Collective Behavior

Secure and secret cooperation in robot swarms

Eduardo Castelló Ferrer1,2,3*, Thomas Hardjono2, Alex Pentland1,2, Marco Dorigo3

The importance of swarm robotics systems in both academic research and real-world applications is steadily increasing. However, to reach widespread adoption, new models that ensure the secure cooperation of large groups of robots need to be developed. This work introduces a method to encapsulate cooperative robotic missions in an authenticated data structure known as a Merkle tree. With this method, operators can provide the “blueprint” of the swarm’s mission without disclosing its raw data. In other words, data verification can be separated from data itself. We propose a system where robots in a swarm, to cooperate toward mission completion, have to “prove” their integrity to their peers by exchanging cryptographic proofs. We show the implications of this approach for two different swarm robotics missions: foraging and maze formation. In both missions, swarm robots were able to cooperate and carry out sequential tasks without having explicit knowledge about the mission’s high-level objectives. The results presented in this work demonstrate the feasibility of using Merkle trees as a cooperation mechanism for swarm robotics systems in both simulation and real-robot experiments, which has implications for future decentralized robotics applications where security plays a crucial role.

INTRODUCTION

Swarm robotics systems (1) have the potential to revolutionize many industries, from targeted material delivery (2) to precision farming (3, 4). Boosted by technical breakthroughs, such as cloud computing (5, 6), novel hardware design (7–9), and manufacturing techniques (10, 11), swarms of robots are envisioned to play an important role in both industrial (12) and urban (13, 14) activities. The emergence of robot swarms has been acknowledged as 1 of the 10 robotics grand challenges for the next 5 to 10 years that will have substantial socioeconomic impact. Despite having such a promising future, many important aspects that need to be considered in realistic deployments are either undersampled or neglected (15).

One of the main reasons why swarms of robots have not been widely adopted in real-world applications is because there is no consensus on how to design swarm robotics systems that include perception, action, and communication among large groups of robots (15). In addition, recent research has pointed out that the lack of security standards in the field is also hindering the adoption of this technology in data-sensitive areas (e.g., military, surveillance, and public infrastructure) (16, 17). These research gaps are motivating scientists to focus on new fields of study, such as applied swarm security (18, 19) and privacy (20, 21), and to revisit already accepted assumptions in the field.

From the origins of swarm robotics research, robot swarms were assumed to be fault tolerant by design because of the large number of robot units involved (22–25). However, it has been shown that a small number of partially failed (with defective sensors, broken actuators, noisy communications devices, etc.) (26) or malicious robots (27, 28) can have a substantial impact on the overall system reliability and performance. The first surveys of swarm robotics security were presented in (29, 30). These works identified physical capture and tampering with members as serious threats to robot swarms. Physical capture of a robot might lead not only to loss of availability but also to the capture of security credentials or critical details about the swarm operation (31). For instance, if a robot is tampered with and reintroduced into the swarm, then an attacker might influence the behavior of the whole system (25, 27) and eventually hinder the entire mission. Figure 1 depicts an example of such vulnerability, where four tasks must be performed in a sequential order by a group of robots: First, block (1) needs to be pushed to an assembly area. Second, block (2) has to be placed on top of (1) to assemble a tower-shaped structure. Third, the state of a bridge that connects the assembly area and the target location needs to be determined as crossable (i.e., lowered down). Fourth, the tower-shaped structure must be pushed to the target location at the other side of the bridge. If a robot is tampered with (e.g., the robot in charge of determining the status of the bridge), then it can spread erroneous information within the system (e.g., tell the other robots that the bridge is crossable when it is not), therefore hindering the entire mission. This type of attack would be unique to swarm and multirobot systems and is particularly critical in situations where robots must share data with other robots in the swarm or with human operators.

In previous swarm robotics work, researchers hard-coded the complete set of rules that trigger the transitions from task to task in all robots (24, 32, 33); that is, each robot in the swarm had a full copy of the information necessary to accomplish its mission. Although this distributed approach is more robust and fault tolerant than centralized methods, it substantially increases the attack surface (i.e., the total sum of vulnerabilities) for an attacker to figure out the swarm’s high-level goals and disrupt the system’s behavior (31). Because of these concerns, in this work, we aim to find an answer to the following questions: Is there a way to provide the “blueprint” of a robotic mission without disclosing the raw information describing the mission itself? In other words, is it possible for robots in a swarm to fulfill sequential missions without exposing knowledge about the mission’s high-level objectives? To answer these questions, we present a cooperation model based on the idea of encapsulating robotic missions into Merkle trees (MTs). Our approach allows robots in a swarm to cooperate while minimizing security risks due to issues such as physical capture or tampered members. We achieve this by creating a secure data object that is shared by the robots (i.e., an MT) instead of protecting the communication channel among them (e.g., by encrypting the network connections).
Fig. 1. Toward secure and secret cooperation in swarm robotics missions. Example of a sequential mission (e.g., build a two-block tower and transport it to a target location) conducted by a robot swarm that can fail if one of the robots involved is tampered with. If a robot spreads erroneous information within the system (e.g., the robot in charge of making the bridge crossable sends a message confirming that the bridge is lowered when it is not), then the entire mission can fail.

A set of 100 simulation experiments were carried out to analyze our approach in the foraging scenario shown in Fig. 2C. This approach was analyzed in simulation and real-robot experiments for two different sequential missions: foraging and maze formation. In both missions, robots were able to cooperate and carry out sequential tasks without having explicit knowledge about the mission’s high-level objectives. Our findings show that larger swarms tend both to increase the performance of the system and to diversify the amount of information within the swarm. However, larger numbers of robots and longer missions determine a linear increase in the communication requirements of the system. Nevertheless, an analysis of storage usage, communication costs, and computational time for larger-scale missions where the number of tasks takes relatively large values reveals that the use of MTs is within reach of current robotic technology.

The remainder of this paper is structured as follows. Results presents the outcome of our experiments for two different swarm robotics missions: foraging and maze formation. In this section, we show the feasibility of using MTs as a cooperation mechanism for swarm robotics systems in both simulation and real-robot experiments. Discussion analyzes the results obtained and proposes future research directions. Materials and Methods describes in detail the concept of MT and its applications for robot swarms missions. Moreover, this section introduces the concept of MT proof and the analysis metrics used to evaluate the proposed approach. Sections S1 and S2 describe the setup of our experiments including the arena, robots, and simulator used to evaluate the proposed approach. Section S3 presents the control logic for the robots in both missions. Last, section S4 presents a futuristic scenario where the research proposed here can be monetized.

#### RESULTS

**Foraging mission experiments**
A set of 100 simulation experiments were carried out to analyze our approach in the foraging scenario shown in Fig. 3A. The foraging
mission consists of a sequence of tasks to be executed by e-puck robots (Fig. 3B) (35). Each task consists of finding an object of a given color and retrieving it to a target location. The environment is a rectangular area divided into a five-by-five grid, in which objects are represented as colored cells and the center cell represents the target location. The foraging mission is finished when all tasks in the sequence are completed; that is, all colored objects have been delivered in the right order at the target location. In other words, when a robot in the swarm delivers the last object in the sequence. The sequence of colors is encoded by the swarm’s operator as an MT (Fig. 2A) with \( n \) tasks before the experiment starts (Fig. 2B). Therefore, during the mission, robots do not have explicit information about what is the correct sequence of colors to be delivered because they only have the encrypted information included in their MT.

In addition, 10 real-robot experiments for a subset of the experimental conditions considered in the simulation experiments were carried out to validate the approach. Figure 3C shows the real-robot foraging environment where objects are represented by colored cells projected on the arena by using a projector (see the Supplementary Materials for a detailed description of the real-robot experimental setup). A typical real-world run is displayed in movie S2.

In the foraging mission, robots search the environment looking for objects. Once an object is found, robots identify its color and generate \( h_o \). Then, they generate the value \( h_o \) for all possible actions (note that in the considered foraging mission, \( h_s \) is restricted to the “carry to target” action). \( H_i \) is calculated by hashing \( h_o \) and \( h_a \) so as to move to the target location. If not, then the robot keeps wandering. When a task is completed successfully, the robot updates \( i \) so as to move to the next task in its local MT: \( i = i + 1 \) for \( i \in (0 \leq i \leq n - 1) \). During the experiments, robots can exchange information with their peers (e.g., \( i \) and \( \pi \)). If two robots are within communication distance, then they can exchange their \( i \) values. If there is a disparity in the values (i.e., one \( i \) is lower than the other), then the robot with the lower \( i \) will become a verifier (V) and will send queries (Q) to the robot with greater \( i \), the prover (P), asking for proofs (\( \pi \)) about the missing tasks as depicted in Fig. 2C. This process is conducted until the \( i \) value is the same for both robots. By using this method, robots can synchronize their own
MT copies and therefore cooperate toward the fulfillment of the swarm’s mission (see the Supplementary Materials for robot’s interaction and finite state machine diagrams). Sequentiality is preserved because a robot with proof for task $k$, to convince a robot that is searching for task $i$ to switch to task $k$, should also provide all the intermediate proofs for tasks $i, i+1, ..., k-1$. Note that another way to ensure sequentiality would be to add the $h_a$ and $h_i$ values of previous tasks to the current working leaf [$H_i = H_i(h_a, h_i, h_{a,i-1}, h_{i-1})$]. In this way, the swarm operator can ensure (this time, from the MT perspective) that previous tasks in the mission were completed successfully before executing new ones (as required in the example of Fig. 1).

Next, we study how the performance of our approach (i.e., how fast and reliably a particular mission is carried out) changes when varying the MT length and the swarm sizes. We present results obtained in simulation and real-robot experiments where the MT length $n$ is varied in the set $\{2, 4, 5, 6, 7, 8\}$ (simulation) and $\{2, 4, 8\}$ (real) and the swarm sizes $R_n$ in the interval $1 \leq R_n \leq 16$. In addition, the amount of communication ($AC$; i.e., the total amount of data that robots exchanged during the mission) and information diversity ($I_e$; i.e., how widely spread information is within the swarm) metrics are calculated and presented. Details about these analysis metrics and the statistical methods used can be found in Materials and Methods.

Figure 4A shows the finishing time ($F_t$) and its SD for several MT length configurations ($n$) and robot swarm sizes ($R_n$) for both simulation and real-robot experiments. According to Fig. 4A, $F_t$ decreases when robots are added to the swarm, regardless of the length of the MT. These results also suggest that once a certain number of robots is present ($R_n \geq 8$), the length of the MT ($n$) has little impact on the $F_t$ of the system. Figure 4B shows how the probability of success ($P_s$) curves change when changing the number $n$ of tasks for all simulation configurations presented in Fig. 4A. One can also see that adding robots increases $P_s$ because the curves become steeper and converge to higher values sooner. However, these results also suggest that as we increase $n$ (the mission becomes longer), $P_s$ reaches higher values later (especially for $R_n \leq 3$). This effect can be seen in both simulation (Fig. 4, C to E) and real-robot experiments (Fig. 4, F to H). $P_s$ does reach 1 (the maximum value) in all configurations because all experiments terminated with success before reaching the maximum amount of allowed experiment time, also known as time cap (TC).

Figure 5 shows average results and SDs for $AC$ in kilobytes and information diversity measured by Shannon’s equitability index ($I_e$) for different simulation and real-robot experiments. In particular,
Fig. 5A reports simulation results for the $R_n$ ($1 \leq R_n \leq 16$) and $n \in [2,4,6,8]$ configurations, whereas Fig. 5B reports results of real-robot experiments for $R_n = [2,4,8,16]$ and $n \in [2,4,8]$ configurations. In both figures, $I_e \in [0, 1]$ gives information on the variability in the number of tasks performed by each robot in the swarm during the mission. Lower values indicate more uneven distributions (e.g., one single robot completing all tasks), whereas higher values indicate more uniform distributions (e.g., all robots completing the same number of tasks). Figure 5 (A and B) also shows that AC increases linearly with $R_n$ because there are more robots exchanging MT proofs. Moreover, MTs with larger $n$ values make the proofs that robots exchange “heavier” (i.e., $\pi$ is longer), which also contributes to increased AC. However, larger $R_n$ values tend to increase information evenness ($I_e$) in the swarm. In other words, the execution of tasks is more evenly distributed among robots. Results for both simulation and real-robot experiments suggest that, as we increase $R_n$, the distribution of tasks fulfilled by the robots tends to become more uniform. Figure 5 also suggests that information diversity might need a really large $R_n$ value to converge to 1 (i.e., to converge to complete “evenness”).

**Maze formation mission experiments**

In the second experiment, we gave the robot swarm a maze formation mission (Fig. 6, A to C) where we conducted 100 simulations and 15 real-robot experiments for each considered configuration. Figure 6A shows the blueprint of a five-by-five maze where 0 represents an empty space, 1 represents a wall, and * and @ represent the entrance and the exit of the maze, respectively. As in the foraging mission, robots first wander around the arena. However, instead of looking for colored cells, in this mission, robots search for $(x, y)$ maze coordinates. By knowing the cell dimensions (0.5 m by 0.5 m), robots can calculate the $(x, y)$ coordinates of the cells in which they are located. Every time a robot discovers a new $(x, y)$ coordinate pair [e.g., $(1, 1)$], it uses the new coordinate pair to generate $h_i$. Then, it generates the value $h_s$ for all possible actions (note that in the considered maze formation mission, $h_i$ is restricted to the “stop” action). Like in the foraging mission, $H_i$ is calculated by hashing $h_a$ and $h_s$, $H_i = H(h_a, h_s)$, and verified against the current $i$ of the robot.

In case it is possible to generate a valid proof $\pi$ with this information $[3\pi(h_i)]$, the robot finds the center of the cell and stops there (Fig. 6, B and C). If not, the robot keeps wandering. In the same way as in the foraging mission, robots avoid already completed tasks (in this case, they avoid stopping in already occupied cells) by receiving the proof that $i$ was already completed. As with the foraging experiment, the maze formation mission terminates once all tasks have been completed. Additional information about the robots’ behavior in the maze formation mission can be found in the Supplementary Materials.

In our experiments, the number of tasks $n$ was fixed to 16 to match the number of cells where the value 1 is present in Fig. 6A, and the number $R_n$ of robots was varied in the range $16 \leq R_n \leq 28$.

Figure 6D shows average $F_t$ and SDs for the maze formation mission for both simulation and real-robot experiments: As was the case in the corresponding foraging experiment (Fig. 4A), larger $R_n$ values reduce $F_t$. However, beyond a certain $R_n$ value ($R_n \geq 24$), the impact on $F_t$ becomes very small. Figure 6E shows the AC and $I_e$ metrics for the maze formation mission. Figure 6E follows the same pattern as Fig. 5: AC increases linearly with $R_n$. Figure 6E includes the AC upper limit following Eq. 1 (see the “Statistical methods” section). According to Fig. 6E, both simulation and real-robot AC fall under the limit and suggest that real-robot experiments require a lower AC...
than their simulation counterparts. Last, Fig. 6E shows that in this scenario, complete evenness of information (i.e., $I_e = 1$) is achieved.

**Experiments with larger-dimension missions**

Encouraged by these results, we found that it is appropriate to analyze the feasibility of our approach in complex missions where the number of tasks takes relatively large values. Figure 7 shows different LEGO models where a sequential set of tasks is required to achieve the final outcome (i.e., build the replica). These three models have the largest piece count according to the current LEGO catalog and are good examples of potential large-scale missions, especially because $n$ takes a relatively large value. Because of the possibility of accurately calculating the maximum AC ($AC_{min}$) required to make all the robots complete their MTs (see the “Statistical methods” section and Eq. 1) and the overall size of the MT stored by the robots, we can compute the corresponding MTs and measure their memory and maximum AC per robot ($AC_{min}/R_n$) requirements in Fig. 7. We also computed the average computation time to generate the complete MT ($G$) and the time for the generation of a proof ($P$) and for the verification of a proof ($V$). Results for the aforementioned models are given in Table 1.

**DISCUSSION**

We showed that two of the main MT properties (i.e., correctness and security) open a new path toward secure and secret cooperation in robot swarms. Regarding the security aspect, by using this approach, to cooperate, the robots in a swarm are required to prove to their peers that they fulfilled certain actions or that they know or own some particular information [i.e., proof of ownership (36)], rather than merely rely on information received from other robots (sensor data, votes, etc.). This approach makes robots resistant against potential threats such as tampering attacks because any change in the task’s data (i.e., $s$ and $a$) will necessarily change the proof’s outcome. Regarding the secrecy component, with the use of MTs, robot swarms are now able to separate the mission data from its verification. This allows robots to verify that a task was carried out by a member of the swarm without knowing what this task entailed or which robot took part in its completion. This makes physical capture attacks inefficient because individual robots might not have enough raw or unprotected information to describe the high-level swarm’s missions and goals, especially in large systems.

This approach was tested in two different scenarios: a foraging and a maze formation mission. In the foraging case, results suggest that increasing the swarm size has a positive impact on the performance of the system: larger swarm sizes have lower finishing time ($F_n$) and a higher $P_n$. Results also show that AC grows linearly with the swarm size ($R_n$), which, in extreme situations (e.g., very large swarms), could be detrimental to the system because individual robots might not be able to cope with the bandwidth requirements. In contrast, increasing $R_n$ has the positive effect of increasing the information diversity ($I_e$) because we are increasing the probability of reaching more uniform distributions of completed tasks within the swarm.

In the maze formation mission, where $R_n$ and $n$ take larger values, results also suggest that $R_n$ maintains an inverse relationship with $F_n$. In the maze formation mission, $I_e$ is maximized (i.e., $I_e = 1$), which is possible because in this mission, when robots find a cell where they can generate a valid $\pi$ proof, they stop at its center, in this way making robots capable of fulfilling only one task per mission. This is different from the foraging scenario, where one robot might be able to complete several tasks. Although AC grows linearly with $R_n$, this still does not represent a challenging situation for the swarm (e.g., 72 and 70 kilobytes are exchanged in a 28-robot system in simulation and real-robot experiments, respectively). Counterintuitively, real-robot experiments require a lower AC to complete the maze formation mission than their simulation counterparts. This effect is due to the incapability of the simulation engine to take into account network lag: Delays in the network produce situations where robots are able to find and complete the mission, whereas other robots in the swarm are still trying to synchronize on previous completed tasks. Last, because of these properties, robot swarms can complete complex missions such as the maze formation one without the means to infer high-level details such as where the entrance or the exit might be located.

Last, Table 1 shows that neither the memory, nor the communication requirements per robot, nor the average computation times of the corresponding MTs are out of reach of current commodity hardware (e.g., Overo Gumstix), and therefore, our approach is feasible for current robot platforms. More than 99% of the computation time is taken by the generation ($G$) of the MT that only takes place at the beginning of the mission, whereas the proof generation ($P$) and verification ($V$) take a nearly negligible amount of time.

**Limitations**

The complete set of tasks (and its order) must be known at design time

As described previously, the presented approach needs a swarm operator to design and encode the complete sequence of tasks included in the MT before the mission starts. Therefore, any deviation from the original sequence included in the MT might be difficult to adapt to. It is however possible to make incremental updates to the mission and encapsulate the outdated MT within a newer version that takes into account all necessary changes. In that situation, a new MT root might be necessary to reflect the updated mission sequence. However, the most critical point for this solution is to trust the source that distributes the new root node hash. Fortunately, this can be done by adding a security layer in the communication between...
the swarm operator and the swarm itself or even by exploiting new methods, such as blockchain technology, that allows secure data sharing between robots and untrusted sources (27, 28).

**Generation of a valid MT proof does not imply execution**

Another limitation of our current implementation is that even if a robot discovers a valid combination of $h_a$ and $h_s$ values, thus being able to generate a valid MT proof, there is no guarantee that the robot executes the corresponding task in the physical world (e.g., it delivers discovered objects to the target location or stops at the right coordinates). The reason behind this limitation is that, in the demonstration scenarios presented in this research, the verification of a completed task is omitted. However, it would be possible to overcome this problem by adding additional variables to the task-encoding process, forcing the robot to do the right action $a$, with the right sensor input $s$, at the right location $l$: $H_i = H(h_{a,i}, h_{s,i})$. Another possible way to tackle this problem could be the addition of robot “validators,” which verify that every task claimed by the robots is completed.

**Future work**

The increasing attention that swarm robotics is attracting suggests a future in which swarms run by different types of institutions (e.g., private and public) will coexist in a same location (e.g., urban and disaster areas, industrial environments, and battlefield) without exposing sensitive data about their internal processes, goals, and organizations. The findings presented here shed light on the design and implementation of sequential missions in potential applications for distributed robotics such as manufacturing [e.g., collaborative assembly (37)], logistics [e.g., supply chain in warehouses (38)], and security [e.g., surveillance of critical infrastructure (39)]. Perhaps, one of the most promising directions for future research is given by the possibility to implement missions in which heterogeneous agents can work together without having explicit knowledge about the mission’s high-level objectives. For instance, a food company might want to hire robots “on demand” for their processing plant without disclosing the recipe of their products (e.g., due to copyright or intellectual property issues). Along those lines, future research could be extended to man-machine systems, logistic chains, and similar scenarios where coordination is required but the overall mission’s goal should remain secret. In contrast, applications where real-time capabilities are important to fulfill the mission (e.g., motion coordination and collision avoidance) might not be suitable for the method proposed. This is because robots need to discover the correct combination of $h_a$ and $h_s$ to execute a task, which can make it impossible to guarantee that robots complete a task within a fixed amount of time. Last, the possibility of encapsulating the robot commands within an MT and to get a cryptographic proof that the job has been completed opens up the possibility of new type of services. In the last section of the Supplementary Materials, we describe an early-stage example in this direction where a blockchain-based marketplace allows a robot swarm to form shapes on demand. In the example, potential customers upload an MT describing the desired shape and send it to a smart contract in the Ethereum blockchain, which returns a proof and multimedia material when the work is completed by the robot swarm.

**MATERIALS AND METHODS**

**MTs for robot swarms**

An MT (35) is a hash-based structure where data are not stored in the inner nodes but in the leaves (Fig. 8A). An MT of depth $d$ is a tree with $m = 2^{d-1}$ leaf nodes, each one with an index $i \in [0, m - 1]$. A depiction of a complete MT for $d = 3$ is given in Fig. 8A. Each interior node contains the hash of the combination of its two children. Each leaf node represents a task and encapsulates the combined hash of two hashes: $h_a$ (hash of the robot action $a$) and $h_s$ (hash of the sensor’s input $s$). For instance, the hash of the action “push” ($h_a$) and the hash of the sensor input “block (1)” ($h_s$) would be included in one of the tasks by using the hashing function $H: H(h_{a,i}, h_{s,i})$.

At the beginning of the mission, all robots aim to fulfill the first task. However, only one will succeed in doing so. The successful robot first wanders around the environment. Second, when it perceives a new sensor input [e.g., block (1)], it generates a hash ($h_s$) from it. Third, it generates another hash ($h_a$) for an action that it can perform (e.g., push). Fourth, it combines $h_a$ and $h_s$ and rehashes them to obtain the hash string $H(h_{a,i}, h_{s,i})$. Fifth, it matches $H(h_{a,i}, h_{s,i})$ against

---

Fig. 8. Description of a typical MT structure and proof. (A) MT implementation ($d = 3$) with four leaf, two interior, and one root nodes. Each leaf node (green) encapsulates the hash of two hashes: a robot action ($h_a$) and a sensor input ($h_s$). Each leaf node represents one of a series of tasks that robots should complete sequentially ($i = 0, i = 1, \ldots, i = m - 1$) to fulfill the swarm’s mission. Interior nodes encapsulate the hash of their two children, and the root node encapsulates the hashes of its two interior nodes. (B) Describes the path (in color) to get the proof for leaf node $i = 1$. 

---
the hash value that corresponds to the first task of the MT. At this point, the robot executes the resulting command (e.g., “push block (1)”). The robot that fulfills this task can regard the task as completed. Once the task is completed, this robot (prover) has the ability to generate cryptographic proofs (π) that demonstrate to other (verifier) robots that it was aware of the information necessary to carry out the task because it has information that it could not have otherwise (i.e., $h_a$ and $h_b$).

Figure 8B shows the proof $\pi$ for a fetch at task $i = 1$. It consists of four elements in sequence, the two hashes $h_a$ and $h_b$, the hash $H_4$, and the hash $H_3$. Then, the verification proceeds bottom up: Verifier computes the hash of the two hashes $h_a$ and $h_b$, which is $H_5$, and concatenates it with the hash of $H_4$ provided in $\pi$. Next, it concatenates $H_4$ and $H_5$ and computes the hash of the digest for node $H_2$. Then, it concatenates $H_3$ provided in $\pi$, with the computed $H_2$, and hashes the result. Last, the verifier checks whether this computed digest equals $H_1$ (root node hash). In this case, the proof $\pi$ is a list of hashes that verifier robots can use to calculate the MT root node hash (shared by all robots in the swarm) by using $h_a$ and $h_b$ as inputs. These proofs can be sent and received by robots in the swarm, allowing them to synchronize, check, update, and complete their corresponding MT until the whole mission is completed. These proofs can be exchanged without disclosing any raw or unprotected data about the mission itself.

**Statistical methods**

To evaluate and analyze our approach, we rely on three metrics. 

**Performance**

These measures show the speed and reliability with which a mission is carried out. Here, we use the mission’s finishing time $F_1$ to measure the amount of time required to fulfill the swarm’s mission. In addition, we use an estimate $P_i$ of the probability that the system attains its target objective in an amount of time $t$ (40). Formally, let $j \in \{1, \ldots, k\}$ be the index of an experiment, $F_{ij}$ be the finishing time of experiment $j$, and experiments where $F_{ij} < T_C$ be considered successful experiments, where $T_C$ is the maximum amount of allowed experiment time. The estimate $P_i$ of the probability of success of the system over time (up to $T_C$) is defined as $P_i(t \leq t) = \{ | F_{ij} \leq t \}/k$.

**Amount of communication**

The AC is computed by multiplying the number of times the prover-verifier workflow (Fig. 2C) takes place during the mission by the size (in bytes) of the proof ($\pi$) the robots exchanged. When the MT is perfectly balanced (i.e., the number $m$ of leaves is a power of 2), the size of the proof $\pi$ is $\log_2(m) + 2$: the number of hashes to reach the root node plus the $h_a$, $h_b$ hashes. In missions with $n$ tasks where all robots need to complete their MTs, an upper limit $A_{cal}$ of the AC metric can be calculated with the following equation

$$A_{cal} = P_n \cdot P_1 \cdot |H|$$

(1)

where $P_n = ((R_n - 1) \cdot n)$ is the total number of proofs exchanged, $P_1 = \log_2(n)$ + 2 is the size of the proof, $n$ is the smallest power of 2 that is greater than or equal to $n$, and $|H|$ is the size (in bytes) of the hash function used. The hash function used in this work (SHA256) has a hash size (|H|) of 32 bytes. The hash size can be increased (e.g., SHA3-512 and SHA3-1024) to improve security.

**Information diversity**

In this research, robots are in contact only with the raw sensor and action information relative to the tasks they carry out themselves (i.e., input data for the $h_a$ and $h_b$ hashes): Should a robot be subject to attacks (e.g., hacking or physical capture), only this information could get stolen. Therefore, it is important to ensure a uniform distribution of tasks among the robots so that there are no critical robots that accumulate higher amounts of raw or unprotected information. However, in swarm robotics applications, it is difficult to ensure a uniform distribution of tasks among the robots, and it is therefore important to measure to what extent such uniform distribution is achieved. To do so, we use Shannon’s equitability ($I_c$), a measure of evenness that allows us to measure how evenly spread raw information is within the swarm.

Shannon’s equitability ($I_c$) can be calculated by dividing Shannon’s index $I$ by $I_{max}$

$$I_c = \frac{I}{I_{max}}$$

(2)

where Shannon’s index $I = -\sum_{i=1}^{n} p_i \cdot \ln p_i$ [i.e., Shannon’s entropy (41)] is a mathematical measurement used to characterize diversity in a community. In this case, $S$ represents the total number of robots that took part in the mission ($R_n$), $p_i$ is the percentage of tasks robot $i$ conducted compared with the mission’s length ($n$), and $I_{max} = \ln S$. $I_c$ assumes a value between 0 and 1. Lower values indicate more uneven distributions, whereas higher values indicate more uniform distributions. In this case, 1 represents complete evenness: All robots carried out the same number of tasks and therefore were exposed to the same amount of raw information.

**REFERENCES AND NOTES**


M. Swan, Blockchain: Blueprint for a New Economy (O’Reilly Media, 2015).


P. Veena, S. Panikkar, S. Nair, F. Brody, Empowering the Edge: Practical Insights on a Decentralized Internet of Things (IBM Institute for Business Value, 2015).

Funding: This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 751615. M.D. acknowledges support from the Belgian F.R.S.-FNRS, of which he is a research director. Author contributions: E.C.F. designed and realized the system and performed the experiments. E.C.F. and M.D. analyzed the results and wrote the manuscript. T.H., A.S.P., and M.D. acknowledges support from the Belgian F.R.S.-FNRS, of which he is a research director.

Data and materials availability: This paper can be found in the paper or the Supplementary Materials.

Submitted 7 October 2020 Accepted 5 July 2021 Published 28 July 2021 10.1126/scirobots.abf1538

Secure and secret cooperation in robot swarms
Eduardo Castelló Ferrer, Thomas Hardjono, Alex Pentland and Marco Dorigo

Sci. Robotics 6, eabf1538.
DOI: 10.1126/scirobotics.abf1538