Secure and secret cooperation in robot swarms

Eduardo Castelló Ferrer*,1,2,3, Thomas Hardjono2, Alex ‘Sandy’ Pentland1,2, Marco Dorigo3
1MIT Media Lab, 2MIT Connection Science & Engineering
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
3IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
*E-mail: ecstll@mit.edu.

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 co-operative robotic missions in an authenticated data structure known as 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 towards 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.
Summary

Robot swarms exploiting Merkle trees achieve secret and secure cooperation in sequential missions.

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 one of the ten robotics grand challenges for the next 5-10 years that will have substantial socioeconomic impact. Despite having such a promising future, many important aspects which need to be considered in realistic deployments are either underexplored 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 points 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, 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) as well as 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, due to 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
Figure 1: **Towards secure and secret cooperation in swarm robotics missions.** Example of sequential mission (e.g., build a 2-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 the bridge is lowered when it is not) the entire mission can fail.

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 not only lead 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, an attacker might influence the behavior of the whole system (25, 27) and eventually hinder the entire mission. Fig. 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), 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 multi-robot 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). Due to 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. Our approach allows robots in a swarm to cooperate while minimizing security risks due to issues like physical capture or tampered members. We achieve this by creating a secure data object that is shared by the robots (i.e., a Merkle tree) instead of protecting the communication channel among them (e.g., by encrypting the network connections).

A Merkle tree (MT) (37) is a hash-based tree structure that belongs to the family of authenticated data structures, a type of data object that is not compromised even when it is manipulated by an untrusted third party. MTs have two main properties: correctness and security. Correctness implies that a “proof” can be easily generated to verify and demonstrate that a piece of information is known and correct; this can be done without exposing the raw information itself by using cryptographic hashes. Security implies that a computationally bounded, mali-
Figure 2: Encapsulating swarm robotics missions into Merkle trees. (A) Merkle tree (MT) used in this work: a hash-based tree structure where each leaf node stores the hash of a task within the swarm’s mission (i.e., the combination of a robot action hash ($h_a$) and a sensor input hash ($h_s$)), while each interior node contains the hash of the combination of its two children. (B) Mission initialization workflow: (I) The swarm’s operator generates a valid MT where all the tasks that the swarm needs to perform to complete its mission are included in the correct order. (II) The MT is broadcast to all the robots in the swarm. (III) The mission starts. (C) Workflow for prover (P) and verifier (V) robots when they exchange queries (Q) and MT “proofs” ($\pi$) to synchronize and complete their corresponding MT copies.
cious agent cannot forge an incorrect result and therefore only agents that know the appropriate information can generate valid “proofs”.

In this paper, we argue that by using Merkle trees (see Fig. 2(A) and Materials and Methods), swarm operators can provide the “blueprint” of the swarm’s objectives without disclosing raw or unprotected data about the mission itself (Fig. 2(B)). More specifically, we introduce a robotics framework where data verification is separated from data itself. By exchanging cryptographic proofs (i.e., hashes from a MT), robots in the swarm are able to prove to their peers that they know specific pieces of information included in the swarm’s mission and therefore that they are cooperating towards its completion (Fig. 2(C)). 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 to both increase the performance of the system and to diversify the amount of information within the swarm. However, larger numbers of robots as well as 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 Merkle trees is within reach of current robotic technology.

The remainder of this paper is structured as follows. The results section 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 Merkle trees as a cooperation mechanism for swarm robotics systems in both simulation and real-robot experiments. The discussion section analyses the results obtained and proposes future research directions. The materials and methods section describes in detail the concept of Merkle tree and its applications for robot swarms missions. Moreover, this section introduces the concept of Merkle tree “proof” as well
as the analysis metrics used to evaluate the proposed approach. Sections S1 and S2 (Supplementary Materials) describe the setup of our experiments including the arena, robots, and simulator used to evaluate the proposed approach. Sections S3 presents the control logic for the robots in both missions. Finally, S4 presents a futuristic scenario where the research proposed in this paper 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. 3(A). The foraging mission consists of a sequence of tasks to be executed by e-puck robots (41) (Fig. 3(B)). Each task consists in finding an object of a given color and retrieving it to a target location. The environment is a rectangular area divided into a $5 \times 5$ 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. 2(A)) with $n$ tasks before the experiment starts (Fig. 2(B)). Therefore, during the mission, robots do not have explicit information about what is the correct sequence of colors to be delivered since 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. Fig. 3(C) shows the real-robot foraging environment where objects are represented by colored cells projected on the arena by using a projector (see Supplementary Materials for the experimental setup of the real-robot experiments).
Figure 3: **Foraging scenario.** (A) Simulated arena used for the foraging mission. The scenario consists of a rectangular area of $2.5 \times 2.5$ m$^2$ within which robots, objects to be retrieved (represented as colored cells), and a target area ($0.5 \times 0.5$ m$^2$) located at the center are placed (see Supplementary Materials for a detailed description of the simulation experimental setup). A typical simulated run is displayed in movie M1. (B) Diagram of the e-puck robot together with its sensors layout. E-puck’s size (13 cm tall and 7 cm in diameter) and features make it an ideal platform for simulated and real-world swarm robotics experiments. (C) Real-world equivalent of the scenario shown in (A). In this scenario, colored cells are projected on the arena by using a projector (see Supplementary Materials for a detailed description of the real-robot experimental setup). A typical real-world run is displayed in movie M2.
In the foraging mission, robots search the environment looking for objects. Once an object is found, robots identify its color and generate $h_s$. Then, they generate the value $h_a$ for all possible actions (note that in the considered foraging mission, $h_a$ is restricted to the “carry to target” action). $H_i$ is calculated by hashing $h_a$ and $h_s$: $H_i = H(h_a, h_s)$ and verified against the current working task ($i$) of the robot; $i$ is initialized to 0 at the beginning of the mission and is an index that points to the task that the robot is currently trying to fulfill in its local MT. In case it is possible to generate a valid proof (see Materials and methods for details about MT proofs) with the task’s information ($\exists \pi_i, H_i$), the robot “grabs” the object and delivers it at the target location. If not, 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 $i \in (0 \leq i \leq n - 1)$. It is important to note that during the experiments, robots can exchange information with their peers (e.g., $i$, $\pi$). If two robots are within communication distance, they can exchange their $i$ values. If there is a disparity in the values (i.e., one $i$ is lower than the other) 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. 2(C). 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 towards the fulfillment of the swarm’s mission (see Supplementary Materials for robot’s interaction and finite state machine diagrams).

Sequentiality is preserved since a robot with proof for task $k$, in order to convince a robot who is searching for task $i$ to switch to task $k$, should provide also 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 $h_s$ values of previous tasks to the current working leaf ($H_i = H((h_{a,i}, h_{s,i}, h_{a,i-1}, h_{s,i-1}))$). 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 (i.e., the total amount of data robots exchanged during the mission), and information diversity (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 the Materials and Methods section.

Fig. 4(A) shows the finishing time \( F_t \) and its standard deviation for several MT length configurations \( (n) \) and robot swarm sizes \( (R_n) \) for both simulation and real-robot experiments. According to Fig. 4(A), \( F_t \) 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. Fig. 4(B) shows how the probability of success \( (P_s) \) curves change when changing the number \( n \) of tasks for all simulation configurations presented in Fig. 4(A). One can also see that adding robots increases \( P_s \) since 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-E) and real-robot experiments Fig. 4(F-H). It is important to note that \( P_s \) does reach 1 (the maximum value) in all configurations as all experiments terminated with success before reaching the maximum amount of allowed experiment time also known as Time Cap (TC).
Figure 4: **Finishing times and probability of success for the foraging mission.** (A) Average $F_t$ (in seconds) and its standard deviations for different MT lengths ($n$) and robot swarm sizes ($R_n$) for both simulation (bars) and real-robot experiments (circles). (B) Probability of success ($P_s$) for all simulation configurations shown in (A) for the execution of the foraging mission with $R_n$ in the interval $1 \leq R_n \leq 16$. For each of the $R_n$ values a lower-bound (solid line representing $n = 2$), mean of the evaluated interval (dot-dashed line), and the upper-bound (loosely dashed line representing $n = 8$) were included. (C-E) Detailed plots for the simulation experiments. (F-H) Detailed plots for the real-robot experiments. In the graphs is shown the probability of success $P_s$ when the number of tasks is $n = [2, 4, 8]$ and the number of robots is $R_n = 16$ (C & F), $R_n = 4$ (D & G), and $R_n = 2$ (F & G).
Figure 5: **Amount of communication (AC) and information diversity (I_e) metrics for the foraging mission.** The AC and I_e metrics are depicted in blue and red colors, respectively, for different R_n and n values used in (A) simulation and (B) real-robot experiments. The upper-bound limit for the I_e metric is depicted in both figures with a dashed grey line. Averaged results and standard deviations (not visible in (B) because of their very low value) both in simulation and with real robots suggest a direct relationship between the two metrics in the foraging mission.
Fig. 5 shows average results and standard deviations for the amount of communication (AC) in KB and information diversity measured by Shannon’s equitability index (Ie) for different simulation and real-robot experiments. In particular, Fig. 5(A) reports simulation results for the Rn (1 ≤ Rn ≤ 16) and n ∈ {2, 4, 6, 8} configurations, while Fig. 5(B) reports results of real robot experiments for Rn = [2, 4, 8, 16] and n ∈ {2, 4, 8} configurations. In both figures, Ie ∈ [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) while higher values indicate more uniform distributions (e.g., all robots completing the same number of tasks). Figs. 5(A-B) also show that AC increases linearly with Rn since there are more robots exchanging MT proofs. Moreover, MTs with larger n values make the proofs robots exchange “heavier” (i.e., π is longer), which also contributes to increase AC. However, larger Rn values tend to increase information evenness (Ie) in the swarm. In other words, the execution of tasks is more evenly distributed among robots. Results for both simulation and real-robot experiments suggests that, as we increase Rn, the distribution of tasks fulfilled by the robots tends to become more uniform. Fig. 5 also suggests that information diversity might need a really large Rn 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-C)) where we conducted 100 simulations and 15 real-robot experiments for each considered configuration. Fig. 6(A) shows the “blueprint” of a 5 × 5 maze where 0 represents an empty space, 1 a wall, and * and @ 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 × 0.5
Figure 6: **Maze formation scenario.** (A) $5 \times 5$ matrix used to represent a maze. Four different elements are included in the array: 0 (black) represents an empty space, 1 (red) represents a wall, @ and * (blue) represent the entrance and the exit of the maze, respectively. (B) The maze depicted in (A) built by a swarm of simulated robots (a typical simulation run is displayed in movie M3) where the path between entrance and exit is depicted with a dashed line. (C) The maze depicted in (B) built by a swarm of real e-puck robots (a typical real-robot run is displayed in movie M4). (D) Average $F_t$ (in seconds) and its standard deviation for the maze formation mission with different robot swarm sizes for both simulation (blue) and real-robot (orange) experiments. All results were obtained with a 16 tasks ($n = 16$) maze as shown in (A-C). (E) Amount of communication ($AC$) and information diversity ($I_e$) metrics in blue and red colors, respectively, for both simulation (circles) and real robots (squares) and for different $R_n$ values for the maze formation mission. Upper-bound limit values for $AC$ and $I_e$ are depicted with dotted and dashed black lines respectively. Average results suggest no relationship between the two metrics.
m^2) 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 it to generate \( h_a \). Then, it generates the value \( h_a \) for all possible actions (note that in the considered maze formation mission, \( h_a \) 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 (\( \exists \pi(i, H_i) \)) the robot finds the center of the cell and stops there (Fig. 6(B-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 in order to match the number of cells where the value 1 is present in Fig. 6(A) and the number \( R_n \) of robots was varied in the range \( 16 \leq R_n \leq 28 \).

Fig. 6(D) shows average \( F_t \) and standard deviations for the maze formation mission for both simulation and real-robot experiments: as it was the case in the corresponding foraging experiment (Fig. 4(A)), 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. Fig. 6(E) shows the \( AC \) and \( I_e \) metrics for the maze formation mission. Fig. 6(E) follows the same pattern as Fig. 5: \( AC \) increases linearly with \( R_n \). Fig. 6(E) includes the \( AC \) upper limit following equation 1 (see Statistical methods). According to Fig. 6(E), both simulation and real-robot \( AC \) fall under the limit and suggest that real-robot experiments require a lower amount of communication than their simulation counterparts. Finally, Fig. 6(E) shows that in this scenario complete “evenness” of information (i.e., \( I_e = 1 \)) is achieved.
Experiments with larger-dimension missions

Figure 7: **LEGO® models with their corresponding piece count.** LEGO® models are a good example of complex sequential missions that could be encapsulated in Merkle trees.

Table 1: Memory, maximum amount of communication per robot ($AC_{ul}/R_n$), and computation times for the models depicted in Fig. 7. Time measures (in seconds) are averages (standard deviations in brackets) over 100 runs using the Gumstix Overo board, the on-board computer mounted on the e-puck robots.

| Model in Fig. 7 | Memory (KB) | $AC_{ul}/R_n$ (MB) | Average computation times* (s) |
|-----------------|-------------|--------------------|--------------------------------|
| (A) Millennium Falcon (7541 pieces) | 235 KB | 3.39 MB | G: 0.35 (0.001)  
|                  |            |                    | P: 0.0017 (0.0002)  
|                  |            |                    | V: 0.002 (7.96 $10^{-5}$) |
| (B) Taj Mahal (5923 pieces) | 185 KB | 2.54 MB | G: 0.28 (0.004)  
|                  |            |                    | P: 0.0016 (7.40 $10^{-5}$)  
|                  |            |                    | V: 0.002 (0.0003) |
| (C) Bugatti Chiron (3599 pieces) | 112 KB | 1.46 MB | G: 0.16 (0.001)  
|                  |            |                    | P: 0.0015 (3.61 $10^{-5}$)  
|                  |            |                    | V: 0.0018 (9.37 $10^{-5}$) |

* G: time to generate the complete Merkle tree; P: time for the generation of a proof; V: time for the verification of a proof.

Encouraged by these results, we found appropriate to analyze the feasibility of our approach in complex missions where the number of tasks takes relatively large values. Fig. 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 since \( n \) takes a relatively large value. Due to the possibility of accurately calculating the maximum amount of communication (\( AC_{ul} \)) required to make all the robots complete their MTs (see Statistical methods, Eq. 1) as well as the overall size of the MT stored by the robots, we can compute Fig. 7’s corresponding MTs and measure their memory and maximum amount of communication per robot (\( AC_{ul}/R_n \)) requirements. We also computed the average computation time to generate the complete Merkle tree (G), 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 towards secure and secret cooperation in robot swarms. Regarding the security aspect, by using this approach, in order 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 (34)), 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 since any change in the task’s data (i.e., \( s, a \)) 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 since 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 size have lower finishing time \((F_t)\) and a higher probability of success \((P_s)\). Results also show that the amount of communication \((AC)\) grows linearly with the swarm size \((R_n)\), which in extreme situations (e.g., very large swarms) could be detrimental to the system since 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 \((I_e)\) since 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 \(F_t\). In the maze formation mission, \(I_e\) is maximized (i.e., \(I_e = 1\)), which is possible since 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, in contrast to the foraging scenario, where one robot might be able to complete several tasks. Even though \(AC\) grows linearly with \(R_n\), this still does not represent a challenging situation for the swarm (e.g., 72 and 70 KB are exchanged in a 28 robots system in simulation and real-robot experiments respectively). Counterintuitively, real-robot experiments require a lower amount of communication \(AC\) 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, while other robots in the swarm are still trying to synchronize on previous completed tasks. Finally, it is interesting to emphasize that, due to 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.

Finally, 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. It is important to note that 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, while 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. It is important to note that, 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 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 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 $l$: $H_i = H((h_a, h_s, h_l))$. 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, public) will co-exist in a same location (e.g., urban and disaster areas, industrial environments, battlefield) without exposing sensitive data about their internal processes, goals, and organizations. The findings presented in this paper shed light on the design and implementation of sequential missions in potential applications for distributed robotics such as manufacturing (e.g., collaborative assembly (42)), logistics (e.g., supply chain in warehouses (43)); and security (e.g., surveillance of critical infrastructure (44)).

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, collision avoidance) might not be suitable for the method proposed. This is because robots need to discover the correct combination of $h_a$ and $h_s$ in order to execute a task, which can make impossible to guarantee that robots complete a task within a fixed amount of time. Finally, 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 of this work 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 as well as multimedia material when the work is completed by the robot swarm.

Materials and Methods

Merkle trees for robot swarms

Figure 8: Description of a typical Merkle tree structure and proof. (A) Merkle tree implementation \((d = 3)\) with 4 leafs, 2 interior, and 1 root nodes. Each leaf node (green) encapsulates the hash of two hashes: a robot action \((h_a)\) and a robot sensor input \((h_s)\). Each leaf node represents one of a series of tasks that robots should complete sequentially \((i = 0, i = 1, \ldots, i = m - 1)\) in order 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) in order to get the proof for leaf node \(i = 1\).

A Merkle tree (MT) \((37)\) is a hash-based structure where data is not stored in the inner nodes but in the leaves (Fig. 8(A))). 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. 8(A)). 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 \( H: H(h_a, h_s) \).

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 it can perform (e.g., “push”); fourth, it combines \( h_a \) and \( h_s \) and rehashes them to obtain the hash string \( H(h_a, h_s) \); fifth, it matches \( H(h_a, h_s) \) against 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” \((\pi)\) that demonstrates 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 \( h_s \)).

Fig. 8(B)) shows the proof \( \pi \) for a fetch at task \( i = 1 \). It consists of four elements in sequence, the two hashes \( h_a \) and \( h_s \), the hash \( H_4 \), and the hash \( H_3 \). Then, the verification proceeds bottom up: verifier computes the hashes of the two hashes \( h_a \) and \( h_s \), 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. Finally, 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 \( h_s \) 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. It is important to note that 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. In this paper, we use the mission’s finishing time $F_t$ to measure the amount of time required to fulfill the swarm’s mission. In addition, we use an estimate $P_s$ of the probability that the system attains its target objective in an amount of time $\tau$. Formally, let $j \in \{1, \ldots, k\}$ be the index of an experiment, $F_{t,j}$ be the finishing time of experiment $j$, and experiments where $F_{t,j} < TC$ be considered successful experiments, where TC is the maximum amount of allowed experiment time. The estimate $P_s$ of the probability of success of the system over time (up to TC) is defined as $P_s(\tau \leq t) = \{j | F_{t,j} \leq t\}/k$.

**Amount of communication.** The amount of communication $AC$ is computed by multiplying the number of times the prover-verifier workflow (Fig. 2(C)) 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 $h_a, h_s$ hashes. In missions with $n$ tasks where all robots need to complete their MTs, an upper limit $AC_{ul}$ of the $AC$ metric can be calculated with the following equation:

$$AC_{ul} = P_n \cdot P_l \cdot |H|$$

where $P_n = ((R_n - 1) \cdot n)$ is the total number of proofs exchanged, $P_l = \log_2(\hat{n}) + 2$ is the size of the proof, $\hat{n}$ is the smallest power of 2 that is greater 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. It is important to note that the hash size can be increased (e.g., SHA3-512,
SHA3-1024) in order to achieve improved 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 $h_s$ 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 ($I_e$), a measure of “evenness” that allows us to measure how evenly spread raw information is within the swarm.

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

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

(2)

where Shannon’s index $I = -\sum_{i=1}^{S} p_i \ln p_i$ (i.e., Shannon’s entropy (39)) 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 to the mission’s length ($n$), and $I_{max} = \ln S$. $I_e$ assumes a value between 0 and 1. Lower values indicate more uneven distributions while 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**

1. M. Dorigo, M. Birattari, M. Brambilla, Swarm Robotics, *Scholarpedia*, 1463, (2014).
2. R. D’Andrea, Guest Editorial: A Revolution in the Warehouse: A Retrospective on Kiva Systems and the Grand Challenges Ahead, *IEEE Transactions on Automation Science and Engineering*, 638 (2012).

3. T. Blender, T. Buchner, B. Fernandez, B. Pichlmaier, C. Schlegel, Managing a Mobile Agricultural Robot Swarm for a seeding task, *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 6879–6886 (2016).

4. D. Albani, J. IJsselmuiden, R. Haken, V. Trianni, Monitoring and mapping with robot swarms for agricultural applications, *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1–6 (2017).

5. Y. Chen, Z. Du, M. García-Acosta, Robot As a Service in Cloud Computing, *Proceedings of the 2010 Fifth IEEE International Symposium on Service Oriented System Engineering*, SOSE ’10 (IEEE Computer Society, Washington, DC, USA, 2010), 151–158 (2010).

6. H. He, S. Kamburugamuve, G. Fox, W. Zhao, Cloud based Real-time Multi-robot Collision Avoidance for Swarm Robotics, *International Journal of Grid and Distributed Computing* 9, 339-358 (2016).

7. M. H. A. Majid, M. R. Arshad, Design of an autonomous surface vehicle (ASV) for swarming application, *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, 230–235 (2016).

8. Y. Mulgaonkar, A. Makineni, L. Guerrero-Bonilla, V. Kumar, Robust Aerial Robot Swarms Without Collision Avoidance, *IEEE Robotics and Automation Letters* 3, 596–603 (2018).

9. D. Rus, M. T. Tolley, Design, fabrication and control of origami robots, *Nature Reviews Materials* 3, 101–112 (2018).
10. E. Castello Ferrer, S. Y. Hanay, Demo: A Low-cost, Highly Customizable Robotic Platform for Testing Mobile Sensor Networks, *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc ’15 (ACM, New York, NY, USA, 2015), 411–412 (2015).

11. N. Oxman, J. Duro-Royo, S. Keating, B. Peters, E. Tsai, Towards Robotic Swarm Printing, Archit Design, 84: 108-115 (2014).

12. D. Chaudhari, S. A. Bhosale, A review on management of logistics using swarm robotics, *2016 International Conference on Communication and Signal Processing (ICCSP)*, 0371–0373 (2016).

13. A. Heinla, A. Martinson, K. Volkov, A. Macks, L. Roberts, I. Mandre, M. Liivik, T. Liivik, I. Liivik, Method and system for autonomous or semi-autonomous delivery, U.S. Patent 0232839A1, April, 9, 2018.

14. A. L. Alfeo, E. Castello Ferrer, Y. Lizarribar Carrillo, A. Grignard, L. Alonso Pastor, D. T. Sleeper, M. G.C.A. Cimino, B. Lepri, G. Vaglini, K. Larson, M. Dorigo, A. Pentland, Urban Swarms: A new approach for autonomous waste management, *2019 International Conference on Robotics and Automation (ICRA)*, 4233–4240 (2019).

15. 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, Science Robotics, 3 (2018).

16. P. Scharre, Robotics on the Battlefield Part II: The Coming Swarm, *Center for New American Security*, 1–4 (2014).
17. S. Vivek, D. Yanni, P. J. Yunker, J. L. Silverberg, Cyberphysical risks of hacked internet-connected vehicles, *Phys. Rev. E*, **100**, 012316 (2019).

18. R. N. Akram, K. Markantonakis, K. Mayes, O. Habachi, D. Sauveron, A. Steyven, S. Chaumette, Security, privacy and safety evaluation of dynamic and static fleets of drones, *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, 1–12 (2017).

19. S. Gil, S. Kumar, M. Mazumder, D. Katabi, D. Rus, Guaranteeing spoof-resilient multi-robot networks, *Autonomous Robots* **41**, 1383–1400 (2017).

20. A. Prorok, V. Kumar, A macroscopic privacy model for heterogeneous robot swarms, *ANTS 2016. Lecture Notes in Computer Science* **9882**, 15–27 (2016).

21. A. Prorok, V. Kumar, Towards Differentially Private Aggregation of Heterogeneous Robots, *Distributed Autonomous Robotic Systems, Springer Proceedings in Advanced Robotics* **6**, 587–601 (2018).

22. A. L. Christensen, R. O’Grady, M. Birattari, M. Dorigo, Exogenous Fault Detection in a Collective Robotic Task, *Advances in Artificial Life, Proceedings of ECAL 2007* **4648**, 555–564 (2007).

23. A. L. Christensen, R. O’Grady, M. Dorigo, Synchronization and Fault Detection in Autonomous Robots, *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2008)*, 4139–4140 (2008).

24. S. Nouyan, R. Groß, M. Bonani, F. Mondada, M. Dorigo, Teamwork in Self-Organized Robot Colonies, *IEEE Transactions on Evolutionary Computation* **13**, 695–711 (2009).
25. A. G. Millard, J. Timmis, A. F. T. Winfield, Towards Exogenous Fault Detection in Swarm Robotic Systems, *Towards Autonomous Robotic Systems. TAROS 2013. Lecture Notes in Computer Science*, 429–430 (2013).

26. J. D. Bjerknes, A. F. T. Winfield, On Fault Tolerance and Scalability of Swarm Robotic Systems, *Distributed Autonomous Robotic Systems: The 10th International Symposium* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013), 431–444 (2013).

27. V. Strobel, E. Castello Ferrer, M. Dorigo, Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 541–549 (2018).

28. V. Strobel, E. Castello Ferrer, M. Dorigo, Blockchain Technology Secures Robot Swarms: A Comparison of Consensus Protocols and Their Resilience to Byzantine Robots, *Frontiers in Robotics and AI* 7, 54 (2020).

29. A. F. Winfield, J. Nembrini, Safety in numbers: fault-tolerance in robot swarms, *International Journal of Modelling, Identification and Control* 1, 30–37 (2006).

30. F. Higgins, A. Tomlinson, K. M. Martin, Survey on Security Challenges for Swarm Robotics, *IEEE 2009 Fifth International Conference on Autonomic and Autonomous Systems*, 307–312 (2009).

31. I. Sargeant, A. Tomlinson, Review of Potential Attacks on Robotic Swarms, *Proceedings of 2016 SAI Intelligent Systems Conference (IntelliSys). Lecture Notes in Networks and Systems* 16, 628–646 (2018).

32. M. Dorigo, D. Floreano, L.M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A.L. Christensen, A. Decug-
niere, G. Di Caro, F. Ducatelle, E. Ferrante, A. Forster, J. Martinez Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O’ Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, T. Stutzle, V. Trianni, E. Tuci, A.E. Turgut, F. Vaussard, Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms, *IEEE Robotics Automation Magazine* **20**, 60–71 (2013).

33. J. Werfel, K. Petersen, R. Nagpal, Designing Collective Behavior in a Termite-Inspired Robot Construction Team, *Science* **343**, 754–758 (2014).

34. S. Halevi, D. Harnik, B. Pinkas, A. Shulman-Peleg, Proofs of Ownership in Remote Storage Systems. *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 491–500 (2011).

35. E. C. Ferrer, O. Rudovic, T. Hardjono, A. Pentland, Robochain: A secure data-sharing framework for human-robot interaction. *arXiv preprint arXiv:1802.04480* (2018).

36. V. Strobel, M. Dorigo, Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation, *Swarm Intelligence – Proceedings of ANTS 2018 – Eleventh International Conference* **11172**, 425–426 (2018).

37. R. C. Merkle, A Digital Signature Based on a Conventional Encryption Function, *Advances in Cryptology — CRYPTO ’87*, 369–378 (1988).

38. L. Garattoni, M. Birattari, Autonomous task sequencing in a robot swarm, *Science Robotics* **3**, (2018).

39. C. E. Shannon, A Mathematical Theory of Communication, *Bell System Technical Journal* **27**, 379—423 (1948).
40. C. Pinciroli, V. Trianni, R. O’Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L.C. Gambardella, M. Dorigo, ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems, *Swarm Intelligence* **6**, 271–295 (2012).

41. 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, *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* **1**, 59–65 (2009).

42. J.A. Marvel, R. Bostelman, J. Falco, Multi-Robot Assembly Strategies and Metrics, *ACM Comput. Surv.* **51**, 1-32, (2018).

43. A. Farinelli, N. Boscolo, E. Zanotto, E. Pagello, Advanced approaches for multi-robot coordination in logistic scenarios, *Robotics and Autonomous Systems* **90**, 34–44 (2017).

44. I. Roman-Ballesteros and C. F. Pfeiffer, A Framework for Cooperative Multi-Robot Surveillance Tasks, *Electronics, Robotics and Automotive Mechanics Conference (CERMA’06)* **2**, 163–170 (2006).

45. A. Stranieri, A.E. Turgut, M. Salvaro, G. Francesca, A. Reina, M. Dorigo, and M. Birattari, IRIDIA’s Arena Tracking System, (Tech. rep. TR/IRIDIA/2013-013, IRIDIA, Université libre de Bruxelles, 2013), 1–7 (2013).

46. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, B. Wheeler, A. Ng, ROS: an open-source Robot Operating System, *ICRA Workshop on Open Source Software* **3**, (2009).

47. A. Tiderko, F. Hoeller, T. Röhling, The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems, *Robot Operating System (ROS): The Complete Reference* **1**, 629–650, (2016).
48. A. Drogoul, E. Amouroux, P. Caillou, B. Gaudou, A. Grignard, N. Marilleau, P. Taillandier, M. Vavasseur, D.A. Vo, J.D. Zucker, GAMA: A Spatially Explicit, Multi-level, Agent-Based Modeling and Simulation Platform, *Advances on Practical Applications of Agents and Multi-Agent Systems* 1, 271–274 (2013).

49. A. D. Maynard, Navigating the fourth industrial revolution, *Nature nanotechnology* 10, 1005–1006 (2015).

50. W. E. Forum, The future of jobs: Employment, skills and workforce strategy for the fourth industrial revolution. *Global Challenge Insight Report, World Economic Forum*, 1–167 (2016).

51. K. Schwab, The fourth industrial revolution, *World Economic Forum*, (2017).

52. E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, Industrial internet of things: Challenges, opportunities, and directions, *IEEE Transactions on Industrial Informatics* 14, 4724–4734 (2018).

53. S. Park, N. Crespi, H. Park, S.-H. Kim, Iot routing architecture with autonomous systems of things, *2014 IEEE World Forum on Internet of Things (WF-IoT)* 1, 442–445 (2014).

54. G. A. Bekey, Autonomous robots: from biological inspiration to implementation and control, *MIT press*, (2005).

55. F. Fahimi, Autonomous robots: Modeling, Path Planning, and Control, *Springer*, (2009).

56. J. Lee, B. Bagheri, and H.-A. Kao, A cyber-physical systems architecture for industry 4.0-based manufacturing systems, *Manufacturing Letters* 3, 18–23 (2015).

57. Y.-Y. Hsieh, The rise of decentralized autonomous organizations: Coordination and growth within cryptocurrencies, *Ph.D. dissertation, The University of Western Ontario*, (2018).
58. M. Swan, Blockchain: Blueprint for a new economy, *O'Reilly Media* (2015).

59. M. Swan, Blockchain thinking: The brain as a Decentralized Autonomous Corporation [Commentary], *IEEE Technology and Society Magazine* **34**, 41–52 (2015).

60. V. Buterin, A next-generation smart contract and decentralized application platform, *White Paper*, 1–36 (2014).

61. I. Lianos, P. Hacker, S. Eich, G. Dimitropoulos, Regulating Blockchain: Techno-Social and Legal Challenges, *Oxford University Press*, (2019).

62. P. Veena, S. Panikkar, S. Nair, P. Brody, Empowering the edge: practical insights on a decentralized internet of things, *IBM Institute for Business Value*, (2015).
Acknowledgments

Acknowledgments: 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. Marco Dorigo 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 as well as performed the experiments. E.C.F. and M.D. analyzed the results and wrote the manuscript. T.H., A.P., and M.D. directed the project. Competing interests: The authors declare that they have no competing interests. Data and materials availability: All data generated and discussed in the article and in the Supplementary Materials is available at [www.blockchainswarm.eu/experimental_data](http://www.blockchainswarm.eu/experimental_data).

Supplementary materials

Section S1. Detailed description of the real-robot experimental setup  
Section S2. Detailed description of the simulation experimental setup  
Section S3. Description of the finite state machines controlling the robots  
Section S4. IRIDIA Swarm Marketplace  
Figure S1. Experimental setup for the real-robot experiments  
Figure S2. Robot interaction diagram  
Figure S3. Robot Finite State Machine (FSM) diagrams  
Figure S4. IRIDIA Swarm Marketplace  
Movie M1. Typical simulation run for the foraging experiments  
Movie M2. Typical real-robot run for the foraging experiments  
Movie M3. Typical simulation run for the maze formation experiments  
Movie M4. Typical real-robot run for the maze formation experiments
Movie M5. Tangible Swarm (Supporting features)

References (45-62)