Formalizing swarm security

Martin Strand

FFI, martin.strand@ffi.no

Abstract. Cheap, capable processors are becoming readily available, and these are being deployed to a wide variety of applications, ranging from internet-enabled refrigerators to swarms of specialised drones. As often is the case with innovations, security lags behind. We aim to introduce the fields of cryptography and swarm modelling to each other, and present a selection of security definitions and existing schemes that satisfy these. Such schemes are seen in context of different scenarios to increase security where it matters. To this end, we further develop a swarm model to identify when different key structures should be used.

Keywords: adversarial models · autonomy · lightweight

1 Introduction

For small, autonomous devices, the path from novelty to household item has been short. Anyone can come up with a potential application for a small sensor, smart device, drone or even underwater vehicles. Some of these applications are so vital or sensitive that the security requirements should be correspondingly strong. However, previous surveys [9, 19] have made it clear that the foundation for secure designs for wireless sensor networks (WSN), internet of things (IoT) or mobile ad-hoc networks (MANET) is in its infancy.

The challenge in this domain is the multitude of constraints that may apply to our devices: battery life, computing performance, weak transmitters, low bandwidth, short transmission windows, limited storage, and so on. Many of the problems are solved for ordinary networks, but those solutions may not be applicable for this setting. For example, asymmetric cryptography is often too computationally expensive.

The intention of this work is to contribute to bridging the gap between real-life constraints and cryptography theory by building on rigid definitions from the latter, and suggesting ways these definitions could be satisfied while acknowledging the particular and unique limitations that make it hard to deploy existing security practices to small devices. We want to introduce two exciting fields to each other. In one direction by curating and presenting highly successful cryptographic models and definitions to a wider security audience. In particular, we have tried to *not* provide unecessary novelty in this respect. For the other direction, by keeping a critical eye to those definitions as they meet the world of practical considerations, this work is intended to be a part of the feedback loop to the theoretical side. The presentation is also intended to reach both

cryptographers and practitioners in the domain of autonomous devices. We have kept the formalism of the original definitions even though we are not using them to prove concrete theorems in this work. The purpose of that is to provide a complete reference when designing constrained systems.

Our primary contribution is an à la carte menu of security notions an autonomous system could aspire to. We do not wish to add to the number of competing definitions. Instead, we have chosen well-established definitions from cryptography to fit reasonable requirements one could ask from such systems. We also discuss compatibility between the notions.

A secondary contribution is a refinement to how one can model a swarm of autonomous devices using graph theory. This is again a useful tool to characterise how the different definitions should be applied. In addition, we provide examples of how our contributions fit with existing solutions and scenarios.

The third pillar of this work is an introduction to challenges facing systems which may be deployed with secrets into an adversarial environment, but are denied any form of communication. This implies a theoretically impossible problem: encryption is worthless when the key is stored next to the ciphertext. However, we show that there is some leeway once one considers time as well.

1.1 Adversarial capabilities

One cannot assess security without making assumptions about the adversary. We generally assume the existence of a computationally bounded, active and adaptive adversary. For non-cryptographers, that means:

- The adversary can spend at most a polylogarithmic amount of time and space on computations, with respect to size of the keyspace. Conversely, an unbounded adversary would for instance be able to try all keys from a key space, and choose the correct one.
- An active adversary controls the network, and may read, inject, drop and modify messages on the network. In contrast, a passive adversary may only read the traffic.
- The adaptivity means that the adversary can corrupt parties at will throughout the execution, and then control their actions completely. A static adversary has to choose the set of corrupted parties before the protocol starts. All parties (or instances of such) have a freshness flag which is set to false when corrupted. We require that the final attack can only be mounted against parties that are still fresh.

On the other hand, we do not make any assumptions on what the adversary is trying to achieve. Loosely speaking, the adversary wins if it can make the system behave in any other way than defined for each notion. The observant reader will see that we have given the adversary the capability to drop all messages, and that automatically makes the adversary able to run effective denial-of-service (DoS) attacks. In order to avoid pathological cases like that, cryptographers have designed a paradigm where we challenge the adversary to a game for each security goal, and the adversary will win if it does better than a random algorithm.We will later list the specific goals for the adversary to attack.

1.2 Related work and IoT security

Previous work [9, 19] has surveyed the existing state of security modelling for WSNs, IoT, and MANETs, with discouraging results. Do, Martini and Choo note that "[o]ther security-based research should look to cryptographic protocols as the gold standard for adversary models (...)", and add: "IoT security, particularly, is a research field in its infancy." The earlier surveys cite a number of papers that in total give the impression that even routine use of authentication is lacking in many applications.

According to Silvio Micali in his IACR Distinguished Lecture at Crypto 2020, models might be one of cryptography's strongest assets. Starting from mission goals, and assumptions about the environment and adversarial capabilities, one can formulate security requirements and reason about these. From the literature we have previously reviewed, only two works are worth mentioning here. Sen [18] gives a comprehensive overview of the field, and lists a number of security and functionality requirements. The requirements are well-arranged: they formulate guarantees one wants to make, rather than describing the inner workings of a given system. Akram et al. [2] introduce an intriguing system with corresponding security requrements. However, the requirements are in much greater extent tied to the authors' suggested system.

1.3 Organisation of the paper

The paper is structured as follows. The following section introduces a new variant of how to model a swarm. The model takes into account that the topology can change over time. Section 3 introduces relevant security goals, of which application to the model is further discussed in Section 4. We then follow up in Section 5 with three different examples of how one can prioritise the definitions in practical cases. In Section 6, we discuss the related problem of protecting data at rest in cases with a high risk of adversarial corruption. Finally, we conclude in Section 7 and give an overview over some of the numerous open problems regarding strong security under these challenging conditions.

2 What is a swarm?

The mission of cryptography can be formulated as ensuring that functionalities can work even in the presence of a powerful adversary. In light of this view, a secure channel has the same functionality as any reliable channel, but with the extra property that Eve or Mallory are unable to eavesdrop or manipulate the data without detection. Similarly, one can phrase our task here as taking the definition of a swarm, and ensuring that the definition could be fulfilled, even in an adversarial environment.

However, when researching a suitable definition of a swarm, we came across the following statement by Hamann: "It is interesting to notice that there seem to be no explicit definitions of swarms in the literature" [12, Sec. 1.1.1]. We

cannot state precisely what a swarm is, but we can at least model it, with some inspiration from Hamann. In his book, he describes a random graph $G = (V, E)$ as a viable model for swarms, where the nodes represents the devices and each edge represent a communication connection. We refine the model slightly, by instead applying directed random geometric graphs, first described by Michel et al. [14]. The advantage is that we can model devices with non-uniform sending and receiving levels. We present a simplified version, which generalises random geometric graphs in the natural way, but for which some of the results by Michel et al. may not hold. In particular, we do not explicitly assume that the different radii are distributed according to a Pareto distribution.

Definition 1. *A* directed random geometric graph *is a graph $G = (V, E)$ such that each $v \in V$ is a point in $\mathbb{R}^n$ and satisfies the following:*

1. *The nodes are randomly sampled from some distribution on a (potentially bounded) region of $\mathbb{R}^n$.*
2. *For each node $v \in V$ there exists a radius r_v.*
3. *Let $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a metric. There exists an edge $e \in E$ for each pair of nodes (v, w) with $d(v, w) < r_v$.*

The graph will model the communication capabilities in the collection of devices at a given time t. Let $v \in V$ be a node and define the following two sets relative to v:

- $N_v^t = \{u \in V \mid (v, u) \in E\}$ is the neighbourhood around v at time t, which contains the nodes that can receive any messages directly from v.
- Define u to be reachable from v if there exists a path from v to u, and let $\bar{N}_v^t = \{u \in V \mid u \text{ reachable from } v\}$, which we will call the swarm around v at time t.

One can view the graph as a representation of a relation, and then let $\bar{N}_v^t$ be the transitive closure of N_v^t. This last item states that if two devices cannot even communicate through proxies, then they are effectively not in the same swarm at that moment.

Remark 1. We believe that this asymmetric definition may be a realistic model of real challenges regarding devices in challenging environments. For instance, some high-powered devices may be able to reach the whole network, but only receive from its closest neighbours. However, we hypothesize that for most real-life situations, the graphs will be undirected. This will simplify many aspects that follows, but the reader should keep in mind that one should consider the corner cases properly when designing protocols.

Next we must discuss the size of a swarm. Hamann [12] suggests between 10^2 and 10^{23}, beyond which one should use statistics to analyse the system as one would for a gas. However, Hamann remarks, one can also consider swarms of three members, and for the purpose of our work: two will also be applicable.

Looking ahead, our model will allow us to create three different scenarios, depending on the size of the swarm and the computational resources of the devices. The scenarios provide a trade-off between complexity, security and resources.

3 Security goals

Next, we provide a series of potential security goals. We stress *potential*, because this list is intended to be used *à la carte*, depending on the concrete application.

When instantiating any of these notions, one must choose a target security level and use that as input to the key generation algorithm. This security parameter is of particular importance to record the potential loss of security that may come from reductions in proofs. This paper only contains one such proof, and as that proof only contains a standard argument that halves the advantage, we have opted to omit explicitly mentioning the security parameter to increase readability.

3.1 Channel privacy

Our first definition is just the normal definition for IND-CCA2, message indistinguishability under adaptive chosen ciphertext attacks.

Definition 2 (Ciphertext indistinguishability, IND-CCA2). The adversary should not be able to read the contents of a message in the channel. *Let* $\mathcal{ES} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme.*

1. *The challenger runs* $\mathcal{ES}.\mathsf{KeyGen}$ *and sets up two oracles* $\mathcal{O}^{\mathsf{Enc}}, \mathcal{O}^{\mathsf{Dec}}$*. It then chooses a bit* b *at random, and sends a signal to the adversary* $\mathcal{A}$*.*
2. $\mathcal{A}$ *may send a large number of queries to* $\mathcal{O}^{\mathsf{Enc}}, \mathcal{O}^{\mathsf{Dec}}$*. Eventually, the adversary submits two messages* m_0, m_1 *of identical length.*
3. *The challenger encrypts* $c = m_b$*, and returns it to the adversary.*
4. *The adversary may again query the oracles, except for decryption oracle calls on the ciphertext* c*. Finally, the adversary submits a bit* b'*, and wins if* $b = b'$*.*

The scheme $\mathcal{ES}$ *is IND-CCA2-secure if the adversary's advantage*

$$\mathsf{Adv}_{\mathcal{ES}}^{ind\text{-}cca2}(\mathcal{A}) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

is negligibly small.

The definition of a negligible value or function may depend on the application. For asymptotic arguments it is defined as a function eventually being smaller than $1/p(\lambda)$ for any polynomial p in the security parameter λ. For example $1/2^{\lambda}$, is a negligible function. On the other hand, concrete systems may define anything smaller than a given constant as negligible.

3.2 Swarm privacy

Now we discuss what we mean by swarm privacy. A swarm may consist of several devices that leave and join during the mission. This problem is related to that of broadcast encryption and multicast encryption [10].

The fundamental goal is that any message sent within a swarm should be readable for any active member, but not for outsiders and lost devices. This means we have to be able to handle join and leave operations while the system is online. An important question is how one can decide which devices should be considered lost, and henceforth apply the leave operations on those. A more complicated question is that of *who*: is this an individual decision, a swarm consensus, or a question of hierarchy and hence trust?

Remark 2. The reader may at this point compare this problem to that of group chats, and consider group key exchange protocols. Such protocols are outside the scope of this work, as they demand too much network traffic, constant liveliness, and expensive computations. However, it is worth mentioning a recent preprint by Weidner et al. [20], which aims to provide decentralised key agreement for large groups. A special emphasis is placed on post-compromise security (PCS) and forward secrecy (FS). Weidner et al. are discussing the difficult problem of how one should reach consensus about membership.

Definition 3 (Swarm privacy). Only active, honest players may read broadcast messages. *We set up a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.*

- *The challenger runs the setup and key generation algoritms for a system of n users. It also chooses a bit b uniformly at random.*
- *The adversary may query the oracle multiple times, each time one of the following queries*
  - $\mathsf{corrupt}(i)$ *The key material held by user i is revealed to $\mathcal{A}$.*
  - $\mathsf{Enc}(S, m)$ *The message m is encrypted with the members of a subset S of $\{1, ..., n\}$ as intended receipients.*
  - $\mathsf{Dec}(S, c)$ *If S is the correct audience for the ciphertext c, then the oracle outputs the decryption m.*
- *The adversary selects two messages m_0, m_1 of equal length and submits them to $\mathcal{C}$ along with a set S^*. The challenger verifies that S^* does not contain any user previously corrupted and only then responds with $c' = \mathsf{Enc}(S^*, m_b)$.*
- *The adversary may again provide new queries with two restrictions:*
  - *any query $\mathsf{corrupt}(i)$ is on the condition that $i \notin S^*$, and*
  - *the adversary may not query $\mathsf{Dec}(S^*, c^*)$.*
- *The adversary submits a bit b' and wins if $b = b'$.*

We define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{swarm}(\mathcal{A}) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

and say that the system is secure if $\mathsf{Adv}^{swarm}(\mathcal{A})$ is negligible.

This definition is intended to capture the essence of two previous definitions by Gentry and Waters [11], and Panjwani [16], which were specific to the asymmetric and symmetric cases respectively. We return to a more thorough discussion of similarities and differences later.

3.3 Authenticity

A swarm may be used to observe a large area and provide its sightings to a central intelligence service. Recall that we assume that the adversary is able to inject messages into the network. The adversary can then try sending false reports to lure the others. We therefore want swarm members to be able to attribute messages to their senders. This is known as authenticity, and is captured by recalling the definition of ciphertext integrity (INT-CTXT):

Definition 4 (Ciphertext integrity, INT-CTXT). An encrypted message should be received as intended by its sender. *Let* $\mathcal{ES} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a symmetric encryption scheme and let* $\perp$ *denote error. We set up the following experiment* $\mathsf{Exp}^{\textit{int-ctxt}}_{\mathcal{ES}}(\mathcal{A})$*:*

- *Generate a key* $K \leftarrow \mathsf{KeyGen}$*, and set* $S = \varnothing$*.*
- *Give the adversary access to encryption and decryption oracles* $\mathcal{O}^{\mathsf{Enc}}, \mathcal{O}^{\mathsf{Dec}}$*, with the additional programming that:*
  - *whenever the oracle* $\mathcal{O}^{\mathsf{Enc}}$ *responds with some* c_i *when queried on some message* m_i*,* S *is updated as* $S = S \cup \{c_i\}$
  - *when* $\mathcal{A}$ *queries* $\mathcal{O}^{\mathsf{Dec}}$ *with* c*, let* m *be the output from decrypting* c*. If* $m \neq \perp$ *and* $c \notin S$*, halt and output 1*

  The encryption scheme $\mathcal{ES}$ *is INT-CTXT-secure if*

$$\mathsf{Adv}^{\textit{int-ctxt}}(\mathcal{A}) = \Pr[\mathsf{Exp}^{\textit{int-ctxt}}_{\mathcal{ES}}(\mathcal{A}) \textit{ outputs } 1]$$

  is negligible for all adversaries $\mathcal{A}$*.*

Note, however, that this definition does not protect against replay or reordering attacks. For an example of a stronger notion, see Bellare, Kohno and Namprempre's stateful definition INT-sfCTXT [4]. Furthermore, it does not tie the concept of "sender" to a device identification. Such binding may be done implicitly by a key.

3.4 Anonymity

We also want messages to remain anonymous so that any adversary monitoring the network can do no better than traffic analysis to assess who sent which message. We have reviewed two works that investigate this problem for symmetric encryption, by Chan and Rogaway [6] (anonymous nonce-based authenticated encryption, anAE), and Banfi and Maurer [3] (probabilistic authenticated encryption, pAE). The latter considers the special case of probabilistic encryption, while the first is more general, but has to add more machinery around the encryption scheme in order to satisfy their own definition. Nonetheless, we opt for the more general definition, as it is closer to the common use of symmetric encryption. We give an informal presentation of the notion here, and refer to the original work for the details[1].

[1] We justify this omission for two reasons: The whole description with proper context fills a complete section, and our presentation would not improve on Rogaway and Chan's Figure 1, which we highly recommend to the reader.

The adversary is challenged to distinguish between two games: one implementing the real cryptosystem Π, and one implementing an ideal functionality.

The key difference between the two is that the ideal functionality replaces the encryption oracle with one that returns a random string of equal length, while storing the original request. This ensures that the ideal functionality provides both confidentiality and privacy, since the ciphertext is independent of both the plaintexts and the identity. The ideal decryption answers with the error symbol $\perp$ unless the decryption request perfectly matches a record stored by the encryption service, and the nonce is within some accepting policy Nx. The adversary wins if it outputs 1 while interacting with the real game, and loses otherwise.

Security is then defined as the quantity

$$\mathsf{Adv}^{\mathrm{anae}}_{\Pi,\mathsf{Nx}}(\mathcal{A}) = \Big|\Pr[\mathcal{A}^{\mathrm{real}^{\mathrm{anae}}_{\Pi,\mathsf{Nx}}} \to 1] - \Pr[\mathcal{A}^{\mathrm{ideal}^{\mathrm{anae}}_{\Pi,\mathsf{Nx}}}] \to 1\Big|.$$

Interestingly, Rogaway and Chan's concept of nonce policies can also be used to avoid replay attacks.

3.5 Topology hiding

As well as keeping each player anonymous, we would like to hide the network topology from the adversary. We adapt a definition by Moran, Orlov and Richelson [15] to our concrete case. The definition is based on the assumption that any node can detect all neighbours within its range, i.e. count its edges. Notice that the following formulation is static in the sense that the adversary has to choose which players to corrupt early in the game. This seems unavoidable, as adaptive corruptions would essentially give the adversary a way to traverse the graph.

Recall that $G = (V, E)$ is a directed graph, and that N_v^t is the neighbourhood around v at time t. Let $G' = (V, E')$ be the associated undirected graph such that for all $v_1, v_2 \in V$, let $(v_1, v_2) \in E'$ if either $(v_1, v_2) \in E$ or $(v_2, v_1) \in E$. We fix (and therefore omit) the time t for now, due to the non-adaptive nature of this definition.

Definition 5. *Let $\mathcal{G}$ be a set of undirected graphs with at most n nodes, and let Π be a protocol capable of running on all $G \in \mathcal{G}$. Each player $P_1, \ldots, P_N$ is equipped with key material $k_1, \ldots k_N$.*

- *$\mathcal{A}$ chooses a corrupt subset S and learns the key material for all players $P \in S$ and two graphs $G_0 = (V_0, E_0), G_1 = (V_1, E_1)$ such that $S \subseteq V_0 \cap V_1$ and $N_{P,G_0} = N_{P,G_1}$ for all $P \in S$, i.e. that all of the corrupted nodes have equal neighbourhoods in both graphs. $\mathcal{A}$ sends (S, G_0, G_1) to the challenger.*
- *The challenger chooses a random bit b and runs Π on G_b. It interacts with $\mathcal{A}$ for all traffic involving $P \in S$.*
- *Finally, $\mathcal{A}$ outputs a bit b', and wins if $b = b'$.*

The protocol Π is indistinguishable under chosen topology attack *(IND-CTA secure) over $\mathcal{G}$ if the quantity*

$$\mathsf{Adv}_{\pi}^{ind\text{-}cta}(\mathcal{A}) = \left|\Pr[b = b'] - \frac{1}{2}\right|$$

is negligible.

The adversary sees all its neighbours at the physical layer. The consequence of fixing the time is that one may be unable to guarantee anonymity if the network topology changes. Honest parties beyond the range of the adversary may change without impacting the gist of the definition. We have previously hypothesized that most graphs will be undirected in practice. This definition will apply in those cases. Applications must consider these limitations.

4 Applying the definitions to the swarm models

We will now bind the security requirements to our model of a swarm. Recall that we defined the two sets N_v^t and $\bar{N}_v^t$, which are the devices that are directly within range of v, and those reachable through relays. We defined the latter as the swarm around v. Due to our restriction to constrained devices, we only consider symmetric keys, and consider a device a member of the swarm if it has a valid key.

From this, we consider three scenarios:

Individual keys All devices share keys pairwise. Hence, every message must be sent to all other devices individually. This might be beneficial for very small swarms with low communication rate and high bandwidth, but limited resources to negotiate group keys. In this case, N_v^t and $\bar{N}_v^t$ exist only for bookkeeping.

Local group keys The node v maintains a group key for the members of N_v^t, and updates it if $N_v^t \neq N_v^{t+1}$. Limited group keys might reduce the consequence if a device is lost, and may be a favourable choice for larger networks with low mobility and a strong adversarial threat.

Global group keys The node v maintains a group key for $\bar{N}_v^t$ and updates if necessary. If $\bar{N}_v^t = \bar{N}_w^t$ for all $w \in \bar{N}_v^t$, then a fixed v may if necessary act as a key centre for the swarm.

For each of these, we must handle two operations: join and leave. This single sentence is worth a series of papers by itself. A device may join if it can produce valid credentials, i.e. having been keyed appropriately. Depending on the above scenario, appropriate rekeying of the other devices in the swarm may follow. Leave, on the other hand, is only somewhat easy if the device itself announces its intention of parting. If the device leaving was still trusted, it could just be asked to delete its keys, and the swarm would not have to replace keys held by its former member. However, the leave operation should imply that the device is no longer trusted, and quite possibly already have been corrupted by the adversary.

The conditions on when the swarm should initiate a forced leave is outside the scope of this work. We also acknowledge the multitude of problems connected to just the four words "when the swarm should", but have to postpone those to future work.

Furthermore, we need to demonstrate whether it is feasible to satisfy the security requirements we have stated above for the different scenarios.

4.1 Individual keys

We start with the easiest case. Since all neighbours share individual keys, there is no group, and so Definition 3 and Definition 5 are vacuously satisfied. For individual keys, join and leave is handled implicitly. One can choose to do key exchange during operation, or pre-key the devices for all admissible pairs.

To satisfy channel privacy and device accountability we can use any standard authenticated encryption, and the anonymity requirement can be satisfied by using Chan and Rogaway's protocol [6].

4.2 Group keys

We treat local and global group keys together. They differ in scope, but not necessarily in technique. To handle join and leave, we suggest using the protocol of either Gentry and Waters [11], or Panjwani [16]. The former describe a public key protocol: to send a message using the protocol, one specifies a set S of recipients and encrypts it using a function of every recipent's public key. A single private key is sufficient (and necessary) to decrypt. We refer to the original paper for the technical details. This protocol allows the group to distribute a new key which can again be used for communication until the next topology change. The authors prove that their protocol satisfies a security definition similar to ours.

Proposition 1. *The Gentry-Waters system satisfies swarm privacy.*

The proof is a simple composition of two definitions and a conversion between real-or-random and left-or-right notions, and is included in the full version of the paper.

One apparent drawback with the work of Gentry and Waters is that it is based on a variant of the Diffie-Hellman problem and bilinear maps over elliptic curves. While this is considered secure today, it will not be able to stand against a quantum computer, and so there is a need to reinstantiate their concepts using post-quantum techniques.

For the symmetric case, one can use Panjwani's improvement to the Logical Key Hierarchy protocol [13]. The idea is that one builds a binary tree of keys where each leaf node represents a device, and the root node holds the group key for the complete set. All devices know the keys on the path between themselves and the root node. In order to send a message (say, a group key) to a subset S of the leaf nodes, one must choose the minimal set of keys held by members of

S, but such that for any node v not in S, no key in the path between v and the root is included.

The tree must be maintained by a key centre, which is then responsible for distributing new node keys for each topology change, and a group key to the active devices. Definition 3 is an adaptation of the definition used by Panjwani.

In sum, these two approaches provide join/leave functionality. However, what they have in common is that the topology is partly leaked every time it is invoked, and any system employing either of these techniques cannot hope to be topology hiding over time. Further research is needed to see if one can reconcile topology hiding and group encryption.

5 Example applications

We can now apply our work to some real-world examples, for each highlighting their unique limitations. Examples like these can be composed across domains with the help of relay nodes.

5.1 Submarine communications

Consider a small set of autonomous underwater vehicles (AUV). Weight may not be the primary issue, so we can assume sufficient computing abilities. However, every transmission is through an acoustic channel whose bandwidth may only be around 500 bit/s. This requires the overhead to be as small as possible. This exact problem was analysed and tested by Dini and Duca [8]. Their work fits our model very well:

- In the simple case, there is a gateway g with $N_g^t = V$, while for all other nodes v, $N_v^t = \{g\}$. Hence, nodes communicate directly with the gateway, which in turn forwards the message to the intended receiver.
- Channel privacy is provided by AES used in CBC-CTS mode to avoid overhead.
- The authors use a standard message authentication code (MAC) to provide integrity. However, due to the low bandwidth, they truncate it down to 32 bits, noting that the same bandwidth makes an online attack extremely time-consuming. Hence, the adversary's success probability is limited not in having an overwhelmingly large denominator, but by realising that the numerator is bounded by the physical surroundings. This can be modelled by restricting the number of decryption oracle queries in the security game.
- Swarm privacy can be satisfied due to their group key distribution and revokation. Their key distribution is closely related to that of Panjwani.
- Topology hiding is not an issue considered by Dini and Duca. Recall also that the methods we discussed earlier are incompatible with this notion.

5.2 A swarm of drones

Small, airborne drones have a completely different set of limitations. The bandwidth is high, but the onboard computer should be small and use as little energy as possible on computations.

A key feature of drone swarms is that its members can be replaced continuously, and we must assume that the devices shift their position relative to each other. We therefore find ourselves in a situation where group keys may be the right choice.

- Channel privacy and ciphertext integrity can be achieved by using a suitable authenticated encryption scheme.
- Swarm privacy follows the discussion of group keys, but is dependent on a robust mechanism for deciding which members should be excluded.
- Anonymity and topology hiding may or may not be important for this scenario, and may in fact be less important the more members of the swarm, as each drone becomes less crucial. Hence, we expect swarm privacy to take precedence.

In practice, such devices could be augmented with tamper-resistant cryptographic modules, in which one can place a reasonable level of trust.

6 Data at rest

Small, resource constrained devices do not only pose a challenge for communication. One should also expect that such devices could be captured in a working state and brought directly to a highly skilled laboratory: it may inject or extract any instruction or data through means that the original designer could use[2]. For this section, we no longer consider the graph of communicating devices, but focus on one particular, possibly isolated, device. We describe three informal security requirements, and then sketch how they could be satisfied.

Data authenticity Only data from authenticated functionalities should be accepted by the device.

Data obsoletion Any data no longer needed for the mission should remain unreadable.

Secure storage Data still needed for the mission should not be extractable by the adversary.

Data authenticity suggests that the data stored on the device should only originate from pre-loading, communication channels or sensors, all of which could sign the data before providing it to the on-board computer. The on-board computer should then only use the data in computations after verifying the authentication tags. While this may seem cumbersome for small devices, it has already been

[2] In essence, a statement from the manufacturer like "it is theoretically possible, but it would be very hard" here translates to "this lab can do it".

tried on a drone with fascinating success [7]. The advantage would be – given certain assumptions – that the adversary could not insert malicious instructions or data on a device and then release it back into operation. Note that we do not consider the independent problem of the adversary exposing the sensors for misleading surroundings or physical attacks on the device.

Data obsoletion includes data that could be useful *after* the mission, e.g. sensor data stored on a device and only unloaded at a later stage. We propose two simple strategies: either using public key cryptography where only the public key is given to the device, or the perhaps more computationally friendly variant where the device is equipped with a symmetric key K_0. After each data object has been encrypted using some key K_i, replace K_i with $K_{i+1} = f(K_i)$ for some suitable one-way function f. Since K_i is deleted, all data protected by that key can now be considered cryptographically deleted until paired with another device that holds an K_j, $j \leq i$. This idea is closely related to that of forward secrecy.

Secure storage is the most difficult requirement of these. There is data that the device may need to access again and therefore must be able to decrypt. At a theoretical level, that means that the data could just as well have been unencrypted: a key next to a locked chest is effectively an unlocked chest. It becomes slightly more nuanced if one can include time in the picture.

In some sense, the main inspiration is again that of cryptographic deletion. If data only exists in an encrypted form, and the key is destructed, then the data can be considered deleted. In extension, data only comes into existence once the appropriate key is reconstructed.

Correspondingly, our definition does not consider data. Assume that the payload data has been logically structured by some directed graph, and that each part is encrypted with a new key. Identify each key with the corresponding node, so that reaching the node equals knowledge of the key, and hence the data.

Each node also contains information on all outgoing edges. Following the edge should be computationally costly, and preferably non-parallelisable. Concretely, every time the internal state reaches a node one can decrypt the corresponding data packet. It contains one puzzle for each edge, and the key for the other node of the edge can only be found by solving the puzzle. This idea motivates the following definition:

Definition 6. *Let $\mathcal{ES}$ be an encryption scheme. Let $G = (V, E)$ be a graph and $v_i \in V$. For each v_i define $L_i = (v_j, \mathsf{aux}_{v_j})$ as a list of all neighbouring nodes along with auxilliary data for each neighbour. A* Cryptographically Bounded Bandwidth Channel *consists of two algorithms* $(\mathsf{Setup}, \mathsf{RecoverKey})$ *with the following properties:*

Setup *An efficient algorithm that takes in a directed graph $G = (V, E)$ and a delay t, and outputs a related graph $(V', \varnothing)$. For each v_i, generate a key sk_i. Each node $v'_i \in V'$ corresponds to a node $v_i \in V$, and contains an encrypted list of neighbours of v_i, $\{\mathcal{ES}.\mathsf{Enc}(sk_i; L_i)\}$, and the corresponding auxilliary data.*

$\mathsf{RecoverKey}$ *Takes in the auxilliary data aux_{v_j}, and reconstructs the decryption key sk_j for v_j.*

The channel has a weakly (t, p)-bounded output rate if, with probability p, it takes at least tn time to recover n nodes. The channel has strongly (t, p)-bounded output rate if, with probability p, the time to recover each node is at least t.

Informally, view v' as an encrypted version of v and its edges. Implicitly, we assume that G is connected. If G has $n > 1$ components, one can simply consider the case of n instances of the channel.

We now present a simple, informal example to demonstrate how to use this definition. Assume Alice wants to navigate through an area without communicating with others, but wants to keep as much as possible of the map obscured at all times, yet get access every part of the map when necessary.

Divide the map into logical sections, and create a map graph: Each region is a node, and neigbouring regions are connected with edges. Assume that each area takes time t to traverse. Use the graph and time t as input to the setup phase, and use the keys generated in the definition to encrypt each region. Before starting the expedition, delete all keys but one, say sk_1, and set off in region 1.

Alice can see the labeling of the neighbouring regions, and choose one, say 2. In the time she needs to relocate to the next region, she reconstructs the corresponding key, and is able to decrypt the next map section once she reaches the border. She can then choose a new region to move into. Alice must now delete all data from region 1: the map, the key and the reconstruction data for other neighbours of region 1. She can only return to 1 if 1 is a neighbour of 2 (remember, G is a directed graph), or any of the other regions she will explore during her expedition.

This notion can be instantiated using time-lock puzzles [17], whose objective is to delay decryption. The problem is that such problems depend on CPU time (hence, a data center would have a large advantage relative to a resource-constrained device), whereas we would like a scheme that was able to keep the lock until a certain amount of wall-clock time had passed. This is an intrinsically hard problem given that we do not want to allow communication.

The best candidate so far is that of Abadi, Burrows, Manasse and Wobber [1]. In contrast to other candidates, Abadi et al. use memory latency to generate a problem. This number depends on physics and is relatively constant across platforms. The idea is to generate a problem such that the quickest way to solve it repeatedly is by generating a table of all solutions and do lookups for each instance. However, that table should be so large that it cannot fit in cache. For each instance, a new portion of the table must therefore be loaded from memory to cache, causing the delay.

The problem is typically to invert a hash function from a small domain. By repeatedly hashing and xor-ing the preimage with a counter value, one is forced to undo each step on the way. However, it has so far not proven suitable to generate a large key, and so we conclude that further work is needed, while also noting that continuous computations are less than ideal for a resource constrained device. In lack of a sound cryptographic solution, we note that tamper protection plays a crucial role in protecting data at rest.

7 Conclusion

We have presented a selection of security definitions suitable for swarms of devices with limited resources, possibly operating in hostile environments. Given that this is highly relevant for governments, military applications and also commercial interests, our hope is that this paper spurs a strengthened interest for the features modern cryptography has to offer the greater field of cyber security.

We have throughout this work pointed out loose ends and possibilities for further research. We conclude the paper by summarising these.

Forced leave The swarm needs a mechanism to determine which devices to be considered lost, and therefore exclude them. Given that we assume that the adversary might control some of our devices, how can we avoid the adversary trying to blame and exclude an honest (and possible crucial) device? In some settings one might consider special devices with elevated rights. Other times, one have to find a suitable approximation to a solution to the problem of Byzantine agreement.

Anonymity and swarm privacy Can there exist notions that combine efficient multicast with anonymity for any individual device in the network?

Quantum-safe broadcast and multicast Gentry and Waters' asymmetric multicast encryption scheme is based on classic cryptography. A quantum-safe instantiation is an open problem.

Keys and data on the same device Is it possible to delay decryption in such a way that it is at least highly correlated with offline wall-time?

References

1. Martín Abadi, Michael Burrows, Mark S. Manasse, and Ted Wobber. Moderately hard, memory-bound functions. *ACM Trans. Internet Techn.*, 5(2):299–327, 2005. doi:10.1145/1064340.1064341.
2. Raja Naeem Akram, Pierre-François Bonnefoi, Serge Chaumette, Konstantinos Markantonakis, and Damien Sauveron. Secure autonomous UAVs fleets by using new specific embedded secure elements. In *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016*, pages 606–614, 2016. doi:10.1109/TrustCom.2016.0116.
3. Fabio Banfi and Ueli Maurer. Anonymous symmetric-key communication. Cryptology ePrint Archive, Report 2020/073, 2020. https://eprint.iacr.org/2020/073.
4. Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 1–11, Washington, DC, USA, November 18–22, 2002. ACM Press. doi:10.1145/586110.586112.
5. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. doi:10.1109/SFCS.2001.959888.
6. John Chan and Phillip Rogaway. Anonymous AE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 183–208, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-34621-8_7.

7. Jung Hee Cheon, Kyoohyung Han, Seong-Min Hong, Hyoun Jin Kim, Junsoo Kim, Suseong Kim, Hosung Seo, Hyungbo Shim, and Yongsoo Song. Toward a secure drone system: Flying with real-time homomorphic authenticated encryption. *IEEE Access*, 6:24325–24339, 2018. doi:10.1109/ACCESS.2018.2819189.
8. Gianluca Dini and Angelica Lo Duca. A secure communication suite for underwater acoustic sensor networks. *Sensors*, 12(11):15133–15158, 2012. doi:10.3390/s121115133.
9. Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. The role of the adversary model in applied security research. *Computers & Security*, 2018. URL: http://www.sciencedirect.com/science/article/pii/S0167404818306369, doi:https://doi.org/10.1016/j.cose.2018.12.002.
10. Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany. doi:10.1007/3-540-48329-2_40.
11. Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 171–188, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-01001-9_10.
12. Heiko Hamann. *Swarm Robotics - A Formal Approach*. Springer, 2018. doi:10.1007/978-3-319-74528-2.
13. Hugh Harney and Eric J. Harder. Logical Key Hierarchy Protocol. Internet-Draft draft-harney-sparta-lkhp-sec-00, Internet Engineering Task Force, April 1999. Work in Progress. URL: https://datatracker.ietf.org/doc/html/draft-harney-sparta-lkhp-sec-00.
14. Jesse Michel, Sushruth Reddy, Rikhav Shah, Sandeep Silwal, and Ramis Movassagh. Directed random geometric graphs. *Journal of Complex Networks*, 7(5):792–816, 04 2019. arXiv:https://academic.oup.com/comnet/article-pdf/7/5/792/30157011/cnz006.pdf, doi:10.1093/comnet/cnz006.
15. Tal Moran, Ilan Orlov, and Silas Richelson. Topology-hiding computation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 159–181, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_8.
16. Saurabh Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 21–40, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-70936-7_2.
17. Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, 1996.
18. Jaydip Sen. Security in wireless sensor networks. *CoRR*, abs/1301.5065, 2013. URL: http://arxiv.org/abs/1301.5065, arXiv:1301.5065.
19. Martin Strand and Jan Henrik Wiik. Kryptografisk sikring av autonome og ubemannede enheter – eksisterende forskning. FFI-rapport 19/02042, FFI, 2019.
20. Matthew Weidner, Martin Kleppmann, Daniel Hugenroth, and Alastair R. Beresford. Key agreement for decentralized secure group messaging with strong security guarantees. Cryptology ePrint Archive, Report 2020/1281, 2020. https://eprint.iacr.org/2020/1281.