ROS 2 libraries [31]) to perform C-SLAM with a decentralised swarm of resource-limited robots. Swarm-SLAM outperforms other methods by allowing robots to use diverse sensors and operate with sporadic connectivity and significantly reduced communication demands. While Swarm-SLAM is a promising framework, there are still pending research questions on how to improve system scalability, achieve consistent data aggregation, and mitigate security risks. In this paper, we address the relatively unexplored problem of security in C-SLAM in general and Swarm-SLAM in particular. Indeed, when authors refer to Swarm-SLAM's robustness, they indicate the problem of perceptual aliasing [26]. However, in robotic systems operating in the real world, robots that exhibit non-ideal behaviour—e.g., due to faults or malicious intentions—may compromise the reliability of the entire system.

Securing Robot Swarms Using Blockchains. Swarm robotics, originally inspired by natural collectives, aims to create decentralised, robust, and scalable behaviour for groups of robots [11, 18]. However, recent research has shown that protecting the swarm against Byzantine robots can be difficult and requires dedicated strategies [21, 50]. A new and promising line of research suggests that blockchain-based smart contracts [53] can increase the Byzantine fault tolerance of robot swarms [10]. This research has shown that a solution to prevent the spreading of erroneous information is implementing a reputation management system where only information from high-reputation robots is used [48, 50, 52, 55]. Reputation management is implemented using a blockchain to record all robots' information exchanges and a smart contract to implement outlier detection algo-

gorithms and assign reputation to robots whose actions align with the majority. We build on these promising results to design a smart contract that exploits geometric constraints in the collectively constructed map to identify and neutralise Byzantine robots.

3 Vulnerability of Swarm-SLAM to Byzantine Robots

Swarm-SLAM can be compromised by Byzantine robots in three ways: two involve the corruption of loop closures and one consists of tampering with the creation of the pose graph, an optimisation process that creates a collective map based on existing loop closures. In single-robot SLAM, loop closures enhance map and self-localisation accuracy by detecting when a robot revisits the same location. In multi-robot C-SLAM, this process can become more accurate and faster when two different robots that visit the same location generate inter-robot loop closures. However, this exchanging of information makes the system vulnerable to incorrect data sent by Byzantine robots. Current C-SLAM systems are not resilient to incorrect data points that significantly deviate from the overall map being constructed. Although some systems incorporate techniques to negate the effects of perceptual outliers [5, 8, 15, 20, 25, 27, 51], these systems remain vulnerable to Byzantine robots that conspire to overcome the rejection of these outliers, or to Sybil attacks, in which a single Byzantine robot forges many identities to gain control of the system. Protecting against these attacks is crucial, yet research on Byzantine fault tolerance in C-SLAM is lacking.

In C-SLAM, an inter-robot loop closure happens when robot A recognises a scene that another robot B saw previously. When they meet, B shares an image of the scene, which is then used by A to calculate a geometric loop closure. We found that a loop closure can have two main sources of error: (1) incorrect calculation by A (using correct information sent by B), or (2) incorrect information sent by B (while A makes the correct computation based on the received information).

The third security issue concerns the pose graph optimisation (PGO) management, which is a crucial step in merging the partial maps acquired by each robot. The pose graph is the mathematical representation of the relationships between robot poses during their trajectories and the scenes in the map. The configuration of this graph that best satisfies all the constraints can subsequently be used for building a map and simultaneously correcting the location of the robot poses inside the map. Always assigning the task of performing PGO to the same robot creates a single point of failure where a Byzantine robot in this role could compromise the entire SLAM operation, while on the opposite end, having all the robots perform the PGO could be unnecessarily costly. Figure 1 illustrates a comparison between a correct Swarm-SLAM process and the three scenarios where the actions of a Byzantine robot can lead to a wrong outcome.

4 Securing Swarm-SLAM Through Blockchain Technology

We employ blockchain technology to enhance the Byzantine fault tolerance and reliability of Swarm-SLAM. Blockchain is a shared, tamper-proof ledger

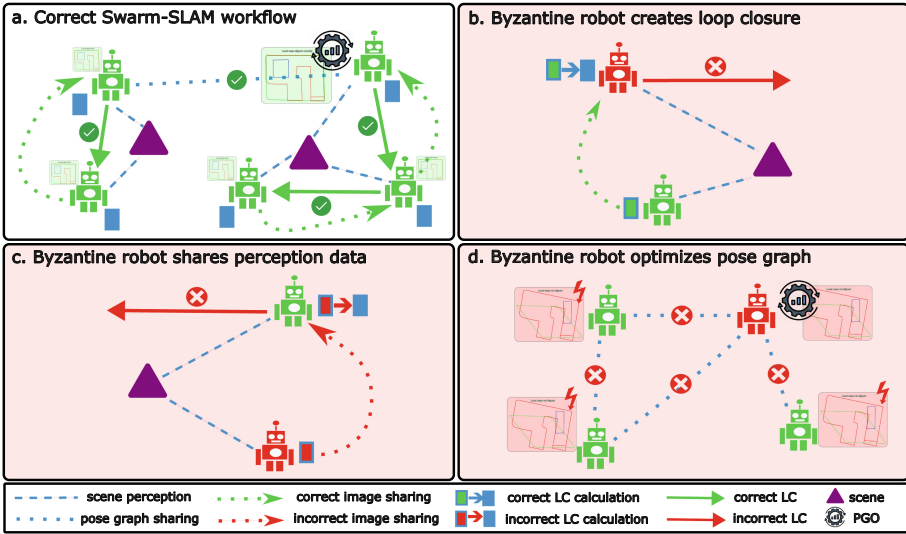


Fig. 1. Panel a represents the ideal workflow in a Swarm-SLAM application with reliable (green, non-Byzantine) robots, where robots acquire information about the same scene (purple triangle). When two of these robots meet, they have two matching descriptors for the scene and thus one of the robots (the receiver robot) shares its image (blue square) that has the matching descriptor with the sender robot, which calculates an inter-robot loop closure towards the receiver. These loop closures are then used by one robot (top right) to perform pose graph optimisation (PGO). Panels b-d depict three possible scenarios where the action of a Byzantine robot leads to incorrect global pose graph results. In panel b the sender is a Byzantine (red robot) which generates an incorrect loop closure (pointing to a wrong position) from itself to the receiver (green robot), while the image is shared correctly from the receiver. Panel c represents the construction of an incorrect loop closure due to false information shared by the Byzantine receiver robot (red robot): it shares an incorrect image with the sender, which calculates a wrong loop closure although being a reliable robot. Panel d depicts the action of PGO, which, in Swarm-SLAM, is performed by one single elected robot (in this case the optimiser is the red Byzantine robot). A Byzantine robot performing the PGO can introduce significant errors in the pose graph which is then shared throughout the robot swarm. Once an incorrect loop closure is introduced in the PGO, this error cannot be recovered nor verified by other robots. (Colour figure online)

that enables secure information storage and decentralised transaction validation [34]. This removes the need for a central authority, thus offering the opportunity to manage robots’ permissions and reputation in swarm robotics [10]. Blockchain technology offers several benefits that are key to improving Swarm-SLAM’s reliability and Byzantine fault tolerance: decentralisation eliminates single points of failure, tamper-resistance protects data integrity, and smart contracts enable the execution of algorithms among untrusting agents. Blockchain technology can have a limited computational overhead, making it also suitable

for resource-constrained devices such as robots, as demonstrated in previous studies [38, 39, 50].

The integration of blockchain with Swarm-SLAM consists of a smart contract that validates loop closures and manages robots’ reputations based on the correctness of the contributed loop closures. This smart contract uses geometric relationships to identify and reject incorrect loop closures, preventing the injection of faulty data in the construction of the collaborative map, whether they are intentionally wrong or by mistake. While this step introduces some latency, validating loop closures before they are utilised in the PGO can dramatically improve map accuracy in the presence of Byzantine robots.

An important aspect of Swarm-SLAM is that it minimises communication costs by requiring only one robot to share its (usually large) raw data [25]. The receiving robot then uses this data to compute the loop closure, which is directly used for PGO, making it difficult to assess the correctness of both the data and the loop closure. With our solution, loop closures are first stored in the blockchain, and only used in PGO once they are validated by other robots (see below). As the blockchain stores loop closures, which are lightweight geometric transformations, the system’s scalability is improved by avoiding redundant sharing of raw image data. Storage requirements can also be reduced by letting robots delete the raw data once a corresponding loop closure is stored in the blockchain.

Our main contribution is a blockchain-based smart contract that leverages geometric constraints to validate loop closures in Swarm-SLAM and address the threat of Byzantine robots generating incorrect loop closures (Figs. 1b-c). To validate a loop closure, a second and a third loop closure are required to establish a triangle, as shown in Fig. 2. The triangle identity can then be used as a geometric constraint to validate the loop closures. The presence of more inter-robot loop closures from different robots leads to the formation of multiple triangles, thus increasing the confidence in the correctness of the loop closures.

By creating blockchain transactions, robots can store their loop closures in the blockchain, triggering the smart contract to evaluate whether the new loop closure formed any new validation triangle. The triangle identity constraint is validated when: $l_x^i + l_x^j + l_x^k < \epsilon$ and $l_y^i + l_y^j + l_y^k < \epsilon$, where l_x^i is robot i ’s loop closure on the x coordinate and ϵ is a sensitivity threshold. The threshold ϵ is a parameter to set the maximum Euclidean distance between the transformation head and tail—which should ideally match at each vertex—controlling the smart contract’s sensitivity to errors in loop closure calculations. This threshold defines what the smart contract labels as Byzantine (above ϵ) and what it labels as noise (below ϵ and accepted as triangle identity). The best threshold largely depends on the application in which the algorithm is used. The three loop closures involved in a validation triangle must be submitted by three different robots (possibly at different times) and follow a cyclic orientation of transformations where each robot is only once a receiver and once a sender (see Fig. 2c). In this way, we promote peer-to-peer validation and prevent a single Byzantine robot from inputting multiple incorrect (but consistent) loop closures that pass

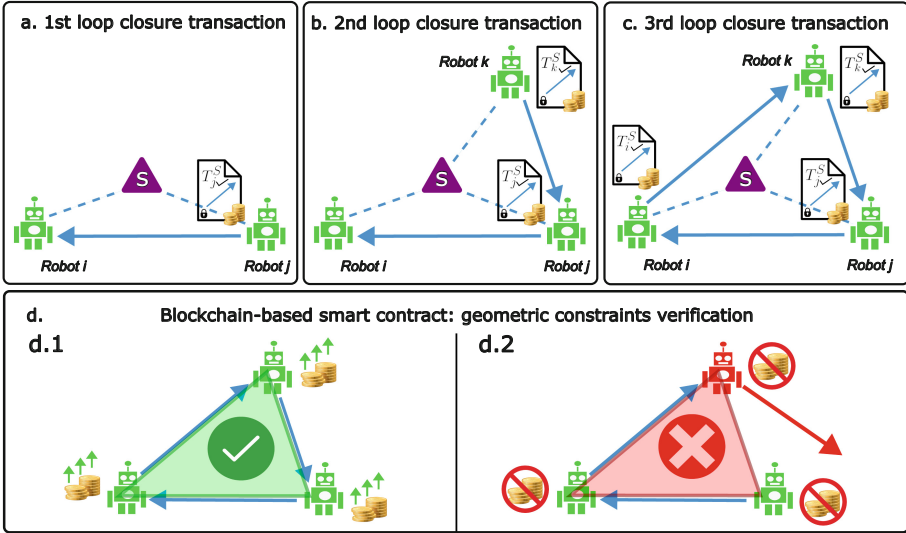


Fig. 2. The figure depicts the process by which our smart contract validates loop closures (LC). The blue arrows correspond to correct LCs proposed by reliable robots (green robots), while the red arrow illustrates an incorrect inter-robot LC created by a Byzantine robot (red robot). (a-c) The robots j , k , and i make blockchain transactions (T_j^S , T_k^S , and T_i^S , respectively) which contain the proposed LC (blue arrow) for the purple scene S and some authorisation tokens that the robots must deposit along with every transaction. (d) There are two possible outcomes of a LC validation. (d.1) The triangle identity is validated and the robots receive back their authorisation tokens and one reputation token. (d.2) The triangle is rejected and none of the robots receives back their reputation tokens, since it is not possible to infer which robot is Byzantine. (Colour figure online)

the geometric test. However, two colluding Byzantine robots could bypass this protection mechanism. For this reason, we introduce a parameter called *security level* (discussed below) to increase security against collusion. During PGO, only the individual robots’ maps linked by validated inter-robot loop closures are merged into a global map. Hence, our framework also prevents the use of unreliable Byzantine robot’s trajectories for map generation because our triangle validation mechanism ensures that Byzantine-generated loop closures remain invalidated, thus preventing their inclusion in the PGO and the global map.

Reputation Management Mechanism. Besides protecting the process from the injection of incorrect loop closures, our smart contract is also designed to identify and neutralise Byzantine robots. It does so through a reputation management mechanism based on crypto tokens which are scarce digital tokens stored in the blockchain. Each robot starts with a given amount of crypto tokens loaded in its blockchain wallet (accessed with standard public-key encryption). The

blockchain stores the history of every token transaction between any wallet. As every robot maintains a synchronised copy of this ledger, the reputation (crypto tokens) of every robot is publicly known. In our implementation, we employ two types of crypto tokens: authorisation tokens and reputation tokens. The former are tokens that each robot needs to deposit to make a transaction through which a new loop closure is proposed (Fig. 2a-c). The deposited authorisation tokens are withheld by the smart contract and returned to the robot only once the loop closure is validated (Fig. 2d). This mechanism fixes the maximum number of unvalidated loop closures that a robot can have, preventing Byzantine robots from flooding the blockchain with incorrect loop closures as they will eventually deplete their authorisation tokens. In this way, robots submitting incorrect information are neutralised as they run out of authorisation tokens. Because only a set of designated wallets (one per robot) can receive authorisation tokens, the system is protected from Sybil attacks, in which a single robot could validate its own loop closures by using different identities.

The identification of Byzantine robots can be achieved through reputation tokens which are emitted by the smart contract every time a triangle identity validates three loop closures. Each of the three robots that submitted these loop closures receives one reputation token, which is publicly stored in the blockchain. Robots that do not increase their reputation for a long period may be identified as Byzantine.

Security Level Parameter. The security level indicates how many times an inter-robot loop closure has been validated by a different triplet of robots and hence, how secure it is. When a new loop closure is proposed, its security level is zero. When the loop closure becomes part of a validation triangle with other two loop closures proposed by two different robots, its security level is set to one. Each time the loop closure is included in other validation triangles involving different robot pairs, its security level increases by one. In our work, as soon as a loop closure reaches the security level one, it is included in the PGO and the smart contract returns the deposited authorisation tokens to the robot. This means that we secured the system from individual Byzantine robots, but not from colluding ones. However, the minimum security level can be increased to protect the system against potential collusion of Byzantine robots, albeit at the cost of a higher latency. The smart contract also gives a reputation token to the robots each time their loop closures lead to a security level increment. Therefore, robots that submit valid loop closures increase their reputation over time as more peers validate the loop closures, increasing their security level.

5 Results

We test our approach through a series of 40-minute-long simulation experiments. The simulations are run in the Gazebo simulator [24], with teams of 8 TurtleBots3-Waffle robots [1] collaboratively mapping an environment sized

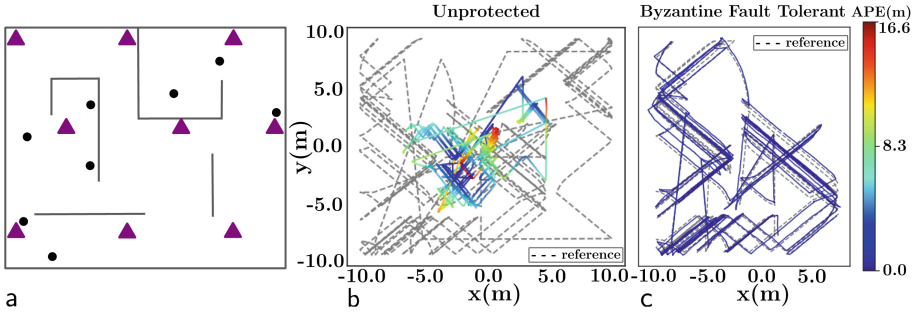


Fig. 3. (a) The tested environment with size $20 \times 22 \text{ m}^2$ and 9 scenes (purple triangles). The walls are depicted as black lines and the 8 TurtleBots3 robots as black dots. (b-c) Comparison between the ground truth (dashed grey lines) and the aggregated robot trajectories resulting from PGO of a representative run with 3 Byzantine robots that introduce loop closures with a constant error of 10 m. The trajectories are colour-coded (see right colour bar) indicating the error (APE) of each point. Swarm-SLAM without our blockchain-based protection layer produces trajectories with high error, whereas the error is close to zero when loop closures are validated by the smart contract. (Colour figure online)

$20 \times 22 \text{ m}^2$ with various walls separating the space and 9 scenes used for inter-robot loop closures (see Fig. 3a). Robots move through a random walk consisting of straight motion interrupted by random turns when an obstacle is detected at a distance smaller than 0.5 m; two robots can communicate within a maximum range of 5 m. Each robot acts as a blockchain node which maintains the blockchain and generates new blocks following the Proof-of-Authority consensus protocol. We employ the Toychain blockchain [37], a simple Python-based blockchain designed for scientific research. We integrated Swarm-SLAM using a smart contract that handles the security checks and crypto token distribution. The complete simulation software is open-source and available in two packages that are accessible from our project repository¹.

To measure the system’s robustness, we compute two metrics using the *evo* software [16]: the Absolute Positional Error (APE) as the Euclidean distance between each point in the robot trajectories and the ground truth, and the Root Mean Square Error (RMSE) of these distances. We consider two types of Byzantine robots that differ in the error they apply to the loop closures. In both cases, Byzantine robots add a value—represented by a vector (ℓ_x, ℓ_y) —to the correct loop closure before broadcasting it to the other robots. In Fig. 4, we report the results for the case when Byzantine robots add a *constant* value $+10 \text{ m}$ to each vector component. In the supplementary material [32], we report the results for Byzantine robots that add a *random* value drawn from a uniform distribution $\mathcal{U}[-9, 9] \text{ m}$ to each vector component (ℓ_x, ℓ_y) . In both cases, the system without our protection layer suffers large errors as soon as one Byzantine robot is intro-

¹ <https://github.com/clmoro/Blockchain-Based-BFT-Swarm-SLAM.git>.

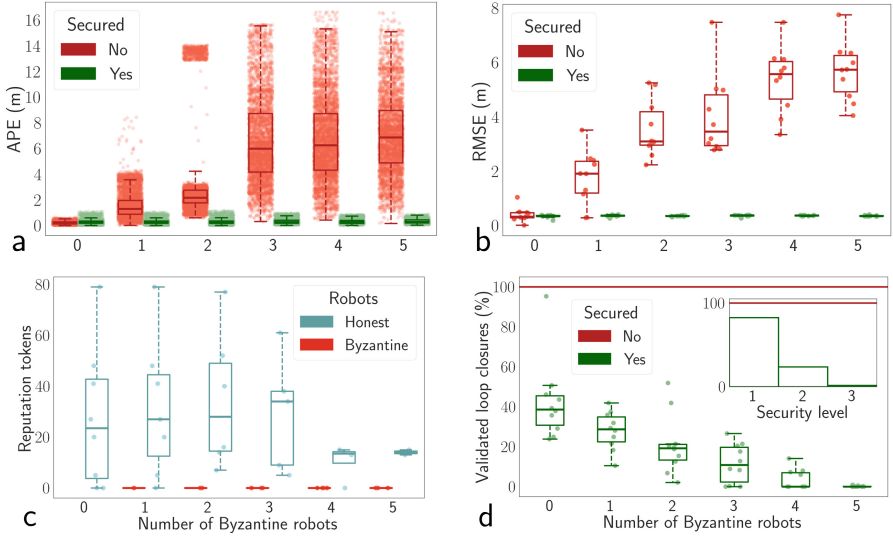


Fig. 4. (a-b) Comparison of the error in the aggregated robot trajectories after PGO for different numbers of Byzantine robots (x-axis) in a swarm of 8 robots using the original Swarm-SLAM (unsecured) or the Byzantine fault tolerant Swarm-SLAM (secured with smart contract). The boxplots in (a) show the APE for one representative simulation experiment while in (b) the RMSE is computed over 10 simulation runs (in the same environment but with different starting conditions). (c) Reputation tokens at the end (minute 40) of one representative experiment for Byzantines and non-Byzantine robots. The red boxes (on the left of each green box) are always flat at zero. (d) Proportion of validated loop closures that are used in the Swarm-SLAM’s PGO (results for 10 simulation runs for each condition). The red line indicates that 100% of the proposed loop closures are used in PGO in the original Swarm-SLAM. In all panels, we show both the raw data (individual points) and the aggregated data distribution as boxplots (with interquartile range IQR box, median line, and whiskers to data within 1.5 IQR). The inset in d shows the proportion of loop closures reaching security level 1, 2, or 3 in the absence of Byzantine robots. The data for the inset are collected from one 45-min long run, independent from the main figure. (Colour figure online)

duced in the swarm, and the error increases with the number of Byzantine robots (Figs. 4a-b). Instead, when our blockchain-based smart contract secures the system, the error remains close to zero even when 5 of the 8 robots are Byzantine. The error is, however, never exactly zero because robots are subject to odometry noise. This odometry noise is smaller when there are more non-Byzantine robots (which compensate for each other’s noise). Figures 3b-c show two examples of the accumulated APE on the aggregated robot trajectories with 3 Byzantine robots.

We also measure how the reputation tokens are distributed between Byzantine and non-Byzantine robots. Figure 4c shows that the number of reputation

tokens at the end of the simulation allows distinguishing between the two types of robots as the Byzantine robots never receive any reputation token (red boxplot is flat at zero) because, in our experiments, they always submit incorrect loop closures that never get validated. Figure 4d shows that our security layer comes at the cost of reduced speed as the number of loop closures processed by the PGO is lower than half even in the absence of any Byzantine robots. We recall that the PGO only processes loop closures validated by the smart contract (y-axis of Fig. 4d). After 40 min, only a portion of loop closures are validated (and thus processed by the PGO), meaning that the other pending loop closures will be validated at a later time (thus adding some latency between the creation of an inter-robot loop closure and its use in the PGO). As the number of Byzantine robots increases, the number of validated loop closures decreases because fewer non-Byzantine robots contribute with correct data and forming loop-closure triangles becomes slower. Increasing the security level (i.e., number of validations before using the loop closure in the PGO) is another factor negatively impacting the system efficiency, yet protecting the system against Byzantine robot collusion. Figure 4d’s inset shows that most loop closures achieve security level 1, however reaching levels 2 and 3 is less frequent in a swarm of 8 robots as each loop closure must be validated by more than half of the swarm.

6 Discussion and Conclusion

Swarm-SLAM [25] is a promising framework to enable robot swarms to perform decentralised collaborative mapping of unknown environments. However, our analysis shows that Swarm-SLAM is highly vulnerable to the presence of even a single Byzantine robot that shares incorrect loop closures or tampers with the pose graph optimisation (PGO) process. Through a blockchain-based smart contract that uses geometric constraints among loop closures to check their validity before using them to build the collective map (PGO step), we considerably improve Swarm-SLAM’s security against Byzantine robots. However, this increased security comes at the cost of an increased latency between the computation of a loop closure and its use in the PGO, reducing the system’s mapping speed. The proposed method allows both the identification and the neutralisation of Byzantine robots through the use of two types of crypto tokens that assign reputation (for identification) and rights to participate (for neutralisation).

Future research should investigate situations in which (i) Byzantine robots collude with each other to validate incorrect loop closures and (ii) Byzantine robots dynamically change their behaviour, proposing a mix of correct and incorrect loop closures. While we expect that dynamic Byzantines are harder to identify and neutralise, we expect that our solution will still prevent them from corrupting the map generation. Future research should also extend the smart contract to protect the system against PGO tampering through peer-reviewing of each other contributions, similar to our proposed loop closure peer validation.

Acknowledgements. We thank Miquel Kegeleirs, David Garzón Ramos, and Guillermo Legarda Herranz for the helpful discussions. V.S. and M.D. acknowledge support from the Belgian F.R.S.-FNRS. A.R. acknowledges support from DFG under Germany’s Excellence Strategy - EXC 2117-422037984.

References

1. Amsters, R., Slaets, P.: Turtlebot 3 as a robotics education platform. In: Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., Obdržálek, D. (eds.) RiE 2019. AISC, vol. 1023, pp. 170–181. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-26945-6_16
2. Aswale, A., López, A., Ammartayakun, A., Pincioli, C.: Hacking the colony: on the disruptive effect of misleading pheromone and how to defend against it. In: AAMAS 2022: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, pp. 27–34. IFAAMAS, Richland, SC (2022)
3. Ayache, N., Faugeras, O.: Building, registering and fusing noisy visual maps. *Int. J. Robot. Res.* **7**(6), 45–65 (1988). <https://doi.org/10.1177/027836498800700605>
4. Campos, M., Chanel, C., Chauffaut, C., Lacan, J.: Towards a blockchain-based multi-UAV surveillance system. *Front. Robot. AI* **8**, 557692 (2021). <https://doi.org/10.3389/frobt.2021.557692>
5. Chang, Y., et al.: LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments. *IEEE Robot. Autom. Lett.* 9175–9182 (2022). <https://doi.org/10.1109/LRA.2022.3191204>
6. Chen, W., et al.: Overview of multi-robot collaborative SLAM from the perspective of data fusion. *Machines* **11**(6), 653 (2023). <https://doi.org/10.3390/machines11060653>
7. Cieslewski, T., Choudhary, S., Scaramuzza, D.: Data-efficient decentralized visual SLAM. In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2466–2473. IEEE (2018). <https://doi.org/10.1109/ICRA.2018.8461155>
8. Cramariuc, A., et al.: maplab 2.0 – a modular and multi-modal mapping framework. *IEEE Robot. Autom. Lett.* **8**(2), 520–527 (2023). <https://doi.org/10.1109/lra.2022.3227865>
9. Crowley, J.L.: World modeling and position estimation for a mobile robot using ultrasonic ranging. In: Proceedings of the 1989 International Conference on Robotics and Automation (ICRA), vol. 2, pp. 674–680 (1989). <https://doi.org/10.1109/ROBOT.1989.100062>
10. Dorigo, M., Pacheco, A., Reina, A., Strobel, V.: Blockchain technology for mobile multi-robot systems. *Nat. Rev. Electr. Eng.* **1**(4), 264–274 (2024). <https://doi.org/10.1038/s44287-024-00034-9>
11. Dorigo, M., Theraulaz, G., Trianni, V.: Swarm robotics: past, present, and future [point of view]. *Proc. IEEE* **109**(7), 1152–1165 (2021). <https://doi.org/10.1109/JPROC.2021.3072740>
12. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part i. *IEEE Robot. Autom. Mag.* **13**(2), 99–110 (2006). <https://doi.org/10.1109/MRA.2006.1638022>
13. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. *J. ACM* **35**(2), 288–323 (1988). <https://doi.org/10.1145/42282.42283>

14. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 834–849. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10605-2_54
15. Fernandez-Cortizas, M., Bavle, H., Perez-Saura, D., Sanchez-Lopez, J.L., Campoy, P., Voos, H.: Multi S-graphs: an efficient distributed semantic-relational collaborative SLAM. *IEEE Robot. Autom. Lett.* **9**(6), 6004–6011 (2022). <https://doi.org/10.1109/LRA.2024.3399997>
16. Grupp, M.: EVO: python package for the evaluation of odometry and SLAM. <https://github.com/MichaelGrupp/evo> (2017)
17. Guerrero-Bonilla, L., Prorok, A., Kumar, V.: Formations for resilient robot teams. *IEEE Robot. Autom. Lett.* **2**, 841–848 (2017). <https://doi.org/10.1109/LRA.2017.2654550>
18. Hamann, H.: *Swarm Robotics: A Formal Approach*. Springer, Cham (2018). <https://doi.org/10.1007/978-3-319-74528-2>
19. Higgins, F., Tomlinson, A., Martin, K.M.: Survey on security challenges for swarm robotics. In: 2009 Fifth International Conference on Autonomic and Autonomous Systems, pp. 307–312. IEEE (2009). <https://doi.org/10.1109/ICAS.2009.62>
20. Huang, Y., Shan, T., Chen, F., Englot, B.: DiSCo-SLAM: distributed scan context-enabled multi-robot LiDAR SLAM with two-stage global-local graph optimization. *IEEE Robot. Autom. Lett.* **7**(2), 1150–1157 (2022). <https://doi.org/10.1109/LRA.2021.3138156>
21. Hunt, E., Hauert, S.: A checklist for safe robot swarms. *Nat. Mach. Intell.* **2**, 420–422 (2020). <https://doi.org/10.1038/s42256-020-0213-2>
22. Kegeleirs, M., Garzón Ramos, D., Birattari, M.: Random walk exploration for swarm mapping. In: Althoefer, K., Konstantinova, J., Zhang, K. (eds.) TAROS 2019. LNCS (LNAI), vol. 11650, pp. 211–222. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25332-5_19
23. Kegeleirs, M., Grisetti, G., Birattari, M.: Swarm SLAM: challenges and perspectives. *Front. Robot. AI* **8**, 618268 (2021). <https://doi.org/10.3389/frobt.2021.618268>
24. Koenig, N., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2149–2154 (2004). <https://doi.org/10.1109/IROS.2004.1389727>
25. Lajoie, P.Y., Beltrame, G.: Swarm-slam: sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE Robot. Autom. Lett.* **9**(1), 475–482 (2024). <https://doi.org/10.1109/LRA.2023.3333742>
26. Lajoie, P.Y., Hu, S., Beltrame, G., Carlone, L.: Modeling perceptual aliasing in SLAM via discrete–continuous graphical models. *IEEE Robot. Autom. Lett.* **4**(2), 1232–1239 (2019). <https://doi.org/10.1109/lra.2019.2894852>
27. Lajoie, P.Y., Ramtoula, B., Chang, Y., Carlone, L., Beltrame, G.: DOOR-SLAM: distributed, online, and outlier resilient SLAM for robotic teams. *IEEE Robot. Autom. Lett.* **5**(2), 1656–1663 (2020). <https://doi.org/10.1109/lra.2020.2967681>
28. Lajoie, P.Y., Ramtoula, B., Wu, F., Beltrame, G.: Towards collaborative simultaneous localization and mapping: a survey of the current research landscape. *Field Robot.* **2**(1), 971–1000 (2022). <https://doi.org/10.55417/fr.2022032>
29. Lamport, L., Shostak, R., Pease, M.: The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* **4**(3), 382–401 (1982). <https://doi.org/10.1145/357172.357176>

30. Lourakis, M., Argyros, A.: SBA: a software package for generic sparse bundle adjustment. *ACM Trans. Math. Softw.* **36**(1), 1–30 (2009). <https://doi.org/10.1145/1486525.1486527>
31. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: design, architecture, and uses in the wild. *Sci. Robot.* **7**(66), eabm6074 (2022). <https://doi.org/10.1126/scirobotics.abm6074>
32. Moroncelli, A., Pacheco, A., Strobel, V., Lajoie, P.Y., Dorigo, M., Reina, A.: Supplementary material for the paper: byzantine fault detection in swarm-SLAM using blockchain and geometric constraints (2024). <https://sites.google.com/view/bft-swarm-slam>
33. Mur-Artal, R., Montiel, J., Tardos, J.: ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. Rob.* **31**(5), 1147–1163 (2015). <https://doi.org/10.1109/TRO.2015.2463671>
34. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. electronic document. <http://www.bitcoin.org> (2008)
35. Nerurkar, E.D., Roumeliotis, S.I., Martinelli, A.: Distributed maximum a posteriori estimation for multi-robot cooperative localization. In: *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1402–1409. IEEE (2009). <https://doi.org/10.1109/ROBOT.2009.5152398>
36. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. In: *2011 International Conference on Computer Vision*, pp. 2320–2327 (2011). <https://doi.org/10.1109/ICCV.2011.6126513>
37. Pacheco, A., Denis, U., Zakir, R., Strobel, V., Reina, A., Dorigo, M.: Toychain: a simple blockchain for research in swarm robotics (2024). <https://doi.org/10.48550/arXiv.2407.06630>, arXiv preprint:2407.06630 [cs.RO]
38. Pacheco, A., Strobel, V., Dorigo, M.: A blockchain-controlled physical robot swarm communicating via an Ad-Hoc network. In: Dorigo, M., et al. (eds.) *ANTS 2020. LNCS*, vol. 12421, pp. 3–15. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60376-2_1
39. Pacheco, A., Strobel, V., Reina, A., Dorigo, M.: Real-time coordination of a foraging robot swarm using blockchain smart contracts. In: Dorigo, M., et al. *Swarm Intelligence, ANTS 2022, LNCS*, vol. 13491, pp. 196–208. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20176-9_16
40. Placed, J.A., et al.: A survey on active simultaneous localization and mapping: state of the art and new frontiers. *IEEE Trans. Robot.* **39**(3), 1686–1705 (2022). <https://doi.org/10.1109/TRO.2023.3248510>
41. Prorok, A., Malencia, M., Carlone, L., Sukhatme, G.S., Sadler, B.M., Kumar, V.: Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems (2021). <https://doi.org/10.48550/arXiv.2109.12343>, arXiv preprint:2109.12343 [cs.RO]
42. Queralta Peña, J., Qingqing, L., Zou, Z., Westerlund, T.: Enhancing autonomy with blockchain and multi-access edge computing in distributed robotic systems. In: *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 180–187. IEEE (2020). <https://doi.org/10.1109/FMEC49853.2020.9144809>
43. Reina, A.: Robot teams stay safe with blockchains. *Nat. Mach. Intell.* **2**, 240–241 (2020). <https://doi.org/10.1038/s42256-020-0178-1>
44. Rodriguez-Losada, D., Matia, F., Jimenez, A.: Local maps fusion for real time multirobot indoor simultaneous localization and mapping. In: *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1308–1313. IEEE (2004). <https://doi.org/10.1109/ROBOT.2004.1308005>

45. Saeedi, S., Trentini, M., Seto, M., Li, H.: Multiple-robot simultaneous localization and mapping: a review. *J. Field Robot.* **33**(1), 3–46 (2016). <https://doi.org/10.1002/rob.21620>
46. Schmuck, P., Ziegler, T., Karrer, M., Perraudin, J., Chli, M.: COVINS: visual-inertial SLAM for centralized collaboration. In: 2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 171–176. IEEE Computer Society, Los Alamitos, CA, USA (2021). <https://doi.org/10.1109/ISMAR-Adjunct54149.2021.00043>
47. Smith, R.C., Cheeseman, P.: On the representation and estimation of spatial uncertainty. *The Int. J. Robot. Res.* **5**(4), 56–68 (1986). <https://doi.org/10.1177/027836498600500404>
48. Strobel, V., Castelló Ferrer, E., Dorigo, M.: Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In: Proceedings of 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, pp. 541–549. IFAAMAS, Richland, SC (2018)
49. Strobel, V., Castelló Ferrer, E., Dorigo, M.: Blockchain technology secures robot swarms: a comparison of consensus protocols and their resilience to byzantine robots. *Front. Robot. AI* **7**, 54 (2020). <https://doi.org/10.3389/frobt.2020.00054>
50. Strobel, V., Pacheco, A., Dorigo, M.: Robot swarms neutralize harmful Byzantine robots using a blockchain-based token economy. *Sci. Robot.* **8**(79), eabm4636 (2023). <https://doi.org/10.1126/scirobotics.abm4636>
51. Tian, Y., Chang, Y., Herrera Arias, F., Nieto-Granda, C., How, J.P., Carlone, L.: Kimera-multi: robust, distributed, dense metric-semantic SLAM for multi-robot systems. *IEEE Trans. Rob.* **38**(4), 2022–2038 (2022). <https://doi.org/10.1109/TRO.2021.3137751>
52. Van Calck, L., Pacheco, A., Strobel, V., Dorigo, M., Reina, A.: A blockchain-based information market to incentivise cooperation in swarms of self-interested robots. *Sci. Rep.* **13**, 20417 (2023). <https://doi.org/10.1038/s41598-023-46238-1>
53. Wood, G.: Ethereum: a secure decentralized generalised transaction ledger. *Ethereum Found.* **151**, 1–41 (2014). <https://ethereum.github.io/yellowpaper/paper.pdf>
54. Zhang, Y., Wu, Y., Tong, K., Chen, H., Yuan, Y.: Review of visual simultaneous localization and mapping based on deep learning. *Remote Sens.* **15**(11), 2740 (2023). <https://doi.org/10.3390/rs15112740>
55. Zhao, H., et al.: A generic framework for byzantine-tolerant consensus achievement in robot swarms. In: Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8839–8846. IEEE (2023). <https://doi.org/10.1109/IROS55552.2023.10341423>
56. Zhong, S., Qi, Y., Chen, Z., Wu, J., Chen, H., Liu, M.: DCL-SLAM: a distributed collaborative LiDAR SLAM framework for a robotic swarm. *IEEE Sens. J.* **24**(4), 4786–4797 (2024). <https://doi.org/10.1109/JSEN.2023.3345541>