COLLECTIVE BEHAVIOR

Robot swarms neutralize harmful Byzantine robots using a blockchain-based token economy

Volker Strobel*, Alexandre Pacheco, Marco Dorigo

Through cooperation, robot swarms can perform tasks or solve problems that a single robot from the swarm could not perform/solve by itself. However, it has been shown that a single Byzantine robot (such as a malfunctioning or malicious robot) can disrupt the coordination strategy of the entire swarm. Therefore, a versatile swarm robotics framework that addresses security issues in inter-robot communication and coordination is urgently needed. Here, we show that security issues can be addressed by setting up a token economy between the robots. To create and maintain the token economy, we used blockchain technology, originally developed for the digital currency Bitcoin. The robots were given crypto tokens that allowed them to participate in the swarm's security-critical activities. The token economy was regulated via a smart contract that decided how to distribute crypto tokens among the robots depending on their contributions. We designed the smart contract so that Byzantine robots soon ran out of crypto tokens and could therefore no longer influence the rest of the swarm. In experiments with up to 24 physical robots, we demonstrated that our smart contract approach worked: The robots could maintain blockchain networks, and a blockchain-based token economy could be used to neutralize the destructive actions of Byzantine robots in a collective-sensing scenario. In experiments with more than 100 simulated robots, we studied the scalability and long-term behavior of our approach. The obtained results demonstrate the feasibility and viability of blockchain-based swarm robotics.

INTRODUCTION

A robot swarm is a multirobot system consisting of many autonomous robots whose self-organized collective behavior results from local interactions of each robot with its peers and with the environment (*1*–*6*). The deployment of robot swarms in the real world has been identified as one of the 10 grand challenges of robotics for the next decade (*7*). Once they are deployed in the real world, however, robots in a swarm will be subject to events that can harm the swarm's collective behavior: The robots in the swarm could malfunction, could become fully nonoperational, or might be hacked by malicious agents. In addition, peer-to-peer messages could be manipulated while traveling through the network (*8*).

An original and pivotal assumption of the swarm robotics research field has been that a large number of robots and their decentralized organization are sufficient to attain both robustness against malfunctioning robots and security against the presence of malicious robots, thanks to the property of fault tolerance by redundancy (*9*, *10*). A large body of research has shown that the effects of malfunctioning robots on swarm behavior can be managed (*11*–*18*). However, this is not the case in the presence of malicious robots: Recent research has shown that even a very small minority of such robots can disrupt the functioning of the whole swarm (*19*, *20*).

Although security issues are of paramount importance for the deployment of robot swarms in the real world, only a few research papers have been published on this topic so far. One recent important contribution has proposed a method to give commands to robots in a swarm without revealing the overall goal of the mission (*21*). A cooperative robot mission is encapsulated in a Merkle tree, a data structure in which the information in the nodes consists of cryptographic hashes. It thus becomes possible to provide the blueprint of a robot mission without disclosing the raw information describing the mission itself. Information about the swarm's goals cannot be acquired by listening to robot communications or physically capturing a robot. Another recent important contribution has shown that a probabilistic component in the consensus algorithm can minimize the effects of deceitful messages in a binary decision-making scenario (*22*, *23*). This damage mitigation occurs implicitly; that is, the approach minimizes the harmful effects without determining whether or how many misbehaving robots are in the swarm. In practice, security research in swarm robotics is in its infancy, and further research on this subject is urgently needed, as mentioned in a number of relatively recent articles (*6*, *8*, *24*, *25*).

An important concept for cybersecurity research is the notion of Byzantine agents (*26*) or Byzantine robots (*19*). We call an agent or a robot Byzantine if it shows a discrepancy between its intended behavior and its actual behavior. This discrepancy can be a result of programming errors, failed components, or malicious attacks (*27*).

In peer-to-peer networks, the problem of Byzantine agents has been successfully addressed by blockchain technology. This technology combines cryptographic algorithms and decentralized consensus protocols to allow agents to securely share and store data in networks that may contain Byzantine agents, as is the case in robot swarms. The Bitcoin protocol (*28*) was the first successful implementation of a tamper-proof decentralized digital currency. In the Bitcoin protocol, a blockchain hosts a decentralized ledger of Bitcoin transactions. Other blockchain protocols, most notably Ethereum (*29*), have extended the decentralized ledger to a decentralized computing platform. In these protocols, besides financial transactions, programming code can also be stored in the blockchain, and the agents in the network can reach a consensus on the outcome of these programs. Programs that are stored on a blockchain and executed using a decentralized computing platform are

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.
*Corresponding author. Email: volker.strobel@ulb.be

called blockchain-based smart contracts (or smart contracts, in short form).

Initially, the combination of swarm robotics and blockchain technology was proposed at the conceptual level by describing various potential applications (such as data integrity, consensus agreement, and secure communication) (*30*) but without implementation or empirical testing. More recently, there have been several proofs of concept in which robot swarms and other multi-robot systems interacted via smart contracts (*19*, *31*–*36*). The main contributions of these simulation studies were to show that smart contracts can coordinate the robots' behavior and can successfully neutralize Byzantine robots, whereas existing consensus protocols (*37*–*39*) might break down in the presence of even a single Byzantine robot. However, only small simulated swarms were studied, or the blockchain protocol was executed on external infrastructure, and most studies used the widespread proof-of-work protocol (*40*), which requires high computational power and is therefore inappropriate for the relatively simple robots used in most robot swarms. Until now, it has been unclear to what extent blockchain and smart contracts could be used to control large swarms of real robots.

Here, we go beyond the existing research and present a comprehensive study with real robots (Movie 1). We performed the experiments using Pi-puck robots (*41*), which are e-puck robots augmented by a Raspberry Pi Zero W computer (*42*). To allow the robots in the swarm to exchange blockchain information in a decentralized way, we implemented a proof-of-authority blockchain (see the "Proof-of-authority consensus algorithm" section in Supplementary Methods) and a mobile ad-hoc network using the Pi-puck robots. In addition, we extended our existing simulation framework (*33*) to handle a large number of simulated robots (see the "Simulation framework: Installation and execution of the software" section in Supplementary Methods). We first validated our simulation by showing that the results in simulation and reality did not differ substantially from each other; we did so by conducting experiments that involved different arena sizes and swarm sizes (up to 24 real and simulated robots). We then used the validated simulator to study the scalability of blockchain-based robot swarms (up to 120 robots) and their behavior over longer experiment times (up to 600 min). To facilitate the future prototyping and development of a range of blockchain-based applications, we provided both the real-robot framework and the simulation framework as open-source software (see the "Simulation framework: Installation and execution of the software" and "Physical Pi-puck robots: Installation and execution of the software" sections in Supplementary Methods).

Movie 1. Overview of the research and of the experiments. The video briefly illustrates the working of our blockchain-based robot swarms and the obtained results.

In our experiments, each robot was a node in a peer-to-peer blockchain network that was maintained by the robots in the swarm (see Materials and Methods). The size of the robot swarm was fixed at the beginning of each experiment (see the "Proof-of-authority consensus algorithm" section in Supplementary Methods for a brief discussion of swarms that can change their size at runtime). The blockchain protocol that we chose to execute on our robots is Ethereum (*29*), one of the most widely used and mature protocols that support smart contracts. For robot swarms, one great advantage of Ethereum is that it allows the usage of the proof-of-authority consensus protocol instead of the more computationally expensive proof-of-work consensus protocol. Proof of authority requires a majority of preselected nodes (that is, a majority of robots in this work) to agree on the state of the blockchain database. We argue that Ethereum's proof of authority is an appropriate and energy-efficient choice for robot swarms in which the robots have limited computational power and battery life. Our experimental results show that limited robots could reliably execute the Ethereum software and the associated proof-of-authority consensus protocol.

Using both real and simulated robots, we studied the behavior of robot swarms in the presence of Byzantine robots in a collective-sensing scenario. Here, we focused on a specific type of Byzantine fault: the dissemination of wrong or deceitful messages to other members of the robot swarm. This type of fault is harder to detect and more severe than complete robot failures because it is, in general, not easy to understand whether a specific robot is or is not trustworthy (*11*) and because deceitful messages can have a strong influence on the overall behavior of the robot swarm (*33*).

Our robots communicated with each other using a smart contract that was executed on the blockchain. The smart contract enabled a token economy to be established among the robots in which each robot owned crypto tokens that it could spend to participate in security-critical swarm activities by sending transactions to the smart contract. In our scenario, robots sent transactions that contained information extracted from their sensor readings. This information was used by the smart contract to enable the swarm to estimate an environmental property. The smart contract repeatedly provided, at exponentially increasing time intervals, crypto tokens to every robot that was part of the experiment, regardless of how the robot performed. We call this crypto token allocation procedure universal basic income (UBI) (see Materials and Methods for an explanation of the smart contract and of the UBI mechanism). Thanks to the UBI, all robots—at least initially—could send transactions. However, the smart contract also ensured that non-Byzantine robots were rewarded for sending "good" transactions, whereas Byzantine robots did not get rewards for sending "bad" transactions, which resulted in the Byzantine robots running out of crypto tokens and, therefore, in their inability to send transactions and to influence swarm behavior. In our experiments, a "good" transaction was one that was not rejected by the outlier detection algorithm implemented in the smart contract (see Materials and Methods), whereas a "bad" transaction was a rejected one.

We analyzed the performance and resilience of our approach when facing diverse Byzantine faults. We also analyzed its scalability

and long-term behavior (up to 600 min for single runs) using different swarm and arena sizes in simulation experiments.

RESULTS

We evaluated our blockchain-controlled robot swarms both in simulation and with real robots (see the "Simulation framework: Installation and execution of the software" and "Physical Pi-puck robots: Installation and execution of the software" sections in Supplementary Methods for setup and operation instructions). The simulation was programmed to reflect as closely as possible the conditions of the real-robot experiments (see Materials and Methods) and was used to run experiments in situations that are challenging to study with real robots because of time, budget, battery, and space limitations.

In all experiments, the goal of the swarm was to reach a consensus on the percentage of white tiles (a value that can be set between 0 and 100% by the experimenter) in an environment in which the floor is covered with white and black tiles (Fig. 1). To obtain their individual estimates, robots equipped with ground sensors performed a random walk in the environment.

To reach a consensus on a value that was as close as possible to the correct one (25% white tiles in all our experiments), the robots' estimates were broadcast (communication range: 10 cm) by the robots in the form of transactions. The decentralized blockchain protocol ensured that the robots were able to agree on the addition of transactions to the blockchain. The smart contract then aggregated them, produced the collective estimate, and determined when the collective estimate had converged (see Materials and Methods).

The swarm estimation activity was complicated by the fact that a part of the swarm consisted of Byzantine robots that sent wrong estimates. Depending on the experiment, the Byzantine robots either constantly sent "0% white tiles" or drew their individual estimates from a random distribution (either a Bernoulli distribution with $P = 0.5$ or a uniform distribution).

To study the performance of our system, we ran two experiments (see also Table 1). The first experiment investigated whether the presented system tolerated an increasing number of Byzantine robots in the swarm. In this experiment, we tested our system under the conditions of different Byzantine faults and floor layouts. The second experiment addressed the scalability of the approach and its long-term behavior. Regarding scalability, we studied how both the swarm performance changed when the swarm size increased in an arena of constant size and, therefore, the robot density increased and when the size of the arena increased proportionally to the swarm size such that the robot density remained constant. We then performed longer-term experiments in which we let our system run for up to 600 min (a 40-fold increase with respect to the previous experiments) and studied the behavior of the swarm estimates generated by the robots, the growth of the blockchain size, and how the crypto tokens became distributed among robots. The performance of the presented system was evaluated by calculating the absolute error of the swarm, that is, the absolute difference between the actual percentage of white tiles and the swarm estimate at the end of each run. The swarm estimate was also compared with a baseline that was calculated offline by aggregating the individual sensor values without outlier detection and without using a blockchain (see Materials and Methods). Furthermore, depending on the experiment, we assessed performance according to the following metrics (see Materials and Methods for definitions): convergence time (the time it takes until the smart contract determines that the robots have agreed on an estimate), the size of the blockchain in megabytes, the number of crypto tokens owned by each robot, and the bandwidth utilization.

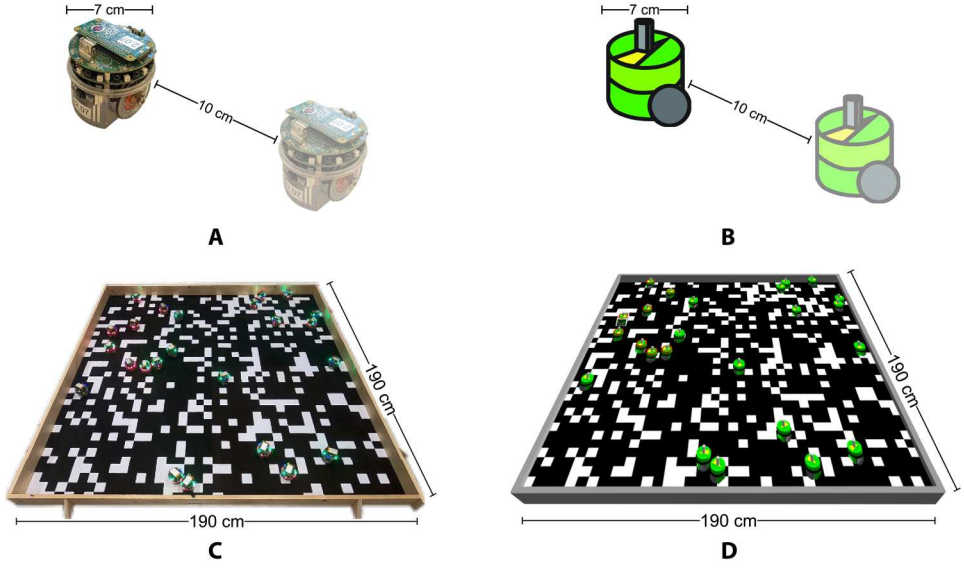

Fig. 1. Overview of the experimental setup. The goal of the robot swarms was to estimate the fraction of white floor tiles in a checkerboard environment (see also Movie 1 for an overview of the experiments). Reaching the goal was complicated by the presence of Byzantine robots that send incorrect sensor data. The robots used blockchain-based smart contracts to coordinate with each other. Each robot in the swarm was a node in the decentralized blockchain network and relied only on local communication (communication range: 10 cm). The experiments were conducted both in reality, using Pi-puck robots (*41*), and in simulation, using the ARGoS robot swarm simulator (*48*). (**A**) The Pi-puck robot and (**B**) its simulation model. (**C**) The real checkerboard environment and (**D**) the corresponding simulated environment.

Table 1. Overview of the experiments and their parameters. When an experimental setting is underscored, it means that the corresponding experiment was performed both with real robots and in simulation; when it is not underscored, it means that the experiment was performed exclusively in simulation.

Experiment	Swarm size	Number of Byzantine robots	Arena size (m^2)
1 – Robustness against an increasing number of Byzantines			
a – Byzantine fault: 0% white tiles	24	0, 3, 6, 9	3.6
b – Generality study	24	0, 3, 6, 9	3.6
2 – Scalability and long-term behavior			
a – Increasing swarm size in constant arena	8, 16, 24	0%, 25%	3.6
b – Increasing arena size with constant robot density	8, 16, 24; 48, 72, 96, 120	25%	1.2, 2.4, 3.6; 7.3, 10.9, 14.4, 18.1
c – Long runtime: swarm estimate behavior and blockchain growth	8, 16, 24; 48, 72, 96, 120	25%	1.2, 2.4, 3.6; 7.3, 10.9, 14.4, 18.1
d – Long runtime: token flow dynamics	24	25%	3.6

Experiment 1: Robustness against an increasing number of Byzantines

In this experiment, we studied the influence Byzantine robots have on the performance of a swarm of fixed-size $N = 24$ robots. The number of Byzantine robots in the swarm was increased from 0 to 9 in steps of 3; that is, they represented from 0 to 37.5% of the whole swarm in steps of 12.5%. We considered a few different experimental conditions.

First, in experiment 1a, we considered the situation in which Byzantine robots always sent 0% white tiles, independent of their actual sensor readings. Our hypothesis was that the lowest absolute error was obtained when there were no Byzantine robots in the swarm.

The results (see Fig. 2A) show that, when there were no Byzantine robots, the median absolute error was very low for all approaches (reality: 0.6%, simulation: 0.6%, baseline: 0.5%). The low error, together with the low variance of the error, shows that our setup was functional: The ground sensors worked as intended, the random walk routine was appropriate, and our blockchain approach performed as well as the baseline. This means that even when there are no Byzantine robots in the swarm, a blockchain approach can be appropriate because it does not degrade performance. This is important because, in general, we do not know a priori whether the swarm will contain Byzantine robots.

When the number of Byzantine robots increased, the median error of our approach increased only slightly. By contrast, the baseline showed an approximately linear increase in the error (Fig. 2A). The difference occurred because the estimates of the Byzantine robots were rejected by the smart contract, and, therefore, the Byzantine robots eventually ran out of crypto tokens and could no longer influence the collective estimate.

The presence of Byzantine robots also slowed down the convergence process (Fig. 2B). When the number of Byzantine robots increased, the number of valid estimates—those that were not rejected by the smart contract—in a given time period decreased because the smart contract filtered out the estimates sent by the Byzantine robots. Therefore, a longer period of time was necessary for the swarm estimate to converge.

Second, in experiment 1b, we investigated the generality of our approach by checking whether the results obtained in the previous experiment held when varying some of the experimental conditions (these tests were run in simulation only). We considered the following three conditions: First, for each repetition of the experiment, a new layout for the tiles on the floor was randomly generated (see Materials and Methods), and the Byzantine robots always sent 0% white tiles. Second, every time a Byzantine robot sent an estimate, it selected the estimate to be either 0% white tiles or "100% white tiles" with the same probability (the estimate was selected using a Bernoulli distribution with $P = 0.5$). Third, every time a Byzantine robot sent an estimate, it selected the estimate to be "x% white tiles," where x was a random value drawn from the uniform distribution between 0 and 100. As can be seen in Fig. 3, the results are very similar to those of the previous experiment 1a, which suggests that our approach could handle a range of different tile distributions and Byzantine faults.

Experiment 2: Scalability and long-term behavior

In the second experiment, we first studied how our approach scaled when the swarm size was increased while the arena size remained constant (experiment 2a) and when the size of the arena grew proportionally to the swarm size (experiment 2b). We then considered much longer runtimes to study how the swarm's collective estimate behaves after convergence was reached—that is, whether the collective estimate got more accurate, the estimate stagnated, or even whether convergence was lost (experiment 2c)—and to study the dynamics of the token economy and the long-term consequences of sending wrong data (experiment 2d). In all experiments, the Byzantine robots sent an estimate of 0% white tiles.

In experiment 2a, we investigated to what extent the size of the swarm had an influence on the absolute error and the convergence time. In particular, we were interested in understanding whether a larger swarm yielded more accurate results and whether this might be at the expense of a longer convergence time. In addition, we were interested in studying how the approach performed when connectivity was sparse. To this end, we used three swarm sizes: 8, 16, and 24 robots in an arena with a constant size of 3.61 m^2. We performed the experiments both with and without Byzantine robots (25% when present).

Results show (see Fig. 4, A and C) that, if there were no Byzantine robots in the swarm, the median of the absolute error was low: under 1.0% for all swarm sizes and approaches. When using the blockchain-based approach, the median of the absolute error remained low (less than 2.1%) even in the presence of Byzantine robots, whereas it grew to more than 6.5% in the baseline.

The results additionally show that the approach still converged when the swarm was only sparsely connected. In particular, in the experiment with eight robots in the 3.61-m^2 area, connectivity was very sparse. Therefore, most of the time, the robots did not have any

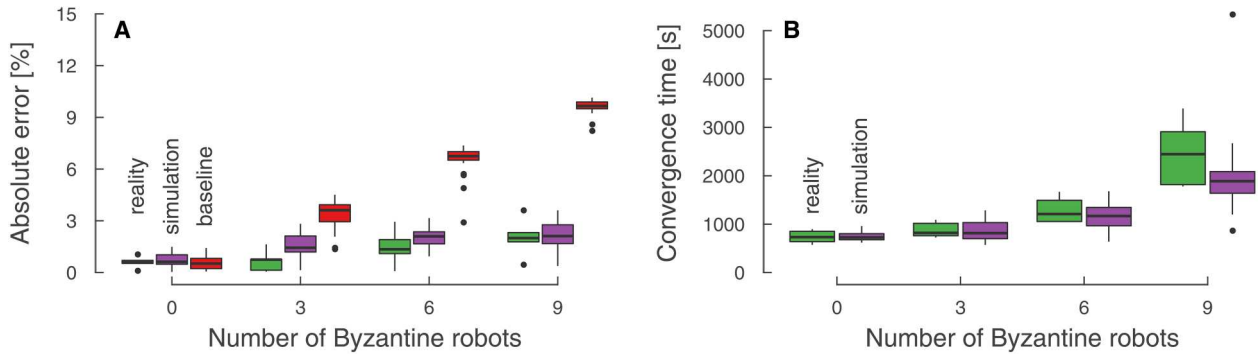

Fig. 2. Experiment 1a – Increasing Byzantines. Twenty-four robots in total, of which 0, 3, 6, or 9 were Byzantine robots that always sent a 0% estimate. In all the plots presented here, the boxes in the boxplots represent the first quartile, median, and third quartile; the whiskers extend up to the 1.5–interquartile range values, and the dots represent the outliers. The green boxes are the results for real robots, the purple boxes are the results for the simulations, and the red boxes are the results for the baseline (see Materials and Methods for the calculation of the baseline). For each setting of each experiment, we conducted 5 repetitions with real robots and 20 repetitions in simulation. (**A**) The approach worked well both in reality and in simulation, and the median of the absolute error stayed below 2.2% even with 9 Byzantine of 24 robots (that is, 37.5% of the robots in the swarm were Byzantines). (**B**) The higher the number of Byzantines, the stronger their influence on the convergence time. Convergence time is not reported for the baseline because it was computed offline.

connections to other robots. The sparse connectivity led to many network partitions that were eventually resolved.

The convergence time tended to decrease as the swarm size increased (see Fig. 4, B and D). The main reason for the shorter convergence time is that the average connectivity— that is, the average number of other robots to which a robot was connected in the time unit—was higher in larger swarms: It was 0.24 robots per second in swarms with 8 robots, 0.59 robots per second with 16 robots, and 0.91 robots per second with 24 robots. In addition, a larger swarm had more sensor readings, resulting in a lower variance of the sample mean and thus a shorter convergence time.

In experiment 2a, we showed that a larger swarm yielded both a lower absolute error and a shorter convergence time when the arena size was kept constant. In experiment 2b, we studied whether the same held true when the arena size increased proportionally with the swarm size and therefore the robot density, measured as the number of robots per m^2, remained constant (see Materials and Methods for exact arena sizes). Results show (see Fig. 5A) that, in this case, in all experiments, the median of the absolute error stayed below 3%. The larger the number of robots, the more accurate the estimate: Both variance and absolute error slightly decreased with an increasing number of robots. Once again, this is a good indication of the scalability of the approach.

The convergence time (see Fig. 5B) increased with the swarm size, but this increase was small: From an 8-robot swarm to a 120-robot swarm (a factor of 15), the median of the convergence time was less than doubled. The increase in convergence time happened because, in a very large arena, despite a constant robot density, longer disconnection times can occur (for example, when almost all robots are in one corner and a single robot is in another corner), and it can therefore take longer for all robots to receive the convergence signal. Overall, the low absolute error for all swarm sizes and the small increase in the convergence time further demonstrate the scalability of the approach and suggest that a blockchain-based approach might also work in real-world deployments with much larger arenas and higher numbers of robots.

Experiments 2c and 2d were designed to test the long-term behavior of our system. Therefore, we did not stop the experiments after the swarm had converged to a common estimate; instead, we let them continue to run for a total duration of 600 min. We ran these experiments in simulation only because of the limited battery life of currently available robots.

Figure 6 shows the results of experiment 2c, in which we considered swarms of 8 to 120 robots. Twenty-five percent of the robots were Byzantine, and the runtime was 600 min. We analyzed individual runs in this experiment: The reported data represent one repetition for each swarm size. At the beginning of the experiments, the robots exclusively relied on the UBI to pay for sending their estimates. Therefore, there was a delay before the first swarm estimate was generated (Fig. 6A) because the robots could send their individual estimates only after they had received enough UBI payments. Note that the time it took to generate the first swarm estimate was largely independent of the swarm size N, because, as we scaled the arena size proportionally to N (ensuring a constant robot density), the block time (see the "Proof-of-authority consensus algorithm" section in Supplementary Methods) and the initial delay in the transfer of the UBI payments did not change substantially. After the first swarm estimate was generated, the estimate quickly converged to the actual fraction of white tiles (Fig. 6A). The estimate converged from lower values, because the Byzantine robots, which always sent a 0% estimate, initially shifted the estimate down before gradually losing influence.

Results also show (see Fig. 6B) that the blockchain size grew linearly with time, which made it possible to make good predictions about the evolution of the blockchain size over longer runtimes. The rate of growth of the blockchain also increased with the swarm size because a larger swarm created more transactions.

Last, in experiment 2d, we analyzed the inbound token flow over time—that is, the cumulative sum of crypto tokens received by a robot as a function of time, which consisted of UBI payments and reward payments. Figure 7A demonstrates the dynamics of the blockchain token economy and the sound operation of the smart contract logic: The longer the experiment lasts, the greater the difference between the fraction of inbound token flow attributed to Byzantine and non-Byzantine robots. This is because, over time, the UBI payments, which were received by all robots, became less

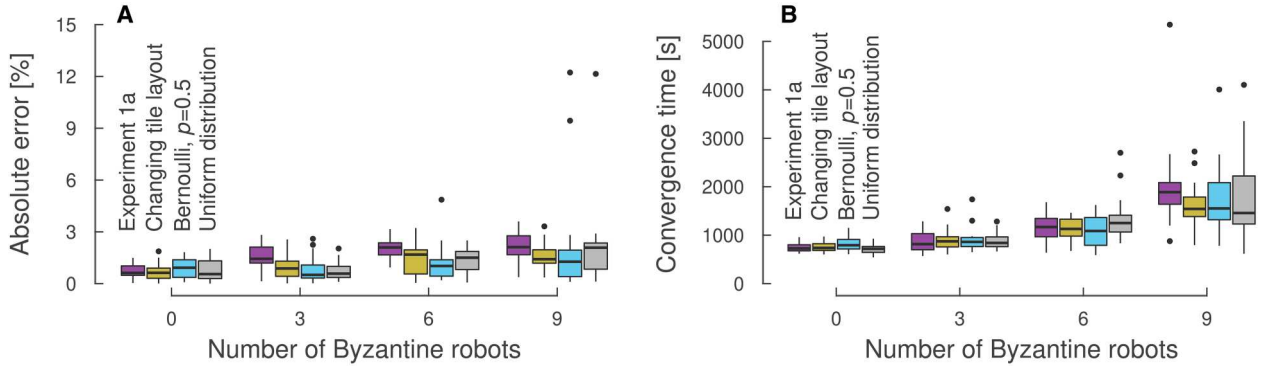

Fig. 3. Experiment 1b – Generality study. Twenty-four simulated robots in total, of which 0, 3, 6, or 9 were Byzantine robots. For each setting of each experiment, we conducted 20 repetitions in simulation. To better compare the results, we show again those from the previous experiment 1a (purple boxes), in which robots always sent a 0% estimate. The ocher boxes show the results for different floor layouts, in which robots always sent, as in experiment 1a, a 0% estimate. The light blue and gray boxes, respectively, show the results for the Byzantine faults where the estimates were drawn from the Bernoulli distribution with $P = 0.5$ and for the Byzantine faults where the estimates were drawn from the uniform distribution; in both cases, the floor layout was the same as in experiment 1a. (**A**) Absolute error. (**B**) Convergence time.

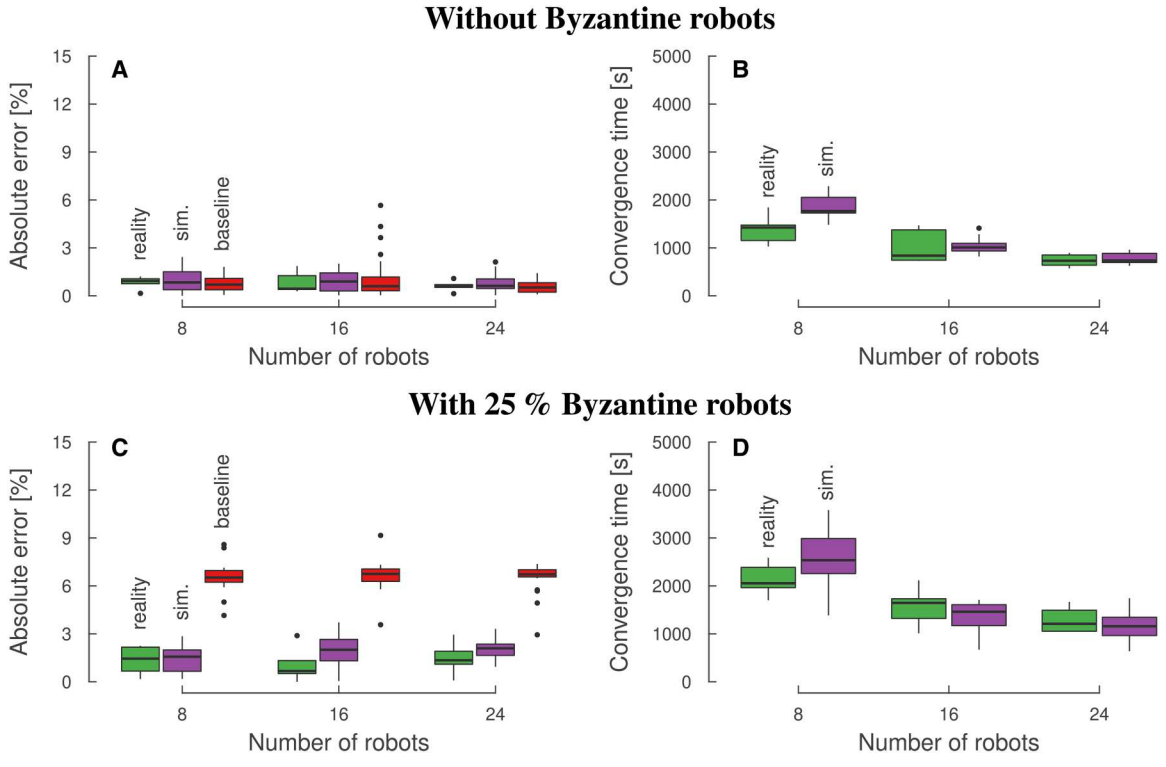

Fig. 4. Experiment 2a – Increasing swarm size in an arena of constant size. Eight, 16, and 24 robots in total without and with 25% Byzantine robots that always sent a 0% estimate. For each setting of each experiment, we conducted 5 repetitions with real robots and 20 repetitions in simulation. (**A** and **C**) Regardless of swarm size, the absolute error of our blockchain-based approach was small, even in the presence of Byzantine robots. (**B** and **D**) The convergence time decreased noticeably with a larger swarm size. This was due in part to the better connectivity and in part to the higher accuracy when more robots made a measurement. The decrease in the convergence time as a result of a larger swarm is an indication of the scalability of our approach. Convergence time is not reported for the baseline because it was computed offline.

and less frequent, and robots obtained crypto tokens mainly from the reward payments (see Materials and Methods), which were mostly earned by non-Byzantine robots (this was because non-Byzantine robots' estimates were often accepted and rewarded by the smart contract, whereas those of Byzantine robots tended to be discarded). Note that, because of this mechanism, all UBI payments, which were distributed to all robots at exponentially increasing time intervals, were eventually redistributed to the non-Byzantine robots. Therefore, the non-Byzantine robots' inbound token flow increased logarithmically over time and so did the rate at which they submitted their estimates (see fig. S1 and the "Rate of estimates" section in Supplementary Results).

Figure 7B shows how the inbound token flow changed over time for each robot in the swarm. In particular, it can be observed that the

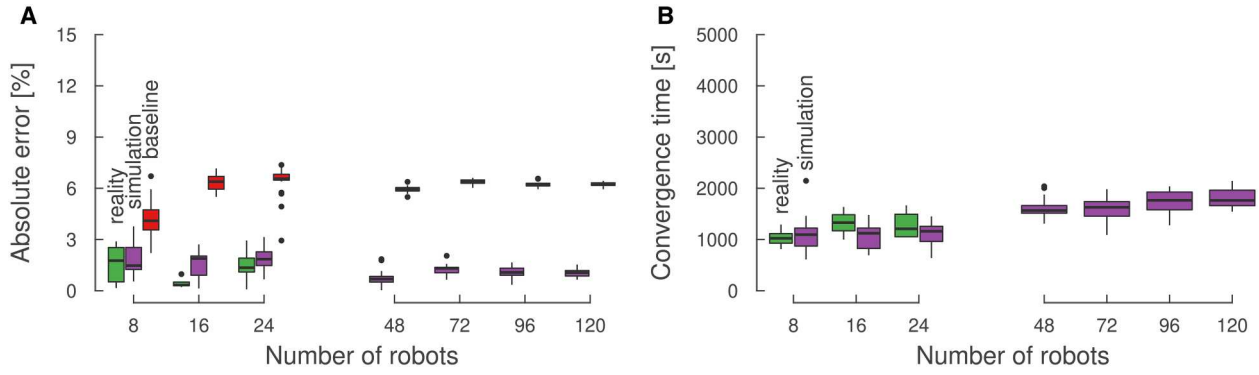

Fig. 5. Experiment 2b – Increasing arena size with constant robot density. Up to 120 simulated robots, of which 25% were Byzantine robots that always sent a 0% estimate. The *x* axes are interrupted to emphasize that, for swarm sizes larger than 24, the number of robots in the swarm is increased in steps of 24 instead of 8. For the experiments with real robots, we ran 5 repetitions for each setting; for the simulations, we ran 20 repetitions. Experiments with 48 or more robots were run in simulation only. (**A**) Independent of the swarm size, the absolute error was small. With an increasing swarm size, however, the variability decreased, so confidence in the swarm estimate was increased. This result supports the scalability of the approach. (**B**) The convergence time increased only slightly with an increasing swarm size: from 8 to 120 robots, the convergence time less than doubled.

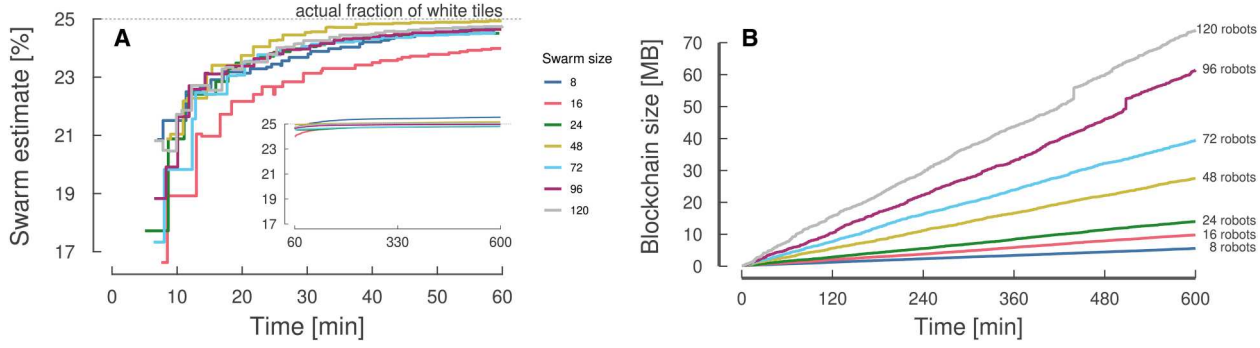

Fig. 6. Experiment 2c – Long runtime: Swarm estimate behavior and blockchain growth. Up to 120 simulated robots, of which 25% were Byzantine robots that always sent a 0% estimate. (**A**) The swarm converged quickly, and the estimate was accurate, as already shown in the previous experiments. The inset shows the swarm estimate from 60 to 600 min. After 60 min, the swarm estimate hardly changed. (**B**) With an increasing swarm size, the blockchain size also increased faster. This is because the robots created more transactions in a larger swarm. The sudden increases in blockchain size for 96 and 120 robots (at the approximate times 500 min and 440 min, respectively) are due to the compaction algorithm of Ethereum's database implementation (LevelDB) and are offset by a later decrease in blockchain size when the blockchain reached about 120 MB. The presented data were obtained with one run-in simulation for each swarm size.

inbound token flow of the Byzantine robots stayed low and only increased when they received UBI payments. It can also be observed that token flow could vary greatly among non-Byzantine robots (two of the non-Byzantine robots had a substantially lower token flow, marked with an asterisk in Fig. 7B). The lower token flow was a result of random fluctuations: At the beginning of the experiment, the robots' estimates were still inaccurate because they had only taken a few ground sensor readings. It could therefore happen that they sent inaccurate estimates early in the experiment (as done by the two robots marked with the asterisk in Fig. 7B). In this case, the smart contract discarded their estimates and did not reward them, although they were not Byzantine robots. These initial inaccuracies could then lead to an overall lower token flow for the concerned robots.

DISCUSSION

The benefits of using blockchain technology in robot swarms

Thanks to blockchain technology's solution to the double-spending problem (the problem of the same token being spent multiple times simultaneously) that allows decentralized systems to maintain scarce digital quantities in the form of crypto tokens, the robots in our swarm were able to perform economic transactions. In a series of experiments, we showed that our blockchain-based token economy could be used to neutralize Byzantine robots. When Byzantine robots were part of the swarm, our blockchain-based approach always performed better than the baseline, and when there were no Byzantine robots in the swarm, the blockchain-based approach still performed as well as the baseline. Because we cannot know a priori whether Byzantine robots will or will not be present in a swarm, this assures that the obtained performance will be very good in both the presence and absence of Byzantine robots.

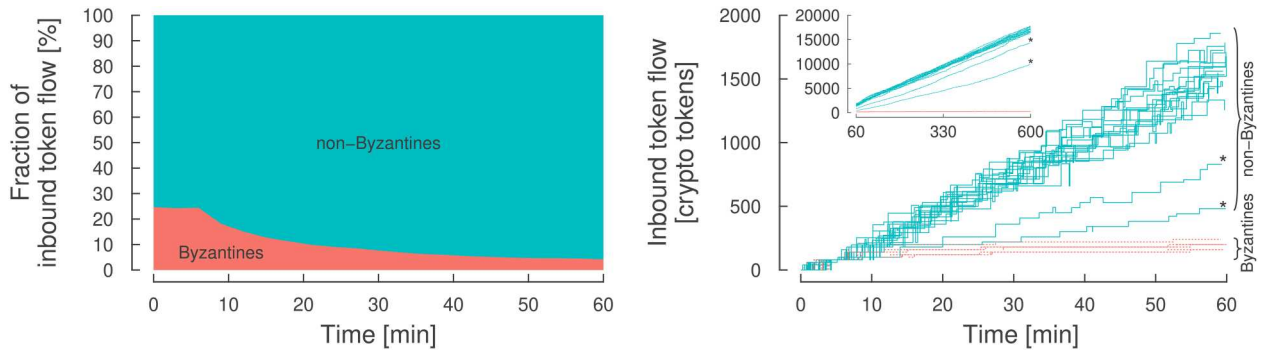

Fig. 7. Experiment 2d – Long runtime: Token flow dynamics. Twenty-four robots, of which 25% were Byzantine robots that always sent a 0% estimate. (**A**) The plot shows the fractions of the inbound token flow received over time by the non-Byzantine robots (cyan area) and Byzantine robots (red area). At the beginning of the experiment, all robots received the UBI payments. Because the Byzantine robots constituted 25% of the swarm size, they received 25% of the inbound token flow. However, later on, reward payments were the major source of token flow; because of the outlier detection mechanism, Byzantine robots did not receive reward payments, and their fraction of the token flow decreased. After 60 min, the token flow attributed to Byzantine robots continued to converge toward 0%. The plot was generated by running 20 repetitions in simulation; standard deviation is not visible because it was always smaller than 1% of the fraction of inbound token flow. (**B**) The plot shows the inbound token flow per robot over time (each curve represents one robot) for one run with 24 robots. The inset shows the token flow from 60 min to 600 min. Whereas the non-Byzantines (cyan curves) continuously received payments, the Byzantines (red curves) received payments only when they received the UBI, which happened rarely, and therefore their token flow remained very low. Note that there were also two non-Byzantine robots that had a substantially lower token flow (marked with an asterisk) than other non-Byzantine robots because they sent inaccurate estimates at the beginning of the experiment.

The smart contract calculated the swarm estimate and determined convergence online—that is, during the experiment and within the swarm, letting the swarm determine in a fully decentralized way whether the shared estimate had converged. This could be done even in the presence of Byzantine robots. The swarm could therefore autonomously decide when to switch to the next task without the need for any external interactions.

In addition, there are two desirable properties that we inherit from using blockchain technology. First, during the autonomous operation of the robot swarm, a conflict-free logbook of the messages exchanged is automatically stored in the blockchain. This logbook could be used for establishing accountability and for performing postliminary investigation and data forensics (*25*) because the blockchain offers non-repudiation (a robot in the swarm cannot deny that it sent a message). When using our blockchain-based approach, to obtain data about the entire course of the experiment it is sufficient to extract data from only one robot because all robots in the swarm—Byzantines included—share the same logbook (this is particularly useful when a majority of the robots are lost or destroyed).

Second, when using blockchain technology, implementing a collective behavior amounts to programming and deploying a smart contract rather than reprogramming every single robot in the swarm. Because the control information is encapsulated in one place, it becomes easier for a user both to modify the swarm behavior and to exchange confidential information with the swarm using approaches such as the one presented in (*21*).

Computational requirements

The addition of a blockchain protocol layer to the control of a robot swarm increased the complexity of the system and the computational load on the robots. An important question is therefore whether a blockchain is necessary to secure a robot swarm or whether a subset of blockchain technology's components would be sufficient. According to the current state of the art, the answer is that the components of blockchain technology are the minimum possible set that is capable of creating a tamper-proof system in a network of mutually untrusting agents (*43*). These components are public-key cryptography and the use of digital signatures; a consensus protocol; a decentralized database; and, if the system should be used as a computing platform, smart contracts. Omitting any of these components would lead to a system that lacks the required security for real-world deployment in the following respective ways: lack of message authentication and non-repudiation, inability to solve conflicting states of the decentralized ledger, lack of a tamper-proof register of events accessible during and after an experiment, and inability to guarantee agreement on the outputs of programs.

Given that we need each of these components, the important remaining question is whether implementing a blockchain protocol layer is feasible in terms of computational load. In other words, we were interested in understanding whether, when executing blockchain software on real robot swarms, the computation, storage, and communication requirements of the blockchain protocol are manageable by a system composed of robots that have limited computational power and low storage capacity and rely on local and dynamic communication. Our experiments have demonstrated that this is possible.

The best-known and most common consensus protocol used in blockchain technology is proof of work, first proposed for Bitcoin. An important drawback of this protocol is that it might require a huge amount of energy and can therefore be problematic for use in a robot swarm (*40*). Our choice was therefore to use the proof-of-authority consensus protocol, which uses much less energy and that we found to be well-suited for the simple robots used in our experiments. With proof of authority, the CPU and RAM (random-access memory) capacities of the robots were far from fully used (see the "CPU and RAM utilization" section in Supplementary Results).

Concerning storage of the blockchain in the robots, we have shown (Fig. 6B) that the blockchain size grew linearly both with the runtime and with the swarm size. Using a swarm of 120 robots, the size of the blockchain data folder in a 600-min run

was 73.6 MB. Because the Pi-puck robots that we used were equipped with a 16-GB secure digital (SD) card, the same experiment could be executed for several weeks before the robots would run out of storage. For this reason, we do not consider storage requirements to be a consequential issue for our system.

To get an estimate of the blockchain's communication overhead, we measured the bandwidth utilization (see Materials and Methods for the definition of bandwidth utilization) for swarms of 120 robots. The mean bandwidth utilization was 2.11 kB/s during the entire experiment. If one considers only the time when the robots communicated with each other, then the mean bandwidth utilization was 4.97 kB/s. During the 600-min experiment, the total size of sent packets per robot was 72.5 MB (the size of sent packages was smaller than the final blockchain size, which was 73.6 MB, because the blockchain data folder also included files needed for the database workings). The results indicate that communication costs were rather low: The total size of sent packets was approximately the same size as the final blockchain size, and the bandwidth utilization did not increase over time. As a consequence, the bandwidth of the Pi-puck's onboard Wi-Fi module was far from being fully used (less than 1% in our experiments).

Challenges

More complex application scenarios and smart contracts

A future challenge will be to use the technology presented here to address situations in which robot swarms need to perform more complex tasks or are faced by smarter Byzantine robots. In these cases, there will be a need for more sophisticated smart contracts. Here, our goal was to build the foundations for the use of smart contracts and blockchain technology in robot swarms: first, by bringing blockchain technology to real robots using the proof-of-authority consensus protocol and suitable communication architecture and, second, by studying Byzantine fault tolerance, economic rewards, scalability, and sparse connections. To do so, we chose to use a simple application scenario and associated smart contract.

For example, the task given to the swarm—that is, the estimation of the tile fraction—was relatively simple and could be performed by a single robot. However, even in this simple case, our blockchain-based robot swarm was desirable because it was robust to the presence of Byzantine robots. By contrast, in the single robot case, if the robot were Byzantine, we would not have any way to neutralize it.

Also, the smart contract that we have designed was rather simple and could be circumvented by smarter Byzantine robots (see the "More sophisticated Byzantine behavior" section in Supplementary Discussion for examples). Although it is, in principle, possible to write smart contracts that take into account any foreseeable behavior of Byzantine robots (this can be done because smart contracts are Turing-complete), we have chosen to implement a relatively unsophisticated smart contract that was, however, good enough for our proof-of-concept study (see Materials and Methods for details about the smart contract that we developed for this research). In practice, the smart contract design will need to be adapted to different circumstances, including tasks, environments, and possible attacks. For example, if the considered environment is dynamic—as opposed to the static environment studied here—then a more sophisticated smart contract could be used that removes older estimates or detects changes in the robots' estimates.

Dynamic swarms

For real-world deployment, it might be desirable to have robot swarms where robots can dynamically join or leave or might belong to different owners. However, in these swarms, it is easier to introduce Byzantine robots because there is no control over who introduces a robot. Also, it becomes, in principle, possible to perform a Sybil attack (an attack in which a robot controls several identities) because the total number of robots is not known, as was the case in the "closed" swarm considered here. One possible approach to this challenge would be to use a permissioned blockchain with proof of authority. An issue with such a permissioned network is that an entity in charge of granting permissions needs to be defined. This entity could be the swarm itself, for example, using a voting mechanism (*44*) or a vouching system (see the "Proof-of-authority consensus algorithm" section in Supplementary Methods), or an external entity, such as the owners of the robots belonging to the swarm.

Network partitions

In a real-world deployment, robot swarms could face more severe network partitions (clusters of robots in disconnected subswarms) than in our experiments. These network partitions could delay information propagation and could be exploited by an attacker to gain control of a relative majority of robots in the swarm. There are, however, ways in which this problem can be addressed.

One possible approach to preventing network partitions would be to let the robots use a self-organized aggregation strategy (as opposed to the random walk behavior used in the present research). One could even develop algorithms that combine blockchain-based consensus protocols and (physical) aggregation of robots: For example, a "proof-of-dissemination" algorithm could reward robots that had the most physical encounters with other robots and, therefore, helped to disseminate information in the swarm. Another approach could be to use messenger robots that have a higher communication range or speed than the rest of the swarm: For example, one could use drones as messenger robots, whereas the rest of the swarm could be composed of ground robots.

Last, one could use partition-tolerant distributed ledger technology frameworks as they become available. It is certainly possible that, because blockchain and distributed ledger technology research is under very active development, in the future, we will see ready-to-use protocols that are more suited for our purposes than Ethereum with the proof-of-authority consensus protocol. For example, partition-tolerant frameworks based on directed acyclic graphs (DAGs) are currently under development and do not require all robots to have an identical copy of a blockchain, and hence, smart contracts do not need to be executed by all robots (*36*, *45*). As in stationary blockchain networks, there will be no universal solution, but the respective blockchain framework and its parameters must be adapted to the respective application (*46*).

51% attacks

Although situations in which a majority of the robots are Byzantine are likely to be rare in real-world applications, it might be desirable to secure robot swarms even in these cases. It should be noted, however, that the security and performance of our approach are limited to what blockchain technology can provide. For example, proof of authority, as any other known consensus protocol, is susceptible to attacks when a majority of participants are Byzantine. In other words, if an attacker can control more than 50% of the robots in the swarm, then the information that is stored in the blockchain is

no longer trustworthy. This is a limitation that is inherent to the chosen technology and that might be removed or made less stringent through future research developments in the field of distributed consensus protocols. However, as of today, there are no other distributed consensus protocols that achieve better results.

Time delays

The use of blockchain technology can introduce time delays. After a blockchain transaction is created, it must be inserted into a block, which takes a certain amount of time depending on the connectivity and the selected blockchain parameters. In our particular case, this was regulated by the proof-of-authority consensus protocol. We have chosen the default value (15 s) for the minimum block time. Setting the block time is a trade-off: A lower block time prevents delays if the blockchain is to be used for real-time coordination (*35*). However, it also increases the probability of blockchain forks, which can lead to (temporary) conflicting information and an increase in communication overhead. Setting the block time to higher values prevents forks due to network partitions at the expense of slower consensus agreement. In our research, the presence of delays did not disrupt the swarm behavior because the smart contract did not handle the robots' low-level control. However, in applications where this is not the case (such as in a task allocation scenario where the smart contract tells the robots which tasks to perform), delays need to be considered when deciding which data should be processed by the smart contract. For example, one might choose to let the smart contract process safety-critical data, whereas the processing of time-critical data might be done locally by the robots. As mentioned above in the network partitions challenge, another approach would be to use a different distributed ledger technology framework with higher transaction throughput or better management of network partitions, such as a DAG-based architecture (*47*).

MATERIALS AND METHODS

Design of the smart contract variables and functions

There were four functions in the smart contract with which the robots could interact:

1) registerRobot() This function was executed by each robot at the start of the experiment; it was required before a robot could use the other functions of the smart contract. When executed, the function increased the swarm size N in the smart contract and registered the robot's public address on the blockchain.

2) askForUBI() This function allowed robots to obtain the UBI by sending a transaction to the smart contract.

3) sendEstimate(<localEstimate>) This function let the robots store their local estimates on the blockchain. To store an estimate, robots had to send 40 crypto tokens through a transaction.

4) hasConverged() This function determined whether the absolute difference between the previous and the current value of the swarm estimate was smaller than 0.2%, in which case it returned "true."

Crypto token economy

In our experiments, the rules for the distribution of crypto tokens formed the basis of an economic mechanism that we designed to minimize the influence of Byzantine robots on the collective estimate. The mechanism limited the number of messages (transactions) a robot could send by attributing an economic cost (in crypto tokens) to them. This worked by requiring all the robots to pay a certain number of crypto tokens to send transactions (40 crypto tokens in our experiments), but only non-Byzantine robots were rewarded for this activity and could therefore continue to participate in the token economy, whereas Byzantine robots soon ran out of crypto tokens and could no longer send transactions.

The robots could obtain crypto tokens in two ways: by being rewarded when sending accepted estimates and by receiving the UBI. These two ways are described in more detail in what follows.

Rewards and detection of Byzantine robots

Every time a robot sent an estimate [using the sendEstimate(<localEstimate>) function], this estimate was stored in a temporary list in the smart contract. As soon as the temporary list contained N estimates for the first time (called the first proposal round), the smart contract computed their arithmetic mean, which became the swarm estimate. The temporary list of estimates was then deleted. From then on, each time this list contained N estimates, the smart contract executed a new proposal round and discarded all the estimates that differed by more than $\delta = 0.2$ from the current swarm estimate; it then used the remaining ones to generate the new swarm estimate (obtained as before by computing the arithmetic mean). The temporary list was deleted again afterward. All robots that sent an accepted estimate received a reward payment that consisted of a refund of the 40 crypto tokens required for the sendEstimate transaction plus a bonus consisting of an equally distributed share of the non-repaid crypto tokens of the discarded estimates.

Here is an example: All $N = 16$ robots in a swarm have sent an estimate and, thus, there are $16 \times 40 = 640$ staked crypto tokens. Of the 16 estimates, say 13 estimates are accepted, and 3 estimates are discarded because their values differ by more than 0.2 from the current swarm estimate. The 13 accepted estimates are used to update the current mean in the smart contract. The 3 robots that sent the nonaccepted estimates lose their 40–crypto token stake. The 13 robots that sent accepted estimates are refunded their 40 crypto tokens, plus a bonus of about 9.2 crypto tokens ($3 \times 40 = 120$ tokens of the nonaccepted estimates distributed among 13 robots).

Universal basic income

The UBI is an economic mechanism that we established within the smart contract to grant crypto tokens to the robots. The UBI functioned as follows: The smart contract was provided with a quasi-infinite number of crypto tokens in the genesis block (these crypto tokens were securely locked and inaccessible). At blockchain block numbers that were a power of 2 (that is, blocks 1, 2, 4, 8,...), the smart contract transferred 20 crypto tokens to each robot in the swarm when the robots executed askForUBI(). This exponential scheme ensured that, at the beginning of an experiment, each robot received sufficient crypto tokens to be able to send estimates to the smart contract; however, over time, sending useful information (more precisely, accepted estimates) became the main means to receive crypto tokens and to be able to continue participating in the experiment.

The robots

For the experiments with real robots, we used the Pi-puck robot platform (*41*), with a maximum swarm size of $N = 24$ robots. The Pi-puck is an e-puck robot (*42*) extended by a Raspberry Pi Zero W single-board computer. The Raspberry Pi Zero W is a low-cost extension that is able to run a Linux-based distribution. It has a Wi-Fi

module and a 1-GHz processor with 512 MB of RAM. For data storage, we used a 16-GB SD card. The Raspberry Pi extension improved the robots' communication capabilities and computational power and, therefore, allowed for the implementation of more complex algorithms compared with previous e-puck robot versions. The low cost and size of the Raspberry Pi Zero W combined with its ability to execute the Ethereum software support our claim that blockchain technology is suitable for swarm robotics applications.

This paper adheres to the traditional assumptions of swarm robotics: resource-constrained hardware, local communication, no access to external infrastructure (such as the Internet and cloud computing), and changing network topologies. As a result, the presented approach can potentially be used in the typically challenging environments where robot swarms are envisioned to be deployed, such as underwater or in space. Parts of this research, in particular economic rewards or Byzantine robot detection, can also be transferred to other multirobot systems, including systems with access to the internet or a cloud server.

The communication architecture

For our robot swarm, we designed a communication architecture based on local-only communication. This choice has the desirable properties of making attacks exploiting long-distance communications (more than 10 cm away) impossible and of being coherent with the self-organizing nature of robot swarms.

For the real-robot experiments, the communication architecture of the swarms consisted of two layers. In the first layer, the robots maintained a WiFi mobile ad hoc mesh network (MANET) that allowed them to establish connections using the Transmission Control Protocol (TCP). This layer provided the bandwidth required for the synchronization of blockchain information. Because a MANET is a peer-to-peer network, it is decentralized and consistent with the requirements of swarm robotics research. However, because of its range of several meters, the MANET would facilitate remote attackers to interfere with the swarm. We have thus implemented a second layer that limited the range of the MANET by having robots reject TCP connections that were not in close vicinity, specifically not at a distance of 10 cm or less from each other (the robots detect the distance using their range-and-bearing module).

The robot controller

The controller of each robot was composed of five high-level routines, which were implemented using Python and the Ethereum blockchain software.

Random-walk with obstacle avoidance

At each moment in time, a robot performed one of the following three basic behaviors: straight movement, rotation on site, or obstacle avoidance. If a robot's infrared sensors detected a close-by obstacle (<5 cm), then obstacle avoidance was selected to prevent collisions (the eight infrared sensors on the robot's frame were used to decide whether the robot should turn left or right to avoid the obstacle). Otherwise, the robot alternated between straight movement and rotation on site; the duration of each phase was sampled from an exponential distribution (*39*).

Estimation

Once per second, a robot sampled the floor via its ground sensor. This sample was then mapped to either black or white and was used to calculate a local estimate of the percentage of white tiles in the environment by dividing the number of white ground sensor readings by the total number of readings.

Peering

Robots used their range-and-bearing actuators/sensors to simultaneously transmit their IDs and to listen for other IDs within a range of about 10 cm. When an ID was received, the robot executed a TCP request to obtain the enode (a unique identifier of an Ethereum blockchain node, used for the peering calls) from the other robot. Once ID exchanges no longer occurred, the peer was removed, and all local information regarding that peer was deleted.

Local estimate dissemination

As soon as a robot had a sufficient number of crypto tokens (40 tokens in our experiments), it sent its local estimate to the smart contract, which used them to generate a swarm estimate. Thus, the frequency of sent measurements was determined by the crypto token economy.

Blockchain synchronization

An instance of the Ethereum software (Geth, version 1.8.0) was executed on each robot during the entire course of the experiment. To create new blocks on the blockchain, each robot acted as a sealer for the proof-of-authority consensus protocol and broadcast the sealed blocks to its peers. When a block was received, the robot validated the contents of the block and updated its blockchain according to the specified consensus rules (see the "Proof-of-authority consensus algorithm" section in Supplementary Methods).

The simulator

We conducted the simulations in the robot swarm simulator ARGoS (*48*), together with the ARGoS e-puck robot plugin (*49*), and the ARGoS-Python interface (*50*). To combine ARGoS and the smart contract framework Ethereum, we used the ARGoS-Blockchain interface (*33*) and extended it to make it compatible with Python (see the "Simulation framework: Installation and execution of the software" section in Supplementary Methods). This interface enabled robots to act as blockchain nodes and provided them with access to blockchain functions. For the programming of the simulation framework, we have paid attention to efficiency and parallelization; thus, it was possible to model swarms consisting of more than 100 robots.

In the simulations, we also implemented a two-layer communication architecture to replicate the communication capabilities of the real robots: The robots determined via their range-and-bearing module whether they were 10 cm or closer to another robot. Whenever this was the case, TCP connections were established via the Docker network (see the "Simulation framework: Installation and execution of the software" section in Supplementary Methods).

The simulations were executed on Amazon Elastic Compute Cloud (EC2). To simulate the limited hardware of the real Pi-puck robots, 1.0 GHz of CPU and 512 MB of RAM were assigned to each Docker container.

There is a close match between the results obtained using real robots and the results from the simulations (see Results). This outcome highlights the quality of our developed simulation environment—because the reality gap is small—and shows that it can be used for the prototyping, the development, and the study of different scenarios. On the basis of this validation, we complemented the experiments, which were executed both with real robots and in simulations, by extended studies in simulation only.

The experiment design

Initialization and termination

The robots were randomly distributed in the arena at the start of each experimental run. In the real-robot experiments, this random arrangement was achieved by performing a random walk with obstacle avoidance before starting an experimental run. In the simulations, the initial positions of the robots were drawn from the uniform distribution.

In addition, at the beginning of each experimental run, a new genesis block (the first block of a blockchain) was created and distributed to the robots. This genesis block contained the smart contract and the list of robots that were allowed to create new blocks in the proof-of-authority consensus protocol (in our experiments, all robots of the swarm). An experimental run was stopped after all robots had received "true" when querying the function hasConverged().

Independent variables

Depending on the experiment, we varied one or more of the following independent variables: the swarm size, the number of Byzantine robots, the type of Byzantine fault, the dimensions of the environment, the layout of the floor, and the convergence criterion. In the following, these variables are described in more detail.

Swarm size

Changing the swarm size, that is, the total number of robots in the swarm, allowed for analyzing our approach in terms of two key features of robot swarms: scalability (the ability of the system to maintain or improve performance as the swarm size increases) and partition tolerance (the ability of the system to reach consensus when there is reduced connectivity, in this case, induced by a more sparse distribution of the robots in the arena).

Number of byzantine robots

To understand how well the presented approach could handle Byzantine robots, we varied their number in the swarm. When the number of Byzantines was increased, we kept the overall swarm size constant; that is, we did not add more robots to the swarm but changed the ratio between the number of Byzantines and non-Byzantines. Keeping the swarm size constant ensured that other variables, such as the average connectivity, did not change. Given a certain number of Byzantine robots, which specific robots in an experiment were Byzantine and which were non-Byzantine were randomly determined by a script at the beginning of each experiment (both in reality and in simulation).

Table 2. Arena sizes for experiments with constant robot density (experiments 2b and 2c).

Number of robots	Number of tiles	Arena size
8	22 × 22 = 484	1.10 m × 1.10 m = 1.21 m^2
16	31 × 31 = 961	1.55 m × 1.55 m ≃ 2.40 m^2
24	38 × 38 = 1444	1.90 m × 1.90 m = 3.61 m^2
48	54 × 54 = 2916	2.70 m × 2.70 m = 7.29 m^2
72	66 × 66 = 4356	3.30 m × 3.30 m = 10.89 m^2
96	76 × 76 = 5776	3.80 m × 3.80 m = 14.44 m^2
120	85 × 85 = 7225	4.25 m × 4.25 m ≃ 18.06 m^2

Type of byzantine fault

By default, in our experiments, a Byzantine robot sent a faulty local estimate of 0% white tiles to the smart contract, independent of its actual ground sensor readings. This fault was well motivated by our tests with physical robots and can occur both in laboratory environments as well as in real-world deployments—for example, in the following situations: a robot gets stuck on a black tile during the experiments because the wheels do not have enough grip and spin on the spot; a robot's ground sensor does not have the correct distance from the floor because of a loose screw; a software error occurs because of a crash of the communication protocol that causes the local estimates to remain at their initial value of 0; or the robot is controlled by a malicious entity that sends extreme values to work against the goal of the swarm. We also considered two additional types of Byzantine faults: In the first new Byzantine fault, Byzantine robots sent either 0 or 100% with an equal probability; in the second new fault, Byzantine robots sent a value randomly drawn from the uniform distribution between 0 and 100%. The two additional types of Byzantine faults were motivated by the following situations: A Byzantine robot might have no knowledge about the environment (for example, because its ground sensor or its motor failed), and therefore the robot might try to "play the lottery" and send random estimates; a partially broken sensor, a broken connection, dust, or a broken software interface could introduce noise, which, without further information, could be modeled by a uniform distribution or a Bernoulli distribution; or the robot is controlled by a malicious entity that tries to add noise to the swarm estimate.

Dimensions of the environment and layout of the floor

We used a 38 × 38 = 1444-tile arena as the default arena for our experiments. Each tile was either black or white and 5 cm by 5 cm in size, resulting in a 3.61-m^2 surface area. In addition, we created smaller and larger arenas for experiments 2b and 2c in which we kept the robot density constant across arenas of different sizes.

We chose the number of tiles per row (22, 31, 38, ...) in such a manner that it was possible to have arenas where the tile size remained constant and whose surface areas were n times as large as the surface area of the arena for eight robots, where n is approximately an integer. In this way, we could have swarm sizes that were multiples of 8 while keeping a constant robot density (see Table 2).

In experiment 2b, we used the three smaller arenas for runs with real robots and all the arena sizes in simulation. In experiment 2c, which was run in simulation only, we used all the arena sizes.

All arenas had 25% white tiles and 75% black tiles. For all arena sizes, the layout of the floor (defined by the position of the tiles) was chosen uniformly at random using a Python script. Once a layout was created, the same one was used for all experiments (with the exception of the first condition in experiment 1b, where a new layout was created for each run). For the real-robot experiments, we printed the floor layouts on thick paper.

Convergence criterion

By default, an experiment was stopped after all robots had received the convergence signal by querying the corresponding function of the smart contract. To evaluate the growth of the blockchain size over time, in experiments 2c and 2d, the convergence signal was disabled, and the experiments were allowed to run for 600 min.

Performance measures

To evaluate the results, we used the following two performance measures: absolute error and convergence time. In experiment 2c, we additionally measured the blockchain size, the inbound token flow, and the bandwidth utilization.

Absolute error

In all experiments, the goal of the swarm was to determine the fraction of white tiles. The swarm estimate was the aggregation of the local estimates of the individual robots, determined by the logic written in the smart contract. Because the purpose of the study was to determine whether the swarm reached a consensus on a value close to the correct one (25% white tiles in all experiments), for each run, we extracted the swarm estimate from only one robot (we used the last robot that received the convergence signal). If we were to extract the swarm estimates from all robots and aggregate them during our analysis, then we would not necessarily test the swarm's ability to reach consensus: This aggregation could average out the variance in the swarm estimates and constitute an additional post hoc consensus process.

The absolute error was the absolute difference between the swarm estimate and the actual fraction of white tiles. For example, if the swarm estimate is 27%, then the absolute error is $|25\% - 27\%| = 2\% = 0.02$.

In Figs. 2, 4, and 5, we also plotted the absolute error obtained with a very simple baseline algorithm. This baseline was calculated by averaging the individual estimates of the robots. The average was computed after a run was finished and did not use outlier detection or a blockchain. We could have chosen as a baseline the more sophisticated Byzantine linear consensus protocol (*37*, *38*). However, as we have shown in (*33*), this algorithm neither is resistant to Sybil attacks nor has many of the desirable properties of our blockchain-based algorithm, such as data logging, non-repudiation, and execution of decentralized programs.

Convergence time

We defined the convergence time as the time it took for all robots to receive the convergence signal from the smart contract. This convergence signal was set to true if the following two conditions were met: that the absolute difference in the swarm estimate between the previous proposal round and the current proposal round was smaller than 0.2% and that there were at least three proposal rounds. The second condition ensured that a minimum number of estimates were stored in the smart contract to increase confidence in the collective estimate.

Blockchain size

To determine the blockchain size, we measured the size in megabytes of the Ethereum data folder of the last robot that received the convergence signal.

Inbound token flow

The inbound token flow of a robot was the cumulative sum of all the payments in crypto tokens (UBI payments and reward payments) it received during the experiment. Robots spent their crypto tokens again as soon as they had sufficient tokens to create a valid sendEstimate(<localEstimate>) transaction; therefore, analyzing the wealth instead of the token flow of the robots would be futile.

Bandwidth utilization

For each robot i, we recorded the total size S_i (in kilobytes) of the sent messages and the total time T_i a robot i was communicating with at least another robot (that is, the time a robot was able to exchange information). From these data, we could obtain the mean bandwidth utilization (that is, the mean size of sent packets per second) both over the total runtime T of the experiment $(\sum_i^N (S_i/T)/N)$ and over the time T_i when the robots are communicating $(\sum_i^N (S_i/T_i)/N)$, where N is the number of robots in the swarm. Note that the total size of the sent packets equaled the total size of the received packets because each sent packet was received by another robot because it was a closed system without packet loss.

Supplementary Materials

This PDF file includes:

Supplementary Results
Supplementary Discussion
Supplementary Methods
Fig. S1
References (*51*–*54*)

REFERENCES AND NOTES

1. M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell.* **7**, 1–41 (2013).
2. M. Dorigo, M. Birattari, M. Brambilla, Swarm robotics. *Scholarpedia* **9**, 1463 (2014).
3. L. Bayındır, A review of swarm robotics tasks. *Neurocomputing* **172**, 292–321 (2016).
4. H. Hamann, *Swarm Robotics: A Formal Approach* (Springer, 2018).
5. M. Dorigo, G. Theraulaz, V. Trianni, Reflections on the future of swarm robotics. *Sci. Robot.* **5**, eabe4385 (2020).
6. M. Dorigo, G. Theraulaz, V. Trianni, Swarm robotics: Past, present, and future [point of view]. *Proc. IEEE* **109**, 1152–1165 (2021).
7. G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, R. Wood, The grand challenges of *Science Robotics*. *Sci. Robot.* **3**, eaar7650 (2018).
8. F. Higgins, A. Tomlinson, K. M. Martin, Survey on security challenges for swarm robotics, in *Proceedings of the Fifth International Conference on Autonomic and Autonomous Systems* (IEEE Press, 2009), pp. 307–312.
9. E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems* (Oxford Univ. Press, 1999).
10. M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J. L. Deneubourg, F. Mondada, D. Floreano, L. M. Gambardella, Evolving self-organizing behaviors for a *swarm-bot*. *Auton. Robots* **17**, 223–245 (2004).
11. A. F. T. Winfield, J. Nembrini, Safety in numbers: Fault-tolerance in robot swarms. *Int. J. Model. Identif. Control* **1**, 30–37 (2006).
12. A. L. Christensen, R. O'Grady, M. Birattari, M. Dorigo, Fault detection in autonomous robots based on fault injection and learning. *Auton. Robots* **24**, 49–67 (2008).
13. A. L. Christensen, R. O'Grady, M. Dorigo, From fireflies to fault-tolerant swarms of robots. *IEEE Trans. Evol. Comput.* **13**, 754–766 (2009).
14. J. D. Bjerknes, A. F. T. Winfield, On fault tolerance and scalability of swarm robotic systems, in *The 10th International Symposium on Distributed Autonomous Robotic Systems (DARS 2013)*, A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. Ani Hsieh, L. E. Parker, K. Støy, Eds. (Springer, Germany, 2013), vol. 83, pp. 431–444.
15. D. Tarapore, A. L. Christensen, J. Timmis, Generic, scalable and decentralized fault detection for robot swarms. *PLOS ONE* **12**, 1–29 (2017).
16. D. Tarapore, P. U. Lima, J. Carneiro, A. L. Christensen, To err is robotic, to tolerate immunological: Fault detection in multirobot systems. *Bioinspir. Biomim.* **10**, 016014 (2015).
17. J. O'Keeffe, D. Tarapore, A. G. Millard, J. Timmis, Adaptive online fault diagnosis in autonomous robot swarms. *Front. Robot. AI* **5**, 131 (2018).
18. D. Tarapore, J. Timmis, A. L. Christensen, Fault detection in a swarm of physical robots based on behavioral outlier detection. *IEEE Trans. Robot.* **35**, 1516–1522 (2019).
19. V. Strobel, E. Castelló Ferrer, M. Dorigo, Managing Byzantine robots via blockchain technology in a swarm robotics collective decision making scenario, in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, M. Dastani, G. Sukthankar, E. André, S. Koenig, Eds. (International Foundation for Autonomous Agents and Multiagent Systems, 2018), pp. 541–549.
20. A. Aswale, A. López, A. Ammartayakun, C. Pinciroli, Hacking the colony: On the disruptive effect of misleading pheromone and how to defend against it, in *Proceedings of the 21st*

International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M. E. Taylor, Eds. (International Foundation for Autonomous Agents and Multiagent Systems, 2022), pp. 27–34.
21. E. Castelló Ferrer, T. Hardjono, A. Pentland, M. Dorigo, Secure and secret cooperation in robot *swarms*. *Sci. Robot.* **6**, eabf1538 (2021).
22. G. Primiero, E. Tuci, J. Tagliabue, E. Ferrante, Swarm attack: A self-organized model to recover from malicious communication manipulation in a swarm of simple simulated agents, in *Swarm Intelligence – Proceedings of ANTS 2018 – Eleventh International Conference*, M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, V. Trianni, Eds. (Springer, 2018), pp. 213–224.
23. G. Maître, E. Tuci, E. Ferrante, Opinion dissemination in a swarm of simulated robots with stubborn agents: A comparative study, in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2020)* (IEEE Press, 2020), pp. 1–6.
24. I. Sargeant, A. Tomlinson, Maliciously manipulating a robotic swarm, in *Proceedings of ESCS'16 – The 14th International Conference on Embedded Systems, Cyber-physical Systems, & Applications* (CSREA Press, 2016), pp. 122–128.
25. E. R. Hunt, S. Hauert, A checklist for safe robot swarms. *Nat. Mach. Intell.* **2**, 420–422 (2020).
26. L. Lamport, R. Shostak, M. Pease, The Byzantine generals problem. *ACM Trans. Programm. Lang. Syst. (TOPLAS)* **4**, 382–401 (1982).
27. M. Castro, B. Liskov, Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.* **20**, 398–461 (2002).
28. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system" (Technical Report, 2008); https://bitcoin.org/bitcoin.pdf [accessed 11 August 2018].
29. V. Buterin, "A next-generation smart contract and decentralized application platform. Ethereum project white paper" (Technical Report, Ethereum Foundation, 2014); https://github.com/ethereum/wiki/wiki/White-Paper [accessed 18 July 2019].
30. E. Castelló Ferrer, The blockchain: A new framework for robotic swarm systems; https://doi.org/10.48550/arXiv.1608.00695 (2016).
31. V. Strobel, M. Dorigo, Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation, in *Swarm Intelligence – Proceedings of ANTS 2018 – Eleventh International Conference*, M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, V. Trianni, Eds. (Springer, 2018), vol. 11172 of *Lecture Notes in Computer Science*, pp. 425–426.
32. J. Peña Queralta, L. Qingqing, Z. Zou, T. Westerlund, Enhancing autonomy with block- chain and multi-access edge computing in distributed robotic systems, in *Proceedings of the 5th International Conference on Fog and Mobile Edge Computing (FMEC 2020)* (IEEE Press, USA, 2020), pp. 180–187.
33. V. Strobel, E. C. Ferrer, M. Dorigo, Blockchain technology secures robot swarms: A comparison of consensus protocols and their resilience to Byzantine robots. *Front. Robot. AI* **7**, 54 (2020).
34. A. Reina, Robot teams stay safe with blockchains. *Nat. Mach. Intell.* **2**, 240–241 (2020).
35. A. Pacheco, V. Strobel, A. Reina, M. Dorigo, Real-time coordination of a foraging robot swarm using blockchain smart contracts, in *Swarm Intelligence – Proceedings of ANTS 2022–Thirteenth International Conference* (Springer, Germany, 2022), vol. 13491 of *Lecture Notes in Computer Science*, pp. 196–208.
36. M. G. Santos De Campos, C. P. Chanel, C. Chauffaut, J. Lacan, Towards a blockchain-based multi-UAV surveillance system. *Front. Robot. AI* **8**, 557692 (2021).
37. L. Guerrero-Bonilla, A. Prorok, V. Kumar, Formations for resilient robot teams. *IEEE Robot. Autom. Lett.* **2**, 841–848 (2017).
38. D. Saldaña, A. Prorok, S. Sundaram, M. F. M. Campos, V. Kumar, Resilient consensus for time-varying networks of dynamic agents, in *Proceedings of the American Control Conference (ACC)* (IEEE Press, 2017), pp. 252–258.
39. G. Valentini, D. Brambilla, H. Hamann, M. Dorigo, Collective perception of environmental features in a robot swarm, in *Swarm Intelligence – Proceedings of ANTS 2016 – Tenth International Conference*, M. Dorigo, *et al.*, eds. (Springer, Cham, Switzerland, 2016), vol. 9882 of *Lecture Notes in Computer Science*, pp. 65–76.
40. G.-T. Nguyen, K. Kim, A survey about consensus algorithms used in blockchain. *J. Inf. Process. Syst.* **14**, 101–128 (2018).
41. A. G. Millard, R. Joyce, J. A. Hilder, C. Fleşeriu, L. Newbrook, W. Li, Liam J. Mc Daid, D. M. Halliday, The Pi-puck extension board: A Raspberry Pi interface for the e-puck robot platform, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE Press, 2017), pp. 741–748.
42. F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J-.C. Zufferey, D. Floreano, A. Martinoli, The e-puck, a robot designed for education in engineering, in *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, P. J. S. Gonc¸alves, P. J. D. Torres, C. M. O. Alves, Eds. (Instituto Politécnico de Castelo Branco, 2009), pp. 59–65.
43. E. B. Hamida, K. L. Brousmiche, H. Levard, E. Thea, Blockchain for enterprise: Overview, opportunities and challenges, in *The Thirteenth International Conference on Wireless and Mobile Communications (ICWMC 2017),* Nice, France, 23 to 27 July 2017, pp. 83–88.
44. P. Szilágyi, EIP 225: Clique proof-of-authority consensus protocol (2017); https://github.com/ethereum/EIPs/issues/225 [accessed May 10, 2020].
45. E. Drasutis, "IOTA smart contracts" (Technical Report, IOTA Foundation, 2021); https://files.iota.org/papers/ISC_WP_Nov_10_2021.pdf [accessed 13 April 2023].
46. J. Polge, J. Robert, Y. Le Traon, Permissioned blockchain frameworks in the industry: A comparison. *ICT Express* **7**, 229–233 (2021).
47. H. Pervez, M. Muneeb, M. U. Irfan, I. U. Haq, A comparative analysis of DAG-based blockchain architectures, in *Proceedings of the 12th International Conference on Open Source Systems and Technologies (ICOSST 2018)* (IEEE Press, 2018), pp. 27–34.
48. C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. D. Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, M. Dorigo, ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intell.* **6**, 271–295 (2012).
49. L. Garattoni, G. Francesca, A. Brutschy, C. Pinciroli, M. Birattari, "Software infrastructure for E-puck (and TAM)" (Technical Report 2015-004, IRIDIA, Univ. libre de Bruxelles, 2015).
50. K. Hasselmann, A. Parravicini, A. Pacheco, V. Strobel, Python wrapper for ARGoS 3 simulator; https://github.com/KenN7/argos-python/ (2021).
51. V. Buterin, V. Griffith, Casper the friendly finality gadget; https://doi.org/10.48550/arXiv.1710.09437 (2017).
52. A. Pacheco, V. Strobel, M. Dorigo, A blockchain-controlled physical robot swarm communicating via an ad-hoc network, in *Swarm Intelligence – Proceedings of ANTS 2020 – Twelfth International Conference*, M. Dorigo, T. Stützle, M. J. Blesa, C. Blum, H. Hamann, M. K. Heinrich, V. Strobel, Eds. (Springer, 2020), vol. 12421 of *LNCS*, pp. 3–15.
53. D. Merkel, Docker: Lightweight Linux containers for consistent development and deployment. *Linux J.* **2014**, 2 (2014).
54. A. Pacheco, V. Strobel, M. Dorigo, "A framework for swarm robotics experimentation with Pi-puck robots and an Ethereum-based blockchain" (Technical Report TR/IRIDIA/2020-001, IRIDIA, Univ. Libre de Bruxelles, Brussels, Belgium, 2020).

Acknowledgments: We thank M. K. Heinrich for providing useful comments and for proofreading this manuscript. **Funding:** This work was partially supported by the Program of Concerted Research Actions (ARC) of the Université Libre de Bruxelles. V.S. and M.D. acknowledge support from the Belgian F.R.S.-FNRS, of which they are a research fellow and a research director, respectively. V.S. also gratefully acknowledges support from the Fondation Jaumotte-Demoulin and the Fonds David et Alice Van Buuren. A.P. acknowledges support via a fellowship from the Faculty of Applied Sciences of the Université Libre de Bruxelles. **Author contributions:** All authors contributed to the conceptualization of this research and to the setup of the experiments. A.P. and V.S. implemented the robots' controllers and the experimental setup. V.S. executed the experiments and analyzed the results. The manuscript was written by V.S. and M.D. and revised by all authors. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All collected data are available as CSV files on Dryad under doi: 10.5061/dryad.zcrjdfnj3. Contact V.S. at volker.strobel@ulb.be for any questions. The real-robot experiments were recorded on videos that can be requested from the authors.

Submitted 16 February 2022
Resubmitted 18 May 2022
Accepted 27 May 2023
Published 28 June 2023
10.1126/scirobotics.abm4636

Science Robotics

Robot swarms neutralize harmful Byzantine robots using a blockchain-based token economy

Volker Strobel, Alexandre Pacheco, and Marco Dorigo

Sci. Robot. **8** (79), eabm4636. DOI: 10.1126/scirobotics.abm4636

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.