A Game Of Drones: Extending the Dolev-Yao Attacker Model With Movement

Andrew Cook
Department of Informatics
King's College London
London, UK
andrew.cook@kcl.ac.uk

Luca Viganò
Department of Informatics
King's College London
London, UK
luca.vigano@kcl.ac.uk

***Abstract*—We introduce the idea of modelling security protocols and attacks on protocols in a mobile setting, where agents need to move in order to send and receive messages. More specifically, we consider an environment involving drones capable of long-distance flight and "base stations" with which they can communicate. To specify and reason about such an environment, we formalise a pi-calculus and an attacker model that extends the Dolev-Yao model with capabilities for movement and "environmental knowledge" of the position of the other agents. As a simple but illustrative case study, we show how, under certain geographical conditions, the man-in-the-middle attack on the NSPK protocol would require multiple movements to complete successfully. We also define a simple normalisation procedure that allows the attacker to avoid carrying out redundant movements. This procedure also sets the basis for future, more refined normalisations and for investigations into how movements of the attacker and of the honest agents could be optimised.**

***Index Terms*—Security Protocols, Attacker Model, Movement, Nodes, Hubs**

1. Introduction

Context and motivations. The model proposed by Dolev and Yao in [6] has become the standard attacker model in symbolic analysis of security protocols, i.e., when one wants to consider an attacker that can do anything except break cryptography, which is assumed to be perfect. In this paper, we extend the Dolev-Yao attacker model by explicitly considering a mobile setting in which agents need to move in order to send and receive messages.

This is not the first time that agent location and movement have been considered in protocol analysis, so let us explain why our approach is different from what has been done so far. *Mobile Ad-Hoc Networks (MANETs)* that use cryptography are a hot research area (see, e.g., [7]). However, in this paradigm, *routing* is often the preferred tactic. While location and geographical conditions are important to analysing MANET performance [1], in routing protocol attacks such as "black hole" [8], movement of the agents is not considered in the attack trace. Some attacker models in the spirit of the Dolev-Yao model have been developed for MANETs (e.g., [4]), but, to the best of our knowledge, there does not exist a standard, full-fledged, moving Dolev-Yao attacker for MANETs.

Distance-bounding protocols are cryptographic protocols that securely establish an upper bound on the physical distance between the participating agents [2]. In order to symbolically analyse such protocols, a number of security protocol analysis approaches and tools have been extended to consider, in particular, timestamps and the location of agents (see, e.g., [5], [9], [12]). While there are obviously tight connections (that we will investigate in more detail in the future), the attacker movement that we consider here is quite different from that usually considered for distance-bounding protocols.

Our work was motivated by *Wireless Body Area Networks (WBANs)*, which, thanks to miniaturised electrical invasive/non-invasive devices, allow for continuous health monitoring of patients. A number of protocols have been put forth for WBANs for desired levels of authentication, e.g., [11]. Analyses on these protocols have revealed key-compromise attacks, but do not consider how the mobility of the agents affects the possibility of attacking even the cryptography-less security protocols in the standard [13]. More specifically, the analyses of WBAN protocols carried out so far, including our yet unpublished one, only marginally looked at movement explicitly. Hence, in this paper, we begin a foundational study of the security analysis of protocols in which agents explicitly need to move to send and receive messages; for instance, in a WBAN a doctor will need to move close to a patient in order to read the health data collected by the patient's implanted sensors.

In addition to movement, we observed that the *environment* in which security protocols are carried out (i.e., the location of the agents participating in the protocol) is typically not considered at the symbolic level. This contrasts with other areas of security such as in *cyber-physical systems*, where attacks on the sensors in the environment influence the content of the messages sent and received by the PLC or SCADA system. Certain environmental conditions must be in place for a given protocol to be *executable*; for instance, that two agents must be near enough to each other in order to communicate over a channel with finite range (assuming routing techniques are not used). Given this, we can infer that the attacker must be also influenced by the environment. Therefore the attacker's abilities must be somewhat reflected by the environmental conditions under which some protocol is

being executed, similarly to the honest agents themselves.

Contributions. This paper reports on the foundational work that we have been carrying out so far in formalising an automated approach in which protocols are analysed in the presence of a Dolev-Yao attacker with explicit movement. More specifically, we define a two-dimensional environment consisting of both moving and static agents. *Nodes* with a circular range of influence are static in our environment, and mobile point-like entities called *Hubs* freely move in the environment in and out of range of the circular nodes. A node's two-dimensional position is explicitly defined, whereas a hub's position is only defined by the nodes it is in range of. This saves us having to delimit the environment into any specific resolution and define any function of movement over the coordinate space. Instead, we can define movements as timed processes in an extended pi-calculus. We also define an extended Dolev-Yao attacker that takes the form of a node that can dispatch infinite hubs at will, to move and collaborate with honest agents in the environment. We can then build traces that show how the movement of the agents and of the attacker significantly affects the protocol execution and attack execution.

As a simple but illustrative case study, we consider a version of the *Needham-Schroeder public key protocol* where agents have to move to send each other messages. We show how, under certain geographical conditions, the man-in-the-middle attack on the NSPK protocol would require multiple movements to complete successfully.

However, the attacker may wander aimlessly during the execution of a protocol attack, making redundant movements that ultimately increase the total time and effort for the attack. We introduce an initial, simple normalisation procedure that rests on rules that transform an attack trace by removing redundant movement. This yields a *normal attack trace*. This procedure also sets the basis for future, more refined normalisations and for investigations into how movements of the attacker and of the honest agents could be optimised.

Organisation. In Section 2, we define the environment of nodes that we consider. In Section 3, we formalise an extended pi-calculus to reason about protocols in which honest agents and the attacker need to move to interact. In Section 4, we consider, as an example, a version of the NSPK protocol with movement. In Section 5, we define a normalisation procedure that allows the attacker to avoid redundant movements. In Section 6, we draw conclusions and discuss future work. The appendix contains additional material that is helpful to understand our results.

2. The Environment of Nodes

We assume that the environment of nodes is an affine plane and that the broadcast range of each node describes a circle. Hubs move through the environment and can associate (resp., disassociate) with nodes by moving inside (resp., outside) their broadcast ranges, and begin mutual communication. Unlike more conventional models of wireless networks, our model only implements hub to node communication. Nodes do not communicate with nodes and likewise hubs do not communicate with hubs.

To reduce the number of possible arrangements of nodes in the environment and thus reduce the number of calculus rules, we adopt the assumptions given in [14] for the "number of arrangements of n circles in the affine plane". As illustrated by the examples in Figure 1, we assume that a node's broadcast circumference cannot have exactly 1 point of intersection and, while three or more nodes may all overlap, there cannot be a *point of intersection* that belongs to more than 2 broadcast circumferences.[1]

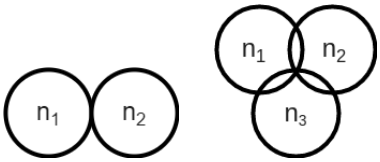

Figure 1: Disallowed "touching" broadcast ranges (left) and disallowed intersecting points (right)

We also assume that for every time step in the protocol's trace, a hub can either stay in an area enclosed by the circles, or cross a *single* boundary. We count a point of intersection as a single boundary that the hubs may cross.

With the current assumptions, the points of intersection of the circles only belong to at most two circles. Hence, a node can be disjoint from all other nodes, inside another larger node, or overlapping other nodes such that no point on their circumference belongs to more than 1 other node. Therefore, the hub can only gain or lose at most 2 in-band nodes at each time step.

Even under these assumptions, the number of possible geographies of nodes combinatorically explodes as the number of nodes increases [14].

3. The Calculus

In this section, we formalise our pi-calculus that extends the Dolev-Yao attacker model with movement, and provides an operational semantics for each agent. The rules themselves model what the nodes, hubs, and attackers can do. The behaviour of each agent, according to a protocol, is given by the respective rules.

3.1. Messages

Our approach is independent of the specific language of messages, but for concreteness we consider the standard operators for message pairing (m_1, m_2), or simply m_1, m_2, symmetric encryption $\{|m_1|\}_{m_2}$ and asymmetric encryption $\{m_1\}_{m_2}$. Other operators (e.g., hashing, exponentiation, XOR) can be added easily.

As usual, for a set M of messages, we define $\mathcal{DY}(M)$ to be the smallest set closed under the *generation (G)* and *analysis (A)* Dolev-Yao-style rules given in Figure 2. Other rules can be added as necessary.

3.2. Time

We model a *global, linear clock* to give the calculus and the protocols a notion of time. The clock is a global

1. The reason for the second assumption is that to geometrically express this kind of intersection by using only the centre points of the three circles and their radii requires algebraic reasoning and cannot be expressed with a simple first-order formula.

$$\frac{m \in M}{m \in \mathcal{DY}(M)}\ G_{axiom} \qquad \frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{(m_1, m_2) \in \mathcal{DY}(M)}\ G_{pair}$$

$$\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{|m_1|\}_{m_2} \in \mathcal{DY}(M)}\ G_{scrypt} \qquad \frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{m_1\}_{m_2} \in \mathcal{DY}(M)}\ G_{crypt}$$

$$\frac{(m_1, m_2) \in \mathcal{DY}(M)}{m_i \in \mathcal{DY}(M)}\ A_{pair_i} \qquad \frac{\{|m_1|\}_{m_2} \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{m_1 \in \mathcal{DY}(M)}\ A_{scrypt}$$

$$\frac{\{m_1\}_{m_2} \in \mathcal{DY}(M) \quad inv(m_2) \in \mathcal{DY}(M)}{m_1 \in \mathcal{DY}(M)}\ A_{crypt} \qquad \frac{\{m_1\}_{inv(m_2)} \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{m_1 \in \mathcal{DY}(M)}\ A^{inv}_{crypt}$$

Figure 2: Message Generation and Analysis Rules of the Attacker

variable t that annotates each process and knowledge set, and is incremented by the function $inc(t)$. The *agents' knowledge* evolves over time as new messages are received (or freshly created); we write K_x^t for the knowledge of the hub, node, or attacker x at global time t.

We assume that messages sent or received by hubs and nodes take no time. Associations and disassociations of hubs and nodes are solely responsible for time passing. Hubs can advance the clock by simply "lurking" (more later) within an area enclosed by node broadcast ranges, or by crossing a boundary of a node or point of intersection of nodes. We write p^t, where p is a process representing the trace of a node, hub, or attacker, to mean the "state" of the process p at global time t.

3.3. Nodes

Let $\mathcal{N}$ be a set of triples describing the arrangement of nodes in the environment. For $0 \leq j \leq |\mathcal{N}|$, each node $n_j = (c_j, (x_j, y_j), r_j)$ is a triple with respective projection functions id, crd and rng, where c_j is the node's name, (x_j, y_j) are its fixed coordinates, and r_j is its finite radial broadcast range. The name $c_j = id(n_j)$ is the node's *default* channel (the radio frequency, Wi-Fi name or other means it will transmit messages on unless other channels are given to it).[2] This name c_j is available to all hubs who move inside r_j, as we will discuss later.

All nodes in the environment are themselves modelled as processes with a set of capabilities. We define a finite set of process names $\{\mathsf{n}_0, \mathsf{n}_1, \ldots\}$, where $|\{\mathsf{n}_0, \mathsf{n}_1, \ldots\}| = |\mathcal{N}|$ and $0 \leq p, q \leq |\{\mathsf{n}_0, \mathsf{n}_1, \ldots\}|$, described by the grammar:

$\mathsf{n}_p, \mathsf{n}_q ::=$	$U(x).\mathsf{n}_p$	input
	$\mid \overline{U}\langle V\rangle.\mathsf{n}_p$	output
	$\mid \mathsf{n}_p \parallel \mathsf{n}_q$	composition
	$\mid \tau.\mathsf{n}_p$	internal
	$\mid 0$	halt

$U(x).\mathsf{n}_p$ is the action of receiving a variable x on a channel U and evolving into process n_p, and $\overline{U}\langle V\rangle.\mathsf{n}_p$ is the action of sending V over a channel U before evolving into n_p.

2. These channels are insecure, so the nodes may wish to upgrade their default channel to an authentic, confidential or secure channel, or establish a new channel with certain properties. How approach straightforwardly extends to cover these cases.

Nodes are simple entities in our calculus that can read and write messages on channels as specified by the rules $Read_{\mathsf{n}}$ and $Write_{\mathsf{n}}$:

$$\frac{}{c_n(m).\mathsf{n}_p^t \xrightarrow{c_n(m)} \mathsf{n}_p^t}\ Read_{\mathsf{n}}$$

$$\frac{m \in \mathcal{DY}(K^t_{\mathsf{n}_p}) \quad c_n \in \mathcal{DY}(K^t_{\mathsf{n}_p})}{\overline{c_n}\langle m\rangle.\mathsf{n}_p^t \xrightarrow{\overline{c_n}\langle m\rangle} \mathsf{n}_p^t}\ Write_{\mathsf{n}}$$

The $Read_{\mathsf{n}}$ rule has no premises: at any time instant t, a node is able to receive a message m on any channel c_n. As explained later: (i) the environmental preconditions required for m to be received are taken care of on the hub's end, (ii) upon receiving a message m on channel c_n, both c_n and m are added to that node's knowledge. The $Write_{\mathsf{n}}$ rule allows the node to send any message m from its current knowledge on some channel c_n in that knowledge.

3.4. Hubs

In addition to their "normal" knowledge, hubs have an environmental knowledge set, which is a set of nodes whose ranges the hub is within. We thus define a set $\mathcal{H} = \{\mathsf{h}_0, \mathsf{h}_1, \ldots\}$ of *hub process names* and a set of corresponding *environmental knowledge sets* $\mathcal{E} = \{E_{\mathsf{h}_0}, E_{\mathsf{h}_1}, \ldots, E_{\mathsf{h}_{|\mathcal{H}|}}\}$.

Hence, while both nodes and hubs can derive messages to send on derived channels from their knowledge sets, only hubs are in possession of an extra knowledge set; to whom they can send a message. This gives rise to an implicit *position* of the hub in the environment. Thus, "the movement" of a hub in the environment is expressed as a change of the hub's environmental knowledge set, based on the relationship between the nodes' coordinates and ranges.

For $0 \leq p, q \leq |\mathcal{H}|$, the hub processes are defined by the following grammar, where the action $lurk$ advances the global clock without moving:

$\mathsf{h}_p, \mathsf{h}_q ::=$	$U(x).\mathsf{h}_p$	input
	$\mid \overline{U}\langle V\rangle.\mathsf{h}_p$	output
	$\mid \mathsf{h}_p \parallel \mathsf{h}_q$	composition
	$\mid \tau.\mathsf{h}_p$	internal
	$\mid lurk.\mathsf{h}_p$	lurk
	$\mid 0$	halt

Next, we define movement as an extension to the abilities of the node, that remains static. We assume that the hubs have full information about the environment and freely choose how to move.

3.4.1. Movement Semantics. The position of a hub h at time t is described by the set $E^t_{\mathsf{h}} \subseteq \mathcal{N}$ of in-band nodes. Every time step the hub can move around the environment and change its E^t_{h}, giving rise to a trace over the global clock t. As mentioned above, we assume that sending and receiving messages takes no time (does not advance the clock).

For the movement transition semantics, we need ways to formalise the hubs' interactions with any arbitrary geography of nodes. To formalise the assumptions on the environment in Section 2, we use two first-order formulas:

$$M(x) = \neg\exists y \in \mathcal{N}. \\ |crd(x) - crd(y)| \ + \ rng(x) < rng(y)\,,$$

which allows us to test that the broadcast range of node $x \in \mathcal{N}$ is *not* encircled by another node $y \in \mathcal{N}$ (if x was consumed by y, a hub could not associate with x without first associating with y) and

$$O(x,y) \ = \ |crd(x) - crd(y)| \ < \ rng(x) + rng(y)\,,$$

which allows us to test if two nodes $x, y \in \mathcal{N}$ are overlapping in their broadcast ranges.

Figure 3 shows the movement rules for a hub. In order to follow the location of the hubs and the nodes they interact with as we build a trace, we parameterise each rule by the nodes involved in the movement ($n_j, n_k \in \mathcal{N}$). A and D are for association and disassociation respectively, OA and OD are reserved for single node association and disassociation from nodes that overlap with other nodes.

The rules A_{single} and D_{single} represent the action of the hub moving into / out of range of a single node n_j. We check that n_j is not surrounded by another node with our formula $M(n_j)$, as due to our two-dimensional environment, a hub would have to enter the range of the surrounding node first, before entering the range of n_j. The rules OA_{single} and OD_{single} model a node moving into / out of range of a node n_j that overlaps with another node n_k. Lastly, A_{double} and D_{double} model the behaviour of the hub moving into the overlapping region formed by two overlapping nodes.

3.4.2. Process Semantics. The read and write rules for the hubs are like those of the nodes, but hubs can also lurk:

$$\frac{}{c_n(m).\mathsf{h}^t_p \xrightarrow{c_n(m)} \mathsf{h}^t_p}\ Read_{\mathsf{h}}$$

$$\frac{m \in \mathcal{DY}(K^t_{\mathsf{h}_p}) \quad c_n \in \mathcal{DY}(K^t_{\mathsf{h}_p})}{\overline{c_n}\langle m\rangle.\mathsf{h}^t_p \xrightarrow{\overline{c_n}\langle m\rangle} \mathsf{h}^t_p}\ Write_{\mathsf{h}}$$

$$\frac{}{lurk.\mathsf{h}^t_p \xrightarrow{lurk} \mathsf{h}^{inc(t)}_p}\ Lurk$$

where $0 \le p \le |\mathcal{H}|$.

3.5. Attacker

As is often the case, we denote the attacker using lowercase and uppercase "i" for "*intruder*". We assume the attacker is made up of a single node and an unbounded set of hubs that can dispatch from the node. We take i^{n} as a *node-behaving attacker process* and i^{h} for a *hub-behaving attacker process*, inheriting all the abilities of nodes and hubs, respectively. Hence, we define a corresponding *attacker node triple* I and a set $\mathcal{A}_{hub} = \{\mathsf{i}^{\mathsf{h}}_0, \ldots, \mathsf{i}^{\mathsf{h}}_{|\mathcal{A}_{hub}|}\}$ of attacking hub processes. For the attacking hubs, we need *environmental knowledge sets* $\tilde{\mathcal{E}} = \{\tilde{E}_{\mathsf{i}^{\mathsf{h}}_0}, \ldots, \tilde{E}_{\mathsf{i}^{\mathsf{h}}_{|\mathbb{A}_{hub}|}}\}$.

In addition to inheriting the capabilities of nodes and hubs as specified by their rules (the process names in the rules are substituted with either i^{n} or i^{h}), we model that attacking nodes can dispatch any number of attacking hubs that are required (rule $Disp$). Upon the dispatching of an attacking hub, the attacking node has the extra ability of restricting a channel with which they can communicate, so long as the attacking hub is in range of the attacking node (rule Res):

$$\frac{\mathsf{i}^{\mathsf{n}^t}_I}{disp.\mathsf{i}^{\mathsf{n}^t}_I \xrightarrow{disp} \mathsf{i}^{\mathsf{n}^t}_I \parallel \mathsf{i}^{\mathsf{h}^t}_p}\ Disp \qquad \frac{\mathsf{i}^{\mathsf{n}^t}_I \parallel \mathsf{i}^{\mathsf{h}^t}_p}{(\nu c_{II})\mathsf{i}^{\mathsf{n}^t}_I \parallel \mathsf{i}^{\mathsf{h}^t}_p}\ Res$$

where $0 \le p \le |\mathcal{A}_{Hub}|$.

3.6. Bridge Rules

When we build a trace of a protocol, there are some transitions that we leave implicit, particularly the message construction rules. However, we need to create *bridge rules* that allow the processes to traverse from the world of the implicit to the explicit.

The following rules represent bridge rules between a node/hub/attacker process p and the Dolev-Yao knowledge. We need a rule to allow the addition of a new message to the knowledge upon receiving it, a rule to allow the sending of a known message on a channel that the process also knows, and a rule for a process to create new messages (e.g., nonces):

$$\frac{c_n(m).\mathsf{p}^t \xrightarrow{c_n(m)} \mathsf{p}^t}{m \in K^t_{\mathsf{p}}}\ ReadMsg$$

$$\frac{m \in \mathcal{DY}(K^t_{\mathsf{p}}) \quad c_n \in \mathcal{DY}(K^t_{\mathsf{p}})}{\overline{c_n}\langle m\rangle.\mathsf{p}^t \xrightarrow{\overline{c_n}\langle m\rangle} \mathsf{p}^t}\ WriteMsg$$

$$\frac{\nu(m).\mathsf{p}^t \xrightarrow{\nu(m)} \mathsf{p}^t}{m \in K^t_{\mathsf{p}}}\ FreshMsg$$

Finally, to draw a connection between the environment and the messages, the rule

$$\frac{n_j \in E^t_{\mathsf{h}}}{id(n_j) \in K^t_{\mathsf{h}}}\ NewAuth(n_j)$$

models an oracle that provides a(n attacking) hub the default name of a node as soon as it is within range of it.

$$\frac{M(n_j) \quad n_j \notin E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \cup \{n_j\}}\ A_{single}(n_j) \qquad \frac{M(n_j) \quad n_j \in E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \setminus \{n_j\}}\ D_{single}(n_j)$$

$$\frac{M(n_j) \quad O(n_j, n_k) \quad n_j \notin E_{\mathsf{h}}^t \quad n_k \notin E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \cup \{n_j\}}\ OA_{single}(n_j, (n_j, n_k))$$

$$\frac{M(n_j) \quad O(n_j, n_k) \quad n_j \in E_{\mathsf{h}}^t \quad n_k \notin E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \setminus \{n_j\}}\ OD_{single}(n_j, (n_j, n_k))$$

$$\frac{M(n_j) \quad O(n_j, n_k) \quad n_j \in E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \cup \{n_k\}}\ A_{cross}(n_j, n_k) \qquad \frac{M(n_j) \quad O(n_j, n_k) \quad n_j \in E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \setminus \{n_j\}}\ D_{cross}(n_j, n_k)$$

$$\frac{M(n_j) \quad M(n_k) \quad O(n_j, n_k) \quad n_j \notin E_{\mathsf{h}}^t \quad n_k \notin E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \cup \{n_j, n_k\}}\ A_{double}(n_j, n_k)$$

$$\frac{M(n_j) \quad M(n_k) \quad O(n_j, n_k) \quad n_j \in E_{\mathsf{h}}^t \quad n_k \in E_{\mathsf{h}}^t}{E_{\mathsf{h}}^{inc(t)} = E_{\mathsf{h}}^t \setminus \{n_j, n_k\}}\ D_{double}(n_j, n_k)$$

Figure 3: Movement Semantics Rules

3.7. Synchronicity Rules

Since agents may carry out actions independently of other agents, we provide rules for synchronicity among agents.

3.7.1. Hubs and Nodes. The following rules allow the hubs and nodes in the environment to operate independently of each other:

$$\frac{\lambda.\mathsf{h}_q^t \xrightarrow{\lambda} \mathsf{h}_q^t}{\mathsf{p}^t \parallel \lambda.\mathsf{h}_q^t \xrightarrow{\lambda} \mathsf{p}^t \parallel \mathsf{h}_q^t}\ Hub_{msgsync}$$

$$\frac{\lambda.\mathsf{n}_q^t \xrightarrow{\lambda} \mathsf{n}_q^t}{\mathsf{p}^t \parallel \lambda.\mathsf{n}_q^t \xrightarrow{\lambda} \mathsf{p}^t \parallel \mathsf{n}_q^t}\ Node_{msgsync}$$

$$\frac{\phi.\mathsf{h}_q^t \xrightarrow{\phi} \mathsf{h}_q^t}{\mathsf{p}^t \parallel \phi.\mathsf{h}_q^t \xrightarrow{\phi} \mathsf{p}^t \parallel \mathsf{h}_q^t}\ Hub_{movsync}$$

where $0 \leq q \leq |\mathcal{H}|$, where λ ranges over sending and receiving, ϕ ranges over the names of the movement rules, and p represents any currently running process.

3.7.2. Attacking Hubs and Nodes. In addition to the rules for nodes and hubs inherited by the attacker, attacking nodes are able to dispatch colluding attacking hubs at will at any time, so we need a set of rules that allow attacking nodes and hubs to communicate internally (with the internal action τ) over a restricted channel:

$$\frac{\lambda.\mathsf{i}_p^{\mathsf{h}^t} \xrightarrow{\lambda} \mathsf{i}_p^{\mathsf{h}^t}}{\mathsf{p}^t \parallel \lambda.\mathsf{i}_p^{\mathsf{h}^t} \xrightarrow{\lambda} \mathsf{p}^t \parallel \mathsf{i}_p^{\mathsf{h}^t}}\ AHub_{msgsync}$$

$$\frac{\lambda.\mathsf{i}_I^{\mathsf{n}^t} \xrightarrow{\lambda} \mathsf{i}_I^{\mathsf{n}^t}}{\mathsf{p}^t \parallel \lambda.\mathsf{i}_I^{\mathsf{n}^t} \xrightarrow{\lambda} \mathsf{p}^t \parallel \mathsf{i}_I^{\mathsf{n}^t}}\ ANode_{msgsync}$$

$$\frac{\phi.\mathsf{i}_p^{\mathsf{h}^t} \xrightarrow{\phi} \mathsf{i}_p^{\mathsf{h}^t}}{\mathsf{p}^t \parallel \phi.\mathsf{i}_p^{\mathsf{h}^t} \xrightarrow{\phi} \mathsf{p}^t \parallel \phi.\mathsf{i}_p^{\mathsf{h}^t}}\ AHub_{movsync}$$

$$\frac{c_n(m).\mathsf{i}_p^{\mathsf{h}^t} \xrightarrow{c_n(m)} \mathsf{i}_p^{\mathsf{h}^t} \quad \overline{c_n}\langle m\rangle.\mathsf{i}_I^{\mathsf{n}^t} \xrightarrow{\overline{c_n}\langle m\rangle} \mathsf{i}_I^{\mathsf{n}^t}}{\mathsf{i}_p^{\mathsf{h}^t} \parallel \mathsf{i}_I^{\mathsf{n}^t} \xrightarrow{\tau} \mathsf{i}_p^{\mathsf{h}^t} \parallel \mathsf{i}_I^{\mathsf{n}^t}}\ ANode_{intercom}$$

$$\frac{\overline{c_n}\langle m\rangle.\mathsf{i}_p^{\mathsf{h}^t} \xrightarrow{\overline{c_n}\langle m\rangle} \mathsf{i}_p^{\mathsf{h}^t} \quad c_n(m).\mathsf{i}_I^{\mathsf{n}^t} \xrightarrow{c_n(m)} \mathsf{i}_I^{\mathsf{n}^t}}{\mathsf{i}_p^{\mathsf{h}^t} \parallel \mathsf{i}_I^{\mathsf{n}^t} \xrightarrow{\tau} \mathsf{i}_p^{\mathsf{h}^t} \parallel \mathsf{i}_I^{\mathsf{n}^t}}\ AHub_{intercom}$$

where $0 \leq p \leq |\mathcal{A}_{Hub}|$, λ ranges over sending and receiving, and ϕ ranges over the names of the movement rules.

3.7.3. Time Synchronicity of Hubs. Hubs are the only agents that are able to move in the environment, thus are the only agents that are able to advance the global clock through movement. There is likely to be more than one hub in the environment moving at the same time (e.g., if two hubs are moving towards the same node n, they should be able to move inside the range of n at the same time).

To represent the synchronicity of movement among hubs, we consider the evolution as a silent action as it can be thought of as an internal process within the environment:

$$\frac{\phi.\mathsf{h}_p^t \xrightarrow{\phi} \mathsf{h}_p^{inc(t)} \quad \phi.\mathsf{h}_q^t \xrightarrow{\phi} \mathsf{h}_q^{inc(t)}}{\mathsf{h}_p^t \parallel \mathsf{h}_q^t \xrightarrow{\tau} \mathsf{h}_p^{inc(t)} \parallel \mathsf{h}_q^{inc(t)}}\ Hub_{intermov}$$

where $0 \leq p, q \leq |\mathcal{H}|$ and ϕ ranges over the names of the movement rules.

3.8. Dispatching Knowledge Rules

When an attacking hub is dispatched by the attacking node, the default channels of all the nodes who's broadcast ranges encircle the attacking node are put immediately into the attacking hub's knowledge set. As these nodes encircle the attacking node, the newly dispatched attacking hub can clearly immediately begin communicating with them. The following rule updates the dispatched attacking hub's environmental information:

$$\frac{disp.\mathsf{i}_I^{\mathsf{n}^t} \xrightarrow{disp} (\mathsf{i}_I^{\mathsf{n}^t} \parallel \mathsf{i}_p^{\mathsf{h}^t})}{id(y) \in K^t_{\mathsf{i}^{\mathsf{h}}_p}} \; Hub_{update}$$

where y is such that $|crd(I) - crd(y)| + rng(I) < rng(y)|$ and $0 \leq p \leq |\mathcal{A}_{Hub}|$.

4. A Concrete Example: Full NSPK With Movement

Let us consider the Needham-Schroeder Public-Key Protocol between two agents A and B and a trusted third party S,

$$\begin{array}{ll} A \to S: & A, B \\ S \to A: & \{PK_B, B\}_{inv(PK_S)} \\ A \to B: & \{N_A, A\}_{PK_B} \\ B \to S: & B, A \\ S \to B: & \{PK_A, A\}_{inv(PK_S)} \\ B \to A: & \{N_A, N_B\}_{PK_A} \\ A \to B: & \{N_B\}_{PK_B} \end{array}$$

and let us assume, for the sake of example, that this protocol explicitly includes movement, i.e., agents have to move to be able to send messages to each other.

There are different environments that we can consider, the simplest one being the one in which A, B and S are all far apart. We first identify the types of processes required to be present in the environment in order for the protocol to be executable:

- A is an initiator, so it must be a hub,
- S is always a responder, so it must be a node, and
- B plays both roles in this protocol, so it must be a node B^{n} and a dispatched hub B^{h} working together.

Figure 8 shows the execution trace of NSPK with no attacker in the simple environment.

Let us now imagine what an attacker might be able to do. Figure 9 shows the execution trace of Lowe's attack in the simple environment.[3] Ironically, because the nodes are all far away from each other in this environment, the job of the attacker is more difficult than in the case of the "normal" NSPK without movement. This attack requires a lot of travelling (13 steps), but we can do better.

3. The attacking node and its dispatched hubs are written as processes in parallel for brevity. For the attacking node process and each dispatched attacking hub process, nothing is executed unless clearly stated in the trace.

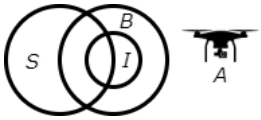

Figure 4: An environment in which the NSPK attack only requires three movements

Take, for example, the environment in Figure 4. In this environment, the attack only requires 3 movements as shown in Figure 10: A moves into the region where S and B overlap, and sends the first message A, I to S and receives back $\{PK_I, I\}$ signed by S. A then moves into range of I, and sends $\{N_A, A\}_{PK_I}$. The attacking node I dispatches an attacking hub, a process which takes no time, and subsequently moves through the environment to go retrieve B's public key, with time passing upon each movement. Crucially in this example though, the attacking hub does not have to move as it is already inside B. Therefore, the attacking hub instantaneously forwards the message $\{N_A, A\}_{PK_B}$ to B. B passes this message onto the co-operating hub B^{h}, who then performs the third and final movement into S to retrieve A's public key.

This has an interesting implication: given the specifications of the protocol and of the environment, there exists a notion of a "normal" attack, i.e., a more efficient attack, with no redundant movements.

5. Attack Trace Normalisation

Movement and message passing are intrinsically connected; for a hub to talk to a node it must move within range. However, an attacking hub may wander aimlessly during the execution of a protocol attack, making redundant movements that ultimately increase the total time and effort for the attack. We introduce a normalisation procedure that rests on rules that transform an attack trace τ into another attack trace τ' by removing redundant movements, in symbols $\tau \rightsquigarrow \tau'$. Removing all redundant movements that an attacking hub could make yields a *normal attack trace*.

5.1. Normalisation Rules

Definition 5.1 (Redundant Association). If a hub enters and leaves the range of a node without sending or receiving a message, then we can remove these movements using the rules

$A_{single}(N).\pi.D_{single}(N).\mathsf{i}^{\mathsf{h}} \rightsquigarrow \mathsf{i}^{\mathsf{h}}$

$A_{double}(N, M).\pi.D_{double}(N, M).\mathsf{i}^{\mathsf{h}} \rightsquigarrow \mathsf{i}^{\mathsf{h}}$

$OA_{single}(N, (N, M)).\pi.OD_{single}(N, (N, M)).\mathsf{i}^{\mathsf{h}} \rightsquigarrow \mathsf{i}^{\mathsf{h}}$

where the possibly empty sub-trace π does not contain any send action $\overline{U}\langle V \rangle$ or receive action $U(x)$.

Definition 5.2 (Redundant Disassociation). If a hub leaves a node and returns to it without sending or receiving a message, the hub could have stayed where it was and we can remove these movements using the rules

$D_{single}(N).\pi.A_{single}(N).\mathsf{i}^{\mathsf{h}} \rightsquigarrow \mathsf{i}^{\mathsf{h}}$

$D_{double}(N, M).\pi.A_{double}(N, M).\mathsf{i}^{\mathsf{h}} \rightsquigarrow .\mathsf{i}^{\mathsf{h}}$

$OD_{single}(N, (N, M)).\pi.OA_{single}(N, (N, M)).\mathsf{i}^{\mathsf{h}} \rightsquigarrow \mathsf{i}^{\mathsf{h}}$

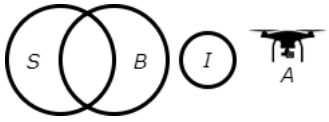

Figure 5: A possible environment for NSPK

where the possibly empty sub-trace π does not contain any send action $\overline{U}\langle V\rangle$ or receive action $U(x)$.

5.2. Normalisation Procedure

It is not difficult to see that the environment influences the effort required of the attacker. Consider NSPK with the environment in Figure 5 in which the trusted third party S and the honest node B are overlapping, and A and the attacker are initially placed far away from B and S. The attacker can choose how to move to carry out his attack, but his choices might result in redundant movements in the subtrace of the attacking hub as, e.g., in the trace in Figure 6.[4] Twice, the attacking hub performs redundant movements. The first case

$$(\mathsf{i}_I^{\mathsf{n}} \parallel D_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{I\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OA_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{S\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OD_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{S\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel A_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{I\}}$$

is a redundant disassociation from I (the move is redundant because the attacking hub joins S without sending or receiving any message). Recall that, for brevity, we write the actions of the attacking node $\mathsf{i}_I^{\mathsf{n}}$ and attacking hub $\mathsf{i}_0^{\mathsf{h}}$ in parallel but they are sequential in the fully expanded trace. This means that we can "override" the $\parallel$ to apply our normalisation rules in Definition 5.2 to reduce this subtrace to $(\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}})$.

The second case

$$(\mathsf{i}_I^{\mathsf{n}} \parallel OA_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{S\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel A_{single}(B).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{B\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel D_{single}(B).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{B\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OD_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{S\}}$$

is a redundant association to S that we can eliminate with the rules in Definition 5.1 to obtain again $(\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}})$. Normalisation thus produces the attack trace in Figure 7.

We devised two algorithms to eliminate all redundant movements in an attack trace. Algorithm 1 removes redundant associations from a trace, which it treats as a list of actions. The algorithm tail-recursively inspects the input trace τ. Whenever the algorithm reaches a redundancy, it skips that part of the trace and does not add those

4. Since we are only interested in attack normalisation, we only consider the trace of the attacker, and (for now) we assume that the behaviour of the other protocol participants does not change. For readability, we also ignore annotating the global clock.

i^{n} :

$$c_I(\{N_A,A\}_{PK_I}).\mathsf{i}_I^{\mathsf{n}} \xrightarrow{\{N_A,A\}\in K_{\mathsf{i}_I^{\mathsf{n}}}}$$
$$disp.\mathsf{i}_I^{\mathsf{n}} \xrightarrow{disp}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel Hub_{update}.\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{c_I\in K_{\mathsf{i}_0^{\mathsf{h}}}}$$
$$res.(\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\nu c_{II}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel D_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{I\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OA_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{S\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OD_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{S\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel A_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{I\}}$$
$$(\overline{c_{II}}\langle\{N_A,PK_A\}\rangle.\mathsf{i}_I^{\mathsf{n}} \parallel c_{II}(\{N_A,PK_A\}).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tau}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel D_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{I\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OA_{single}(B,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{B\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel NewAuth(B).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{c_B\in K_{\mathsf{i}_0^{\mathsf{h}}}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel \overline{c_B}\langle\{N_A,A\}_{PK_B}\rangle.\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\overline{c_B}\langle\{N_A,A\}_{PK_B}\rangle}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel c_B(\{N_A,N_B\}_{PK_A}).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\{N_A,N_B\}_{PK_A}\in K_{\mathsf{i}_0^{\mathsf{h}}}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OD_{single}(B,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{B\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OA_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{S\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel A_{single}(B).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{B\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel D_{single}(B).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{B\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OD_{single}(S,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{S\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel A_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{I\}}$$
$$(c_{II}(\{N_A,N_B\}).\mathsf{i}_I^{\mathsf{n}} \parallel \overline{c_{II}}\langle\{N_A,N_B\}\rangle.\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tau}$$
$$(\overline{c_I}\langle\{N_A,N_B\}_{PK_A}\rangle.\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\overline{c_I}\langle\{N_A,N_B\}_{PK_A}\rangle}$$
$$(c_I(\{N_B\}_{PK_B}).\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}}) \xrightarrow{N_B\in K_{\mathsf{i}_I^{\mathsf{n}}}}$$
$$(\overline{c_{II}}\langle\{N_B\}_{PK_B}\rangle.\mathsf{i}_I^{\mathsf{n}} \parallel c_{II}(\{N_B\}_{PK_B}).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tau}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel D_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\setminus\{I\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel OA_{single}(B,(S,B)).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)}=\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t}\cup\{B\}}$$
$$(\mathsf{i}_I^{\mathsf{n}} \parallel \overline{c_B}\langle\{N_B\}_{PK_B}\rangle.\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\overline{c_B}\langle\{N_B\}_{PK_B}\rangle} 0$$

Figure 6: Attack Trace With Redundancies for the Environment in Figure 5

i^{n} :

$$
\begin{aligned}
&c_I(\{N_A, A\}_{PK_I}).\mathsf{i}^{\mathsf{n}}_I \xrightarrow{\{N_A,A\}\in K_{\mathsf{i}^{\mathsf{n}}_I}} \\
&disp.\mathsf{i}^{\mathsf{n}}_I \xrightarrow{disp} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel Hub_{update}.\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{c_I\in K_{\mathsf{i}^{\mathsf{h}}_0}} \\
&res.(\mathsf{i}^{\mathsf{n}}_I \parallel \mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\nu c_{II}} \\
&(\overline{c_{II}}\langle\{N_A, PK_A\}\rangle.\mathsf{i}^{\mathsf{n}}_I \parallel c_{II}(\{N_A, PK_A\}).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tau} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel D_{single}(I).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tilde{E}^{inc(t)}_{\mathsf{i}^{\mathsf{h}}_0}=\tilde{E}^{t}_{\mathsf{i}^{\mathsf{h}}_0}\setminus\{I\}} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel OA_{single}(B,(S,B)).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tilde{E}^{inc(t)}_{\mathsf{i}^{\mathsf{h}}_0}=\tilde{E}^{t}_{\mathsf{i}^{\mathsf{h}}_0}\cup\{B\}} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel NewAuth(B).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{c_B\in K_{\mathsf{i}^{\mathsf{h}}_0}} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel \overline{c_B}\langle\{N_A, A\}_{PK_B}\rangle.\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\overline{c_B}\langle\{N_A,A\}_{PK_B}\rangle} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel c_B(\{N_A, N_B\}_{PK_A}).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\{N_A,N_B\}_{PK_A}\in K_{\mathsf{i}^{\mathsf{h}}_0}} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel OD_{single}(B,(S,B)).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tilde{E}^{inc(t)}_{\mathsf{i}^{\mathsf{h}}_0}=\tilde{E}^{t}_{\mathsf{i}^{\mathsf{h}}_0}\setminus\{B\}} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel A_{single}(I).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tilde{E}^{inc(t)}_{\mathsf{i}^{\mathsf{h}}_0}=\tilde{E}^{t}_{\mathsf{i}^{\mathsf{h}}_0}\cup\{I\}} \\
&(c_{II}(\{N_A, N_B\}).\mathsf{i}^{\mathsf{n}}_I \parallel \overline{c_{II}}\langle\{N_A, N_B\}\rangle.\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tau} \\
&(\overline{c_I}\langle\{N_A, N_B\}_{PK_A}\rangle.\mathsf{i}^{\mathsf{n}}_I \parallel \mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\overline{c_I}\langle\{N_A,N_B\}_{PK_A}\rangle} \\
&(c_I(\{N_B\}_{PK_B}).\mathsf{i}^{\mathsf{n}}_I \parallel \mathsf{i}^{\mathsf{h}}_0) \xrightarrow{N_B\in K_{\mathsf{i}^{\mathsf{n}}_I}} \\
&(\overline{c_{II}}\langle\{N_B\}_{PK_B}\rangle.\mathsf{i}^{\mathsf{n}}_I \parallel c_{II}(\{N_B\}_{PK_B}).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tau} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel D_{single}(I).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tilde{E}^{inc(t)}_{\mathsf{i}^{\mathsf{h}}_0}=\tilde{E}^{t}_{\mathsf{i}^{\mathsf{h}}_0}\setminus\{I\}} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel OA_{single}(B,(S,B)).\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\tilde{E}^{inc(t)}_{\mathsf{i}^{\mathsf{h}}_0}=\tilde{E}^{t}_{\mathsf{i}^{\mathsf{h}}_0}\cup\{B\}} \\
&(\mathsf{i}^{\mathsf{n}}_I \parallel \overline{c_B}\langle\{N_B\}_{PK_B}\rangle.\mathsf{i}^{\mathsf{h}}_0) \xrightarrow{\overline{c_B}\langle\{N_B\}_{PK_B}\rangle} 0
\end{aligned}
$$

Figure 7: Normal Attack Trace for the Environment in Figure 5

movements to the resulting trace $\hat{\tau}$.[5] The result of this algorithm is a trace $\hat{\tau}$ without this kind of redundancy. Algorithm 2 similarly removes redundant disassociations.

Given that each application of the normalisation rules in the algorithms reduces the length of the trace, it is not difficult to prove that

Theorem 5.1 (Normalisation). Normalisation terminates and produces an attack trace in normal form.

However, other, more refined, normal forms could be defined. The procedure is actually paramaterised by the protocol itself and the environment (i.e., the initial position of the honest agents), but also by the *initial position of the attacking node*, as can be easily seen from the above case study — imagine that the attacker started in range of B and S instead of the initial position of Figure 5.

5. We use π_n, where $n \in \mathbb{N}$, to represent the (possibly empty) subtrace. To recall our semantics, the full stop "." represents concatenation of an *action* to a trace, but here we define $+$ to denote concatenation between two traces. We use $head$ to extract the first action at the top of a trace, and we use $tail$ to retrieve the rest of the trace minus the head. The common list notation $[a, b, c, \ldots]$ we use to denote a list. Finally, $nCont(x, y)$ returns $True$ if x does not contain y.

Algorithm 1: $redAssoc(\tau, \hat{\tau})$

```
% Let λ be a send or receive action.
if τ ≠ [] then
    if head(τ) == A_single(N) then
        if tail(τ) = π1 + [D_single(N).π2] ∧ nCont(π1, λ)
          then
            τ̂ = τ̂ + redAssoc(π2, τ̂)
        else
            τ̂ = τ̂ + [head(τ)] + redAssoc(tail(τ), τ̂)
        end
    else
        τ̂ = τ̂ + [head(τ)] + redAssoc(tail(τ), τ̂)
    end
    if head(τ) == OA_single(N, (M, N)) then
        if (tail(τ) =
          π1 + [OD_single(N, (M, N)).π2] ∧ nCont(π1, λ)
          then
            τ̂ = τ̂ + redAssoc(π2, τ̂)
        else
            τ̂ = τ̂ + [head(τ)] + redAssoc(tail(τ), τ̂)
        end
    else
        τ̂ = τ̂ + [head(τ)] + redAssoc(tail(τ), τ̂)
    end
    if head(τ) == A_double(N, M) then
        if (tail(τ) =
          π1 + [D_double(N, M).π2] ∧ nCont(π1, λ) then
            τ̂ = τ̂ + redAssoc(π2, τ̂)
        else
            τ̂ = τ̂ + [head(τ)] + redAssoc(tail(τ), τ̂)
        end
    else
        τ̂ = τ̂ + [head(τ)] + redAssoc(tail(τ), τ̂)
    end
end
return τ̂
```

An attack can be carried out significantly faster if the location of all node-behaving agents and the attacker satisfy a specific initial geography. This implies that there are environments which take fewer movements to attack and environments which require many movements, and thus more time, to attack. We are currently expanding normalisation to include all of this explicitly, which will lead us also to identify the optimal initial position of the attacking node with respect to the honest participants. The resulting procedure will combine normalisation and optimisation to reason about the relative efforts required of an attacker given a specific environment, but also identify the environments that are more easily prone to attacks.

5.3. Initial Position of the Attacker

Attacking nodes are capable of dispatching as many attacking hubs as are necessary for an attack, but if an attacking node is far away from an honest protocol participant, the newly dispatched attacking hub will need to move further to reach it. In other words, if an attacking node, upon dispatching, must make more than one move in order to associate with another honest node to send or receive a message, then the attacking node should be initialised nearer to the honest node.

To elaborate on this in more detail, consider three attack traces τ_1, τ_2, and τ_3, each with different node geographies:

- τ_1: the attacking node I and another node B are far away;
- τ_2: I and B are overlapping;

Algorithm 2: $redDisassoc(\tau, \hat{\tau})$

```
% Let λ be a send or receive action.
if τ ≠ [] then
    if head(τ) == D_single(N) then
        if tail(τ) = π1 + [A_single(N).π2] ∧ nCont(π1, λ)
          then
            τ̂ = τ̂ + redDisassoc(π2, τ̂)
        else
            τ̂ = τ̂ + [head(τ)] + redDisassoc(tail(τ), τ̂)
        end
    else
        τ̂ = τ̂ + [head(τ)] + redDisassoc(tail(τ), τ̂)
    end
    if head(τ) == OD_single(N, (M, N)) then
        if (tail(τ) =
          π1 + [OA_single(N, (M, N)).π2] ∧ nCont(π1, λ)
          then
            τ̂ = τ̂ + redDisassoc(π2, τ̂)
        else
            τ̂ = τ̂ + [head(τ)] + redDisassoc(tail(τ), τ̂)
        end
    else
        τ̂ = τ̂ + [head(τ)] + redDisassoc(tail(τ), τ̂)
    end
    if head(τ) == D_double(N, M) then
        if (tail(τ) =
          π1.A_double(N, M).π2) ∧ nCont(π1, λ) then
            τ̂ = τ̂ + redDisassoc(π2, τ̂)
        else
            τ̂ = τ̂ + [head(τ)] + redDisassoc(tail(τ), τ̂)
        end
    else
        τ̂ = τ̂ + [head(τ)] + redDisassoc(tail(τ), τ̂)
    end
end
return τ̂
```

- τ_3: I is inside B.

Using our rules, it is clear that in τ_1, after the attacking hub is dispatched inside the attacking node, it must make two movements in order to enter the range of I. More specifically, the trace τ_1 goes as follows:

$$
\begin{aligned}
& disp.\mathsf{i}_I^{\mathsf{n}} \xrightarrow{disp} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel Hub_{update}.\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{c_I \in K_{\mathsf{i}_0^{\mathsf{h}}}} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel D_{single}(I).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)} = \tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t} \setminus \{I\}} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel A_{single}(B).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)} = \tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t} \cup \{B\}} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}})
\end{aligned}
$$

Similarly, τ_2 is

$$
\begin{aligned}
& disp.\mathsf{i}_I^{\mathsf{n}} \xrightarrow{disp} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel Hub_{update}.\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{c_I \in K_{\mathsf{i}_0^{\mathsf{h}}}} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel A_{single}(B).\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{\tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{inc(t)} = \tilde{E}_{\mathsf{i}_0^{\mathsf{h}}}^{t} \cup \{B\}} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}})
\end{aligned}
$$

and τ_3 is

$$
\begin{aligned}
& disp.\mathsf{i}_I^{\mathsf{n}} \xrightarrow{disp} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel Hub_{update}.\mathsf{i}_0^{\mathsf{h}}) \xrightarrow{c_I \in K_{\mathsf{i}_0^{\mathsf{h}}}} \\
& (\mathsf{i}_I^{\mathsf{n}} \parallel \mathsf{i}_0^{\mathsf{h}})
\end{aligned}
$$

Generally speaking, any movement of the hub after dispatching we would like to avoid. If this is not possible in the protocol context, then the fewer movements the better. In that sense, we can partially order the aforementioned attack traces with respect to number of required attacking hub movements: $\tau_3 < \tau_2 < \tau_1$.

Interestingly, it turns out that normalisation and optimisation may collaborate to produce "better" traces, but also interfere with each other: applying a normalisation rule might prevent an even more relevant optimisation step and vice versa. We are working on algorithms that take this explicitly into account and order the normalisation and optimisation activities to produce the "minimal" attack traces.

6. Conclusions and Future Work

Our approach allows us to reason about (normal) attacks on security protocols in a mobile setting, such as WBANs, where agents need to move in order to send and receive messages. While the foundational work that we have carried out so far is promising, it is clear that much more is needed. First of all, it will be useful to carry out a case study on a real-life protocol, e.g., a key-exchange protocol used in wireless delay tolerant networks [3].

There are some interesting research questions that we can derive from our model. Our case study examines Needham-Schroeder Public Key where movement is a requirement for an attack to be executable, due to the physical locations of the relevant honest participants and the means by which messages are transmitted.

An interesting research question to explore might be to prove certain properties that relate our extended Dolev-Yao intruder model with the standard model. For instance, it is quite obvious that for every attack in the standard Dolev-Yao model there exists an attack in the model with movement. Similarly, it is quite straightforward to show that an attacking node with infinite range in our model is equivalent to the standard Dolev-Yao attacker scenario. What will be more challenging is to identify if and when attacks in our model, in which the attacking node does not have infinite range, can be reduced to attacks in the standard model, or to show that there are attacks on certain protocols that must require movement, where attacks in the standard Dolev-Yao model do not exist.

"Escaping" movements are one of many potential ways that a hub could thwart an attacker. To use our case study as an example, if h_A moves out of range of $\mathsf{i}_I^{\mathsf{n}}$ during the attack, the attack is no longer executable. The extent to which the hub movements can prevent attacks is a question we would like to explore in more detail.

Alongside this, we are aiming to develop optimisation rules that obtain a *minimal attack trace*, in addition to our normalisation rules. While a normal attack is an attack without redundancies, a minimal attack is a normal attack consisting of the fewest possible number of movements to carry out the attack, with respect to the environment.

We also plan to model *attacker-agent* games, where the protocol specification and environmental specification describe the rules of the game, while movements can be viewed as costs and reading/writing certain messages can be viewed as rewards. One type of game that may

capture this is Stackelberg Games, which have already been applied for security analysis [10].

Finally, we plan to automate reasoning by using a modelling and verification tool such as Isabelle or UPPAAL to encode our calculus and our normalisation/optimisation procedure.

References

[1] F. Bai, N. Sadagopan, and A. Helmy. The IMPORTANT framework for analyzing the Impact of Mobility on Performance Of RouTing protocols for Adhoc NeTworks. *Ad hoc networks*, 1(4):383–403, 2003.

[2] S. Brands and D. Chaum. Distance-Bounding Protocols. In *Workshop on the Theory and Application of of Cryptographic Techniques*, LNCS 765, pages 344–359. Springer, 1993.

[3] Y. Chen, Q. Huang, and Z. Zhang. Sakai–Ohgishi–Kasahara identity-based non-interactive key exchange revisited and more. *International Journal of Information Security*, 15(1):15–33, 2016.

[4] J. Cordasco and S. Wetzel. An attacker model for MANET routing security. In *2nd ACM conference on Wireless network security*, pages 87–94. ACM, 2009.

[5] C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance Hijacking Attacks on Distance Bounding Protocols. In *IEEE Symposium on Security and Privacy*, pages 113–127. IEEE, 2012.

[6] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[7] A. Hinds, M. Ngulube, S. Zhu, and H. Al-Aqrabi. A Review of Routing Protocols for Mobile Ad-Hoc NETworks (MANET). *International journal of information and education technology*, 3(1):1, 2013.

[8] S. Lu, L. Li, K.-Y. Lam, and L. Jia. SAODV: a MANET routing protocol that can withstand black hole attack. In *2009 International Conference on Computational Intelligence and Security*, volume 2, pages 421–425. IEEE, 2009.

[9] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. Distance-Bounding Protocols: Verification without Time and Location. In *IEEE Symposium on Security and Privacy*, pages 549–566. IEEE, 2018.

[10] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *7th international joint conference on Autonomous agents and multiagent systems*, pages 895–902. IFAAMAS, 2008.

[11] S. A. Salehi, M. Razzaque, I. Tomeo-Reyes, and N. Hussain. IEEE 802.15.6 standard in wireless body area networks from a healthcare point of view. In *22nd Asia-Pacific Conference on Communications*, pages 523–528. IEEE, 2016.

[12] D. Singelee and B. Preneel. Location verification using secure distance bounding protocols. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*. IEEE, 2005.

[13] M. Toorani. On Vulnerabilities of the Security Association in the IEEE 802.15.6 Standard. In *Financial Cryptography and Data Security – FC 2015 International Workshops, BITCOIN, WAHC, and Wearable*, LNCS 8976, pages 245–260. Springer, 2015.

[14] J. Wild. Number of arrangements of n circles in the affine plane. *The Online Encyclopedia of Integer Sequences*, 2014. https://oeis.org.

Appendix A.
Full NSPK With Movement

In this example, the role B must comprise of both a static node n_B and a hub h_B that is able to move to enter the range of other nodes and send messages. Node n_B and hub h_B must thus co-operate to carry out the protocol. To that end, in the following traces, we model that the co-operating node n_B and hub h_B are in fact the same agent by assuming that they are in possession of a symmetric key K_{BB} that they use to exchange encrypted messages.

Figure 8 shows the execution trace of NSPK with no attacker in the simple environment. Figure 9 shows the trace of Lowe's attack in the simple environment. Figure 10 shows the trace of Lowe's attack in the environment of Figure 4.

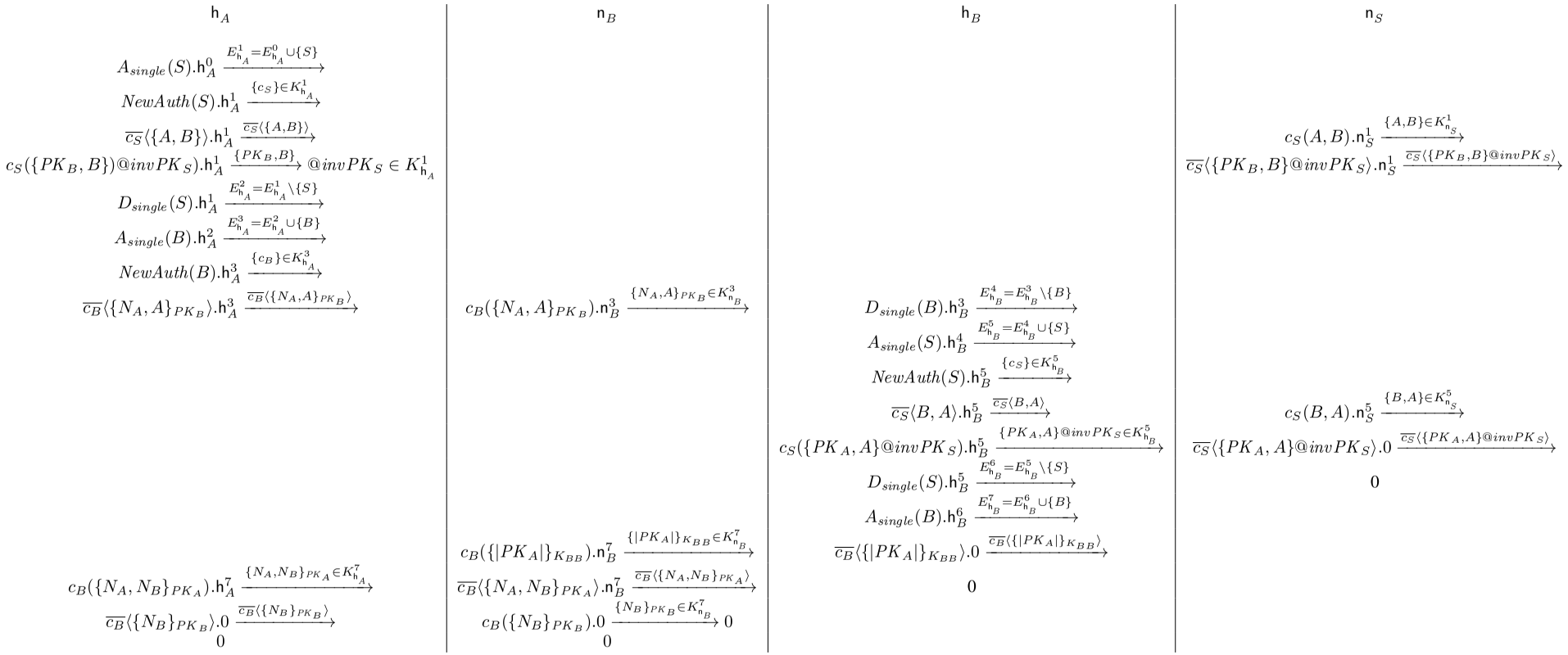

Figure 8: Execution trace of NSPK with no attacker

Figure 9: Trace of Lowe's attack in the simple environment

Figure 10: Trace of Lowe's attack in the environment of Figure 4