Instant Friend-or-Foe Identification for Stealth Devices in Coalition-Drone Swarms

Christina Lassfolk[1] and Hannu Kari[2]
[1]Aalto University, Espoo, Finland
[2]The Finnish National Defence University, Helsinki, Finland

christina.lassfolk@aalto.fi
hannu.kari@mil.fi

Abstract: The war in Ukraine demonstrates the versatility of drones in modern warfare. However, only the initial steps of this technological disruption are visible. The evolution will bring a shift from individually operated drones to swarms of drones. These swarms will operate semi- or fully autonomously, diminishing the role of human operators. Instead of real time operations, humans will set mission objectives and supervise operations. The complexity increases further as drones from diverse origins, with different capabilities, and with different levels of trust, collaborate in joint missions. This poses a challenging research question of how to identify, quickly and securely, other devices in a coalition mission featuring numerous autonomously operating units. This article introduces a novel mechanism for securely identifying autonomously operating drones on the battlefield without prior communication. Pre-deployment configuration enables autonomous decision-making in missions, eliminating the need to consult third parties during the identification process. This research focuses on a scenario where an ongoing mission that has suffered from equipment depletion necessitates replacement with new equipment. The present paper demonstrates how a valuable device can securely identify its neighbors before revealing its existence. A practical example of the benefits is the ability to conceal the precise location of the valuable device by utilizing low-cost, expendable civilian drones as message repeaters. The primary contribution of this paper is a solution that allows a device to operate in a stealthy mode and to distinguish friends and foes instantly and securely without prior encounters. The proposed concept of secure identification facilitates trust management in coalition drone swarms operating on the battlefield. Transparent use of data encryption is possible, but it is beyond the scope of this paper. Beyond flying drones, this solution is applicable to any autonomous or semi-autonomous system across various domains, as well as to human-carried devices that benefit from stealthy operation.

Keywords: Drones, Swarms, Trust, Identification, Coalition, Friend-or-foe identification

1. Introduction

Drones serve in a multitude of military tasks such as surveillance, reconnaissance, or combat action. They also have numerous civilian applications, such as law enforcement, firefighting, search and rescue operations, surveillance for industry and agriculture, communication or broadcasting (Lehto and Hutchinson, 2020; Hoang *et al.*, 2023). The extensive use of drones is a significant feature of the war in Ukraine. Currently, human operators control individual drones but they will not be able to manage future swarms without relying on a high degree of autonomy. Thus, the anticipated paradigm shift from individually controlled drones to drone swarms necessitates autonomy. The key advantage of drones operating in swarms lies in the diverse capabilities of the individual members, which collectively form a team far superior to any single drone. Moreover, the sheer number of drones within a swarm makes it challenging to eliminate the swarm as a whole. Swarms offer increased flexibility, survivability, and configurability compared to monolithic systems while requiring tight cooperation between the members (Javed *et al.*, 2024). This synergy constitutes the overwhelming power of swarms on the battlefield (Lehto and Hutchinson, 2020). Utilizing unmanned drones reduces the risk to human operators, allowing them to focus on giving swarms mission objectives such as “patrol the area and destroy the targets you find” rather than controlling individual drones. The versatility of hybrid drone swarms in future combat is evident.

It is crucial to specify how hybrid drone swarms consisting of drones from different coalition partners can cooperate securely to achieve common combat objectives. In addition to military grade equipment, commercial civilian drones can increase the volume of devices on the battlefield, to perform supportive tasks and to save costs (i.e., by concealing valuable military devices). Massive drone swarm deployment leads to a significant drone turnover. Onboarding and offboarding devices on the battlefield must be frictionless and rapid to enable quick swarm resupplies. This can be done with advance preparation (i.e., pre-deployment configuration) as proposed in (Lassfolk and Kari, 2025). Since the drones autonomously identify each other as friend or foe, they can collectively form and operate as a cohesive swarm. This friend-or-foe detection resembles the idea of air traffic control called Identification of Friend or Foe (IFF) (GlobalSecurity.org, 2011), but with different implementation and cryptographic approaches. In this paper, friend-or-foe identification information is included in every message of the packet based communication and it is illustrated using the IP-protocol header extension

principle (Deering and Hinden, 2017). The core idea presented in this paper is to demonstrate a concrete combat use case in which a valuable device instantly and securely can identify in its neighborhood a trustworthy low-cost device, which can provide it with hazardous services. The valuable node can do this without own radio transmission and prior encounters with its neighbors.

This article consists of three separate sections. First, it outlines the scenario and use case. Second, it provides a verbal description and pseudocode snippets to explain the main operating principles of the proposed solution. Third, it provides an analysis and discussion of the solution, followed by indications of future research needs.

2. Scenario

This study builds on a simplified military coalition scenario (Lassfolk and Kari, 2025) composed of four main phases (Figure 1). In the first phase (preparation), the country takes preparatory actions to establish its defense. In the second phase (invasion), the country defends itself against a foreign attacker using its full arsenal. In the third phase (coalition support), the country receives support from its allies. In the fourth phase (calming down), the attacker has been driven back, and the equipment returns to depots.

In the fury of battle, the commissioning of new devices is intensive, and device depletion is continuous. As the battle prolongs, there is a mix of recently deployed devices alongside those used since before the invasion. Especially during the third phase (coalition support), equipment from various countries and manufacturers collaborates to form hybrid drone swarms. Preparatory actions, taken during the first phase, address interoperability issues in the second and third phases.

Figure 1: Scenario phases

The defense battle in the second phase begins with the country's own resources. However, from the third phase onwards, coalition partner equipment is also involved. In all phases, civilian equipment may supplement military grade equipment. This study focuses on activities during the third phase (coalition support), which is the most challenging one. No single organization has full control over all equipment on the battlefield, as devices from various countries and civilian equipment cooperate to achieve common objectives. The prerequisite for the activities described herein is that the equipment has received settings and credentials during the preparation phase. Afterward, the equipment can self-organize and make autonomous decisions.

This paper examines a specific scenario where a valuable device uses a low-cost device as a proxy for services. Before the valuable device reveals its existence through radio transmission, it must reliably identify its neighbors and verify that they belong to the same coalition. Therefore, it listens to the communication of the neighboring devices. The valuable device is not interested in the actual potentially encrypted data content of these messages, but in the additional information attached onto each message. This additional information allows it to determine reliably the trustworthiness of its neighbors. When convinced of the trustworthiness of another node, it may reveal its own existence by sending a message to another node to request a favor from that node.

Figure 2 illustrates the target scenario, where devices *A* and *B* are communicating while *C* observes their communication. The assumption is that all three devices (*A*, *B*, and *C*) belong to the same coalition but not necessarily to the same organization. Additionally, *A* is a low-cost device offering proxy services to coalition members and *C* is a valuable device needing such proxy services. As *C* aims to conceal its existence and location as long as possible, it refrains from transmitting anything until it has securely identified the proxy (*A*). Hence, device *C* listens to the communication between *A* and *B*, checks *A*'s credentials included in the messages, and verifies that *A* belongs to the same coalition. This enables *C* to decide to use *A* as its proxy and minimize exposure of its own actual location. Device *A* can serve as a repeater, acting as a transmission relay and location disguise for the valuable device. However, the identification procedure does not mandate *C* to use the services of *A*. It is simply the first filtering step that helps *C* choose the most advantageous, trustworthy neighbor in its vicinity as a proxy. The practical use cases may vary, but the common factor is that devices meeting for the first time quickly and securely can identify each other. The suggested approach permits integration into routine communication without the need for extra messaging. Alternatively, devices can broadcast their service offerings at regular intervals.

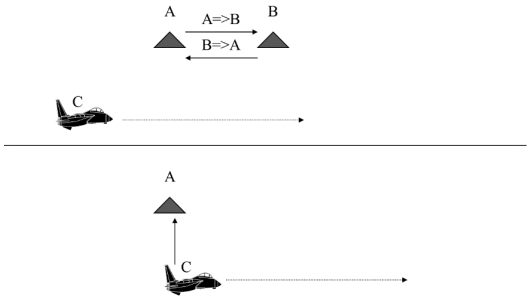

Figure 2: Use case with devices *A*, *B*, and *C*

Low-cost devices use the same identification mechanism to identify their neighbors. Nevertheless, they can be less selective about whom they communicate with, whom they serve, and whether or not they reveal their existence. They may sacrifice their energy resources and location privacy to fulfill the request for the sake of the mission, as they constitute expendable units.

The solution presented relies on three assumptions: radio neutrality, application transparency, and reasonable clock synchronization. Any existing radio technology capable of transmitting information in feasibly sized packets is applicable to the proposed approach. The authors do not advocate frequency hopping or any other protection mechanism of the radio communication. While frequency hopping can serve to filter out unwanted devices and obscure radio transmissions from adversaries, a certificate-based cryptographic approach offers a more robust method for identification. Frequency hopping allows a device with knowledge of the shared hopping sequence to affirm its membership within a group. However, this method is only effective as long as the hopping sequence remains secure and undisclosed to adversaries. Furthermore, frequency hopping lacks the capability of uniquely identifying individual devices—a critical advantage offered by the certificate-based solution presented in this study. The solution does not impose specific requirements or limitations on applications used within the system. Each operating environment sets its own specific accuracy requirements for clock synchronization to detect unauthorized message replays.

3. Information Flow

In the presented solution expressions of trust from partners replace a traditional authentication handshake mechanism (Kaufman, C. ; Hoffman, P.; Nir, Y.; Eronen, P.; Kivinen, 2014; Moskowitz *et al.*, 2015; Rescorla, 2018). Trust is expressed in the form of certificates (Ylonen *et al.*, 1999; Kortesniemi, 2015; ITU-T, 2019). Each device relies on its certificate authority (TTP, Trusted Third Party) from which it has received as pre-configuration data certificates that its TTP has expressed trust in. On the battlefield, devices exchange messages containing certificates, which prove their trustworthiness while digital signatures (such as (NIST, 2023))protect the integrity and timeliness of the message. This novel approach enables verification even though there is no shared secret between the sender and the receiver nor prior communication between them. Thus, the certificate-based solution and digital signatures together with the list of trusted TTPs stored in each device empower the stealthy feature.

The following description assumes the existence of an abstract messaging protocol resembling the IP-protocol whereby it can send messages over a communications channel in packet form. Traditionally such a packet contains a header stating the sender and the receiver followed by the data (Figure 3). In the proposed solution, there is an additional header located between the standard header and the data field. This so-called extra header contains the additional information needed for this specific use case. Only fields relevant to illustrate the proposed solution are included.

Adding such an extra header is standard practice in many messaging protocols (Kent, 2005; Johnson, Arkko and Perkins, 2011; Deering and Hinden, 2017). The structure depicted in Figure 3 illustrates the piggybacked information that allows the sender to prove its identity to the receiver. Based on this information the receiver can deduce whether the sender is a friend or a foe. This communication description is independent of the protocols at the upper and lower layers of the stack. Thus, the proposed solution is compatible with various applications and different transmission standards. It is logical to assume that the suggested mechanism will become a standard option in future protocol suites, similar to how Internet protocol extensions have evolved.

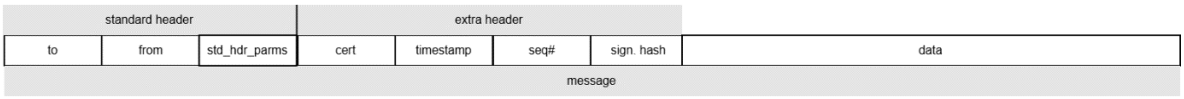

Figure 3: Packet structure

The next description sends and receives packets consisting of a header, an extra header, and data. Figure 4 presents the information structures and fields from Figure 3 in pseudocode (Ulate-Caballero, Berrocal-Rojas and Hidalgo-Céspedes, 2021) (Bennett, 2015; Tarsini and Anggraeni, 2024). In the pseudocode snippets, the "#" symbol indicates the start of a comment whereas angle brackets "<>" indicate explanatory information of data stored in each field. Table 1 collects generic functions and global parameters, which appear in the pseudocode. Figure 5 illustrates settings stored in device *A*, while devices *B* and *C* have similar settings. The devices have received these settings (e.g., coalition membership) during the onboarding procedure at the depot. However later updates are possible.

```
Struct cert {issuer,            <Short identity of the issuer (TTP)>
             subject,           <Public Key of the subject, i.e., the device's identity>
             validity,          <Validity time: from & to>
             property,          <Properties of the subject>
             issuer_signature   <Digital signature of issuer>}
```
a)

```
Struct message {standard_header {
                  to,                 <Destination address>
                  from,               <Sender address>
                  std_hdr_parms,      <Parameters of the standard header of the message including indication of next header "extra
                                       header">},
               extra_header {
                  cert,               <Certificate>
                  time_stamp,         <Time stamp of original sending time>
                  sequence_number,    <Monotonically increasing sequence number>
                  sender_signature    <Digital signature of sender>},
               data                   <Array of bytes of the upper layer message>}
```
b)

```
Struct trusted_issuer [] {issuer,            <Identity of the issuer (TTP)>
                          issuer_public_key  <Public key of the issuer>}
```
c)

```
Struct trusted_neigbor [] {node_address,  <Address of the node>
                           cert           <Certificate of the node>}
```
d)

Figure 4: Structures of a) certificate, b) radio message, c) trusted issuer, d) trusted neighbor

Table 1: Summary of generic functions and constants

Name	Description of the function or the parameter.
SEND(*msg*)	Sends message (*msg)* over the radio.
msg = RECEIVE()	Returns received message (*msg)* from the radio.
hash_value = HASH(msg)	Calculates and returns fixed length digital checksum (*hash_value)* of the message (*msg)* (e.g., SHA-256 function can be used (NIST, 2015)).
result = SIGN(key, text)	Signs using the private key (*key),* the given information (*text*), and returns its signed result (*result*). (e.g., function ECDSA-256 is usable (RSA Security Inc., 2004)).
verification = VERIFY(key, signed_hash, clear_hash)	Verifies using the public key (*key) that* the signed hash (*signed_hash*), which the sender generated, matches to the clear hash (*clear_hash*) that the receiver calculated. If the verification is successful, the return value (*verification)* is OK; otherwise FAIL. (e.g., function ECDSA-256 is usable (RSA Security Inc., 2004)).
validity = VERIFY_CERT(key, cert)	Verifies certificate's (*cert)* validity time and integrity using issuer's public key (*key)*. If the validation is successful, the return value (*validity)* is OK; otherwise FAIL.
PROCESS(*content*)	Delivers the received message's content (*content)* for processing at the upper layers of the node.
time = CURRENT_TIME()	Returns the current time (*time)* of the system.
ADD(*trust*_list, [*addr*, *cert*])	Adds into a list of trusted neighbors (*trust_list)* a new node with an address (*addr)* and a certificate (*cert)*.
found = IN_LIST(issuer_list, *issuer*)	Checks whether a given issuer (*issuer)* is in the list of trusted issuers (*issuer_list)*. If the issuer is in the list, the return value (*found)* is OK; otherwise, FAIL.

Name	Description of the function or the parameter.
in_time_limit =IN_TIME_LIMIT(now, sending_time, hysteresis)	Checks whether the receiver's current time (*now)* and the time when the sender sent the message (*sending_time)* are within the time hysteresis (*hysteresis)*. If the time difference is less than or equal to the time hysteresis allowed, the return value (*in_time_limit)* is OK; otherwise, FAIL.
TIME_HYSTERESIS	Environment dependent parameter that specifies how old messages are still acceptable as valid messages.

```
ME.cert   = cert {issuer                    = ”TTP1”,
                  subject                   = <A’s public key>,
                  validity                  = <Valid from … to …>,
                  property                  = <A’s properties>
                  issuer_signature          = <TTP1’s signature to indicate that A is trustworthy >}
ME.private_key                              = <A’s private key>
ME.public_key                               = ME.cert.subject
ME.sequence_number                          = <Random starting point for the sequence number >
ME.my_address                               = <A’s network address>
ME.trusted_issuers[]                        = trusted_issuer [
            {issuer=”TTP1”, issuer_public_key = <TTP1’s public key>},
            {issuer=”TTP2”, issuer_public_key = <TTP2’s public key>},
            {issuer=”TTP3”, issuer_public_key = <TTP3’s public key>}]
ME.trusted_neighbors[] = []
```

Figure 5: At Drone *A* - Initializations of parameters (done online at the depot and updated during secure onboarding on the battlefield; ref. 1st step in Figure 1

The send_message function (Figure 6) constructs the message and sends it. It fills the standard header fields and constructs the extra header field content. The sender includes its certificate and then increments and adds the message sequence number and a time stamp to the extra header. Finally, it calculates a hash over the message, signs the hash with its own secret key, and places the result into the extra header.

```
function send_message(me, destination_address, source_address, std_header_parameters, message_data) {
  msg.standard_header.to = destination_address
  msg. standard_header.from = source_address
  msg. standard_header.std_hdr_params = std_header_parameters
  msg.data = message_data
  msg.extra_header.cert  = me.cert
  me.sequence_number +=1
  msg.extra_header.sequence_number = me.sequence_number
  msg.extra_header.time_stamp = CURRENT_TIME()
  msg_extra_header.signature = 0
  hash_value = HASH(msg)
  signed_hash = SIGN(me.private_key, hash_value)
  msg.extra_header.signature = signed_hash
  SEND(msg)}
```

Figure 6: Function to send a message

The check_integrity function (Figure 7) verifies the integrity of the message using the TTP keys that it has received in advance. After successful verification, the receiver knows that the message came from a trusted party, that the message is intact, and that it is timely. The verification steps proceed in an optimized priority order, which enables discarding messages resulting from, for example, unsolicited senders, as well as forged or altered messages or replay attacks. First, the procedure checks the trustworthiness of the issuer in the extra header. Second, it checks the freshness of the message i.e., the time stamp that it compares with its own clock plus/minus an allowed difference. The time stamp checking assumes a certain degree of clock synchronization in the swarm and is therefore application dependent. The pseudocode does not show the sequence number checking as this example assumes that no previous communication has occurred and there is no previous sequence number to compare with. Third, it verifies the certificate. Fourth, after the message has passed the previous checks, the computationally most intensive phase concludes the verification. The receiver calculates locally the hash value of the message and then uses the sender’s public key and the signature verification function to check that the signed hash in the message and the locally calculated hash match. If the verification proves successful, the message is intact i.e., it has not been altered or distorted during transfer.

The following code snippets utilize the previously presented functions to carry out their tasks. Devices *A* and *B* communicate (Figure 8, Figure 9) while *C* listens (Figure 10) and determines whether to trust *A*. During this procedure, *C* remains silent and merely observes the communication between *A* and *B*. The pseudocode (Figure 8, Figure 9, and Figure 10) does not address the data payload transferred between the devices, since only the integrity checking is relevant for the interest of this paper.

The presented snippets invite the reader to compare how devices *B* and *C* process a received message differently. Device *C*, upon recognizing that it is not the intended recipient, utilizes the piggybacked information in the extra header to determine whether it can unambiguously identify device *A*. Based on this information *C* can decide whether it trusts *A* for future communication and wants to use *A* as a proxy. Meanwhile, *B* as the destination of the message also verifies the integrity before further processing the embedded message. From the received information stored in the certificate's property field, *C* learns what services *A* offers. It is crucial that *C* receives enough information to decide whether to trust *A*. When *C* decides to initiate a communication with a device, which it trusts (device *A*), it will expose its existence to anyone in the neighborhood. Overall, the code snippets show that *C* can assess the trustworthiness of *A* just by listening to ordinary ongoing messages between its neighboring devices.

```
function check_integrity(me, msg) {
 if (IN_LIST(me.trusted_issuers, msg.extra_header.cert.issuer) == OK) and
   (IN_TIME_LIMIT(CURRENT_TIME(), msg.extra_header.time_stamp, TIME_HYSTERESIS) == OK) and
   (VERIFY_CERT(me.trusted_issuers[msg.extra_hdr.cert.issuer].issuer_public_key,
                    msg.extra_header.cert) == OK {
      signed_hash = msg.extra_header.sender_signature
      msg.extra_header.signature = 0
      calculated_hash = HASH(msg)
      if (VERIFY (msg.extra_header.cert.subject, signed_hash, calculated_hash) == OK) {
        # Integrity check was OK.
        return(OK)
      } else {
        # Integrity check failed, ignore this message
        return(FAIL)
      }
 } else {
      # Initial checks failed, ignore this message
      return(FAIL)
 }
}
```

Figure 7: Function, which checks validity of a received message

```
send_message(
 ME,
 ME.trusted_neighbors["B"].node_address,  # Destination trusted neighbor B
 ME.my_address,                           # Sender is "me"
 <standard header parameters>,            # Standard protocol header
 <message_data>)                          # Actual message to B (can be encrypted on upper protocol level)
```

Figure 8: Logic for *A* to send a message to *B*

```
msg = RECEIVE()                          # Receive message
if (check_integrity(ME, msg) == OK) {  # Verify that message integrity is OK
 if (msg.standard_header.to == ME.my_address} {        # Check if message is for node B,
   PROCESS(msg.data)                   # Yes, process the data part
 }
}
```

Figure 9: Logic for *B* to receive a message

```
msg = RECEIVE()                          # Receive message
 if (check_integrity(ME, msg) == OK) {  # Verify that message integrity is OK
  if (msg.standard_header.to == ME.my_address} {        # Message is not for node C, don't process data part
   PROCESS(msg.data)
  }
  ADD(ME.trusted_neighbors, [msg.from, msg.extra_header.cert])
                                         # New trustworthy neighbor found, add it to list
  if ("something to send to A") {        # Is there a need to send something to A?
   send_message(ME, ME.trusted_neighbors["A"].node_address, ME.my_address, <standard header parameters>, <some data>)
  }
 }
```

Figure 10: Logic for *C* to detect a trustworthy neighbor and send data to *A*

4. Discussion

This paper presents a methodology by which a drone can determine if another device is a trustworthy ally. The proposed scheme enables devices to remain silent while assessing the trustworthiness of their neighbors, whose services they might utilize. Such services could include acting as a repeater (i.e., a proxy), in which the proxy sends messages with high power on behalf of the hidden devices. Although traditional stealth features are losing value (Lehto and Hutchinson, 2020), the indicated silence could serve as an abstract replacement for stealth. When such an abstract stealth mechanism exists, the natural next step is to explore the functionalities it enables and the use cases it supports. This concept facilitates the development of new types of swarm devices that maximize existence concealment. For instance, a valuable drone can listen to available low-cost repeater drones that beacon their service offerings. Such proxy nodes can either piggyback information about their service offerings in their regular communication or send separate broadcasts containing such information. While this "stealthy mode of operation" does not render the device entirely undetectable, it minimizes radioactivity by initiating its own transmissions only after securely identifying its neighbor.

Traditional authentication protocols (Kaufman, C. ; Hoffman, P.; Nir, Y.; Eronen, P.; Kivinen, 2014; Moskowitz *et al.*, 2015; Rescorla, 2018) require the exchange of multiple messages between devices. They force valuable devices, such as *C*, to expose themselves to potential adversaries too easily. Therefore, such protocols are not well suited for battlefield scenarios where a valuable device strives to remain silent for as long as possible, and even then, to use low transmission power and a minimum number of messages. To fulfil the use case discussed in this paper, only one message from the valuable device is necessary. In some situations, devices may not need to reveal their presence at all. For example, this could occur when a valuable device listens to broadcasted location information of fixed devices at regular intervals to countermeasure GPS-jamming. A device placed in a fixed location can reliably advertise its location to mobile devices. The mobile devices only need to listen, because the mechanism in this paper enables reliable identification of the sender and message integrity validation even from one-way communication.

Clock synchronization requirements will vary depending on the applications, but it is crucial to prevent replay attacks. All steps of the presented certificate verification process do not necessarily need to occur for every message. Since the cryptographic operations are the most complicated and time- and energy-consuming parts in the proposed model, it is possible to cache already verified certificates locally. This reduces the number of cryptographic operations in the verification process to one, i.e., to checking the signed hash of every message. We emphasize that the method proposed in this paper concentrates only on message integrity and unambiguous identification of the sender, not protecting confidentiality of the messages. Thus, encryption of information on other layers is beyond the scope of this paper.

The presented solution poses some challenges. Devices can remain in the depot for extended periods. Therefore, the TTP key must be robust enough to withstand long-term storage and to ensure certificate longevity. The strength of the certificate relies heavily on the key lengths. As a TTP certificate is included into every radio network message, longer keys add overhead. Previous studies, which include practical implementations, estimate the overhead to be in the ballpark of 13-27% when the size of the data part of the message is between 500 and 1000 bytes (Lagutin, 2010). To minimize this overhead, this paper proposes to use a shorter identity of the TTP instead of its public key. In practice, short identities of trusted TTPs and their public keys are loaded into the devices already at the depot.

Like many other security mechanisms, this proposal relies on the strength of the selected cryptographic algorithms and proper custody of secret keys. The underlying assumption is that the secret key used for signatures remains confidential and does not leak. A proper hardware implementation of the cryptographic

algorithm and the keystore provides a required level of protection. This article outlines key requirements that, alongside specific use case criteria—such as public key cryptography, key lengths, and computational capacity—facilitate the selection of an appropriate cryptographic algorithm. While the final choice of algorithm lies beyond the scope of this paper and warrants further investigation, this study identifies critical factors to consider. These include key length, computational power, hardware acceleration, protection requirements, signing and verification computational load, expiration timeframes for protection, and resistance to quantum threats. Ultimately, the optimal solution will represent a balanced trade-off among these criteria.

However, if the entire device falls into the wrong hands or begins to malfunction, its trustworthiness is at risk. Such risks can be mitigated either by limiting certificate lifetimes or by introducing a black list -feature. Future research should establish a mechanism for revoking compromised devices or TTPs from the system. It would be straightforward to add checking of black listed TTPs or nodes into the suggested check_integrity function. However, determining who has the authority to make revocation decisions (i.e., to add identities to the black list) is more complex. Additionally, securely distributing such revocation information during an ongoing mission can be challenging. This topic also requires finding a trade-off between revocation list updates and the need for minimalism in transmission and conservation of battery power. It is crucial to investigate how to optimize the risks and benefits of various approaches. This includes weighing factors such as mission length, relevance of keeping communication secure, sending messages, or taking the risk of short-term compromises with longer intervals between certificate updates. Prolonged military operations, such as the war in Ukraine, have demonstrated that the long-term ability to maintain operational readiness as well as the energy resources of autonomous systems require attention. Therefore, implementations must consider the durability of solutions and algorithms as well as the updateability of certificates and keys. This study has shown that a valuable device can maximize its concealment, but practical applications of this concept require further exploration. Additional research topics include assessing requirements, and identifying potentially suitable hash and digital signature functions for battlefield applications of drone swarms. Similarly, it would be useful to clarify and suggest the usage of additional fields in the certificates. For example, the properties field can have a specific indicator for a device that can act as a repeater. Overall, listing and defining what fields to add would be useful. Further studies should investigate suitable protection of the properties field from leaking information to adversaries.

There is a need to study the consequences of increased autonomy of individual drones as well as swarms. The relationship between the human operator and the equipment evolves as humans are phasing out of the loop. With increased autonomy, the operator no longer controls individual drones. Instead, he issues commands that the swarm executes independently using autonomous decision-making and most likely artificial intelligence. Thus, the operator transitions from being the controller of individual drones to being the one issuing commands to a swarm of drones operating autonomously on the battlefield. The presented solution supports a high level of autonomy.

In the literature, approaches to trust management vary. Some focus on reputation, while others aim to solve trust management for drone swarms by optimizing communication, networking, and autonomy against power consumption and immediacy (Maiden *et al.*, 2009). However, each architecture and target system exhibit different restrictions. Some studies examine untrustworthy swarm members and attack taxonomy to enable better protection (Vojnar, Bierska and Buhnova, 2024). Other studies analyze different levels of autonomy versus control over swarms (Saffre, Hildmann and Karvonen, 2021) and swarm intelligence (Qamar *et al.*, 2022). Altogether, trust management studies found in the literature would benefit from the proposed mechanism of this paper that filters out untrustworthy messages and devices. Combined with the revocation of compromised nodes, it serves as a foundation for further development.

5. Conclusions

This paper explores the use of drone swarms in coalition missions. The primary contribution is a solution that allows a drone to distinguish friends from foes instantly and securely without prior encounters. A secure identification mechanism enables autonomous drones to operate effectively. The solution facilitates the construction of trust management in drone swarms operating on the battlefield.

Traditional multiphase security handshakes and identity checking protocols are not optimal for battlefield operations as the radio communication reveals the existence of devices to adversaries prematurely. A novel usage of public key cryptographic methods facilitates trust management in autonomous drone swarms with a minimum of messaging. In particular, the integrated identification information reduces the number of messages over the radio. The proposed solution allows drones to identify allies securely and quickly without revealing their presence untimely. The cornerstone is the pre-deployment configuration that allows drones to self-organize and

collaborate independently on the battlefield. This approach enables a valuable drone to maximize stealthy mode operation and conceal its existence until it has securely identified a friendly device whose services it can utilize. The solution provides a foundation for trust management in coalition drone swarms, by promoting efficient and secure cooperation even in chaotic battlefield environments. This paper does not yet analyze where such a stealthy mode would be suitable. However, the study shows that the presented solution can enable new use cases and approaches for protecting the existence of a valuable device, but the precise application of it requires further research.

Acknowledgements

The authors would like to thank Otso Helminen and Juha Mäkelä for their valuable comments, which helped improve the quality of this manuscript.

Ethical clearance was not required for this research.

AI tools were not used for the creation of this paper.

References

Bennett, N. (2015) *Introduction to Algorithms and Pseudocode*, *Working paper in Project "Exploring Modelling and Computation"*. Available at: https://www.nickbenn.com/sites/default/files/projects/documents/Introduction to Algorithms and Pseudocode.pdf (Accessed: 25 April 2025).

Deering, D. S. E. and Hinden, B. (2017) *Internet Protocol, Version 6 (IPv6) Specification*, *IETF webpage*. https://doi.org/10.17487/RFC8200. Available at: https://www.rfc-editor.org/info/rfc8200 (Accessed: 12 December 2024).

GlobalSecurity.org (2011) *Identification Friend or Foe [IFF]*. Available at: https://www.globalsecurity.org/military/systems/aircraft/systems/iff.htm (Accessed: 15 December 2024).

Hoang, M.-T. O. *et al.* (2023) *Drone swarms to support search and rescue operations: Opportunities and challenges*, *Cultural Robotics: Social Robots and Their Emergent Cultural Ecologies*. https://doi.org/10.1007/978-3-031-28138-9_11. Springer. Available at: https://nielsvanberkel.com/files/publications/culturalrobotics2023a.pdf (Accessed: 25 April 2025).

ITU-T (2019) *X.509 : Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*, *ITU Webpage*. Ínternational Telecommunication Union. Available at: https://www.itu.int/rec/T-REC-X.509-201910-I/en (Accessed: 12 December 2024).

Javed, S. *et al.* (2024) *State-of-the-art and future research challenges in uav swarms*, *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2024.3364230. IEEE. Available at: https://ieeexplore.ieee.org/abstract/document/10430396 (Accessed: 25 April 2025).

Johnson, D. B., Arkko, J. and Perkins, C. E. (2011) *Mobility Support in IPv6*, *IETF webpage*. https://doi.org/10.17487/RFC6275. Available at: https://www.rfc-editor.org/info/rfc6275 (Accessed: 12 December 2024).

Kaufman, C. ; Hoffman, P.; Nir, Y.; Eronen, P.; Kivinen, T. (2014) *RFC 7296 - Internet Key Exchange Protocol Version 2 (IKEv2)*, *Internet Engineering Task Force (IETF)*. https://doi.org/10.17487/RFC7296. Available at: https://datatracker.ietf.org/doc/html/rfc7296 (Accessed: 15 July 2024).

Kent, S. (2005) *IP Encapsulating Security Payload (ESP)*, *IETF webpage*. https://doi.org/10.17487/RFC4303. Available at: https://www.rfc-editor.org/info/rfc4303 (Accessed: 12 December 2024).

Kortesniemi, Y. (2015) *Access Control in Distributed Systems using SPKI Authorisation Certificates*, *Aalto University publication series DOCTORAL DISSERTATIONS, 63/2015*. Aalto University. Available at: https://urn.fi/URN:ISBN:978-952-60-6194-8 (Accessed: 15 July 2024).

Lagutin, D. (2010) *Securing the Internet with digital signatures*, *TKK Dissertations 255*. Aalto University. Available at: https://urn.fi/URN:ISBN:978-952-60-3465-2 (Accessed: 23 October 2024).

Lassfolk, C. and Kari, H. (2025) 'A trust management concept for secure onboarding of military coalition drone swarms', in *Cyber Security - Policy and Technology*. 'unpublished', pp. 1–23.

Lehto, M. and Hutchinson, B. (2020) *Mini-drones swarms and their potential in conflict situations*, *15th international conference on cyber warfare and security*. https://doi.org/10.34190/ICCWS.20.084. Available at: https://urn.fi/URN:NBN:fi:jyu-202005193305 (Accessed: 25 April 2025).

Maiden, W. M. *et al.* (2009) *Trust management in swarm-based autonomic computing systems*, *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. https://doi.org/10.1109/UIC-ATC.2009.87. Available at: https://ieeexplore.ieee.org/abstract/document/5319265 (Accessed: 25 April 2025).

Moskowitz, R. *et al.* (2015) *Host Identity Protocol Version 2 (HIPv2)*, *Internet Engineering Task Force (IETF)*. https://doi.org/10.17487/RFC7401. Available at: https://www.rfc-editor.org/info/rfc7401 (Accessed: 25 April 2025).

NIST (2015) *FIPS 180-4: 'Secure Hash Standard (SHS)'*, *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION*. http://dx.doi.org/10.6028/NIST.FIPS.180-4. Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf (Accessed: 14 December 2024).

NIST (2023) *FIPS 186-5: 'Digital Signature Standard (DSS)'*, *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION*. https://doi.org/10.6028/NIST.FIPS.186-5. Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf (Accessed: 14 December 2024).

Qamar, S. *et al.* (2022) *Autonomous drone swarm navigation and multitarget tracking with island policy-based optimization framework*, *IEEE Access*. https://doi.org/10.1109/ACCESS.2022.3202208. IEEE. Available at: https://ieeexplore.ieee.org/abstract/document/9868790 (Accessed: 25 April 2025).

Rescorla, E. (2018) *The Transport Layer Security (TLS) Protocol Version 1.3*, *Internet Engineering Task Force (IETF)*. https://doi.org/10.17487/RFC8446. Available at: https://www.rfc-editor.org/info/rfc8446.

RSA Security Inc. (2004) *RSA Security Inc. Public-Key Cryptography Standards (PKCS)*, *RSA Laboratories*. Available at: https://www.cryptsoft.com/pkcs11doc/STANDARD/pkcs-11v2-20.pdf (Accessed: 21 December 2024).

Saffre, F., Hildmann, H. and Karvonen, H. (2021) *The design challenges of drone swarm control*, *Engineering Psychology and Cognitive Ergonomics*. https://doi.org/10.1007/978-3-030-77932-0_32. Available at: https://cris.vtt.fi/en/publications/e0eb4f04-db0a-4f87-8325-7ea83ad06f61 (Accessed: 25 April 2025).

Tarsini, I. and Anggraeni, R. (2024) *Explore flowchart and pseudocode concepts in algorithms and programming*, *Indonesian Journal of Multidisciplinary Science*. https://doi.org/10.55324/ijoms.v3i5.807. Available at: https://ijoms.internationaljournallabs.com/index.php/ijoms/article/view/807 (Accessed: 25 April 2025).

Ulate-Caballero, B. A., Berrocal-Rojas, A. and Hidalgo-Céspedes, J. (2021) *Concurrent and Distributed Pseudocode: A Systematic Literature Review.*, *XLVII Latin American Computing Conference (CLEI)*. https://doi.org/10.1109/CLEI53233.2021.9640222. IEEE. Available at: https://ieeexplore.ieee.org/abstract/document/9640222/ (Accessed: 25 April 2025).

Vojnar, D., Bierska, A. and Buhnova, B. (2024) *Scenarios for trust management in swarm robotics*, *Proceedings of the 2024 ACM/IEEE 6th International Workshop on Robotics Software Engineering*. https://doi.org/10.1145/3643663.3643967. Available at: https://dl.acm.org/doi/10.1145/3643663.3643967 (Accessed: 25 April 2025).

Ylonen, T. *et al.* (1999) *SPKI Certificate Theory*, *Network Working Group*. https://doi.org/10.17487/RFC2693. Available at: https://www.rfc-editor.org/info/rfc2693 (Accessed: 13 December 2024).