Presence, Intervention, Insertion: Unifying Attack and Failure Models in Wireless Sensor Networks

Zinaida Benenson Andreas Dewald Felix C. Freiling

Laboratory for Dependable Distributed Systems
University of Mannheim, Germany
{benenson,dewald,freiling}@informatik.uni-mannheim.de

Technical Report, December 2009, University of Mannheim

ABSTRACT

As assumptions about adversaries critically influence the correctness and efficiency of protocols, they should be as precise as possible. We propose a general framework for attacker models in wirless sensor networks. The framework is modular since it allows to compose *basic attacker models* by choosing values from three orthogonal dimensions: presence, intervention, and insertion. All choices of basic attacker models form a lattice according to a well-defined *weaker-than* relation. Sets of such basic attacker models constitute *general attacker models*. In a sense, our framework unifies behavioral aspects of attacker models from the area of cryptography and fault-tolerance We show that our modeling framework can be used to formulate all relevant attacker models from the literature on sensor networks. We demonstrate the benefits of our framework by showing how it can be used to (1) compare protocols and establish relations between protocols and (2) to make attacker assumptions more precise and find errors in protocols.

1. INTRODUCTION

After roughly a decade of research and development, wireless sensor networks are being deployed in practice for the benefit of users and the network operators. Such networks increasingly process sensitive or otherwise valuable data, making them subject to scrutiny by malevolent parties. Such scrutiny not only targets classical weaknesses of sensor systems, such as those of the used cryptographic protocols. A witness of the growing popularity of wireless sensor networks is also that attacks are aiming at concrete weaknesses in the hardware configuration or the software stack [5, 6, 23, 29, 30, 33]. Overall, the attack surface of wireless sensor networks and protocols is growing, as is the need to carefully design sensor networks and protocols such that they can resist serious attacks.

In most cases, tolerating more attacks implies using more resources within the system. For example, an adversary that can break cryptography is more costly to counter than an adversary that cannot. It is therefore important to precisely define the number and types of attacks that are expected for a concrete system. Such a description is usually termed an *attacker model*, *threat model* or *adversary definition*. Imprecise or incomplete attacker models make it hard to assess system security. Gligor [28] pinpoints this by writing that a "system without an adversary definition cannot be insecure. It can only be astonishing." This is particularly true for a domain like wireless sensor networks that—in sharp contrast to classical desktop/server computing—combines new application scenarios with new system architectures.

1.1 Related Work

Attacker models are a well-established topic in traditional computer security with some roots in the area of fault-tolerant systems where they are called *failure models*. For example, the *Byzantine* failure model [37] allows arbitrary behavior of a subset of components in the system. In a sense, this models worst case behavior that may also be interpreted as that of an intelligent adversary that has compromised (i.e., taken full control of) a sensor node.

In computer security, one of the first well-established and rather formal attacker models was proposed by Dolev and Yao [17]. This model considers an outside adversary who has the full control over the communication channel. The Dolev-Yao adversary can eavesdrop, delay, delete, inject, replay and modify messages, but it cannot compromise nodes.

The Dolev-Yao and the Byzantine models can be combined to form a rich set of attacker models that have been used to analyze security protocols (see for example work by Maurer et al. [22, 43]). Such adversaries are usually classified according to a set of independent (often binary) parameters. For example, a *passive* adversary knows the program and the memory of the compromised nodes, but does not influence their protocol behavior, whereas an *active* adversary can make the compromised nodes execute arbitrary programs. As another example, a *computational* adversary cannot break public-key cryptography whereas an *information-theoretical* adversary possesses unlimited computational resources. An *adaptive* adversary can compromise nodes at any time during the protocol run, whereas the *static* adversary has to decide beforehand which nodes to compromise.

There have been prior attempts to apply and fine tune general attacker models to the area of ad hoc and sensor networks. Most of this work [1, 2, 34, 40, 42, 51] formalizes some particular attacker model and applies it to a particular security problem in ad hoc or sensor networks. We explain this claim in more detail further in the sequel.

There are, to our knowledge, only two publications [6, 12] that define a general and modular adversary model for ad hoc or sensor networks. Both papers categorize attackers according to their ability to influence the nodes in the network and then define a set of ordered and increasingly powerful attacker models that offer a toolbox for system designers to choose.

Benenson et al. [6] define a set of attackers for sensor networks according to their goals, presence in the network, ability to influence network nodes and available resources. However, their work does not offer complete coverage of the attacker space since, for example, node replication attacks and hybrid adversaries (adversaries with multiple, incomparable behavior choices) cannot be modeled.

Cordacso et al. [12] recently defined a set of incrementally powerful attackers according to their communication and computation capabilities. Their model however is targeted to one particular application scenario, namely routing in ad hoc networks, and does not allow to model more restricted attackers that are common in sensor networks (for example, an attacker that only influences the data of a sensor node but not the code).

1.2 Idea

In this paper we propose a general framework for attacker models in wireless sensor networks. The framework is modular since it allows to compose basic attacker models by choosing values from three orthogonal dimensions: presence, intervention, and insertion. Roughly speaking, *presence* denotes the *geographical extent* of the adversary's actions in the sensor network, *intervention* refers to the ways in which the adversary can change the behavior of a single sensor node, and *insertion* denotes the fact whether the adversary has inserted new nodes or is using existing nodes to mount an attack. All choices of basic attacker models form a lattice according to a well-defined *weaker-than* relation. Sets of such basic attacker models constitute general attacker models.

We show that our modeling framework can be used to formulate almost all attacker models from the literature [1, 2, 34, 40, 42, 51], including those of Benenson et al. [6] and Cordasco et al. [12].

In a sense, our framework unifies behavioral aspects of attacker models from the area of cryptography (for an overview see Fitzi et al. [22]) and fault-tolerance (see Warns [54]). Although we do not cover *computational* side of attacker modeling, such as the common distinction between *practical security* (computational power of the adversary is polynomially bounded) and *information theoretic security* (no computational restrictions on the adversary), our model can easily be extended to incorporate these aspects.

1.3 Contribution

As assumptions about adversaries critically influence the correctness and efficiency of protocols, they should be as precise as possible. In this paper we make the following contributions:

- We propose a framework of well-defined attacker models that gives algorithm and system designers a toolbox to choose from. Our framework is particularly aimed at wireless sensor networks with their novel physical and logical attack vectors on the sensor nodes themselves.
- We demonstrate the benefits of our framework by showing how it can be used to (1) compare protocols and establish relations between protocols and (2) to make attacker assumptions more precise and find errors in protocols.

Our framework is based on a theory of trace-based system semantics so it can potentially also be used for rigorous formal analysis of protocols.

1.4 Outline

This paper is structured as follows: In the next Section 2 we present our basic attacker model that is composed of three dimensions presence, intervention and insertion. In Section 3 we then present a general attacker model that consists of a composition of basic adversary models. We then show that our model generalizes previous adversary models for sensor networks in Section 4, and how our model can be used to establish precise adversary models and relationships between protocols in Section 5. We present theoretical background of our model in Section 6 and conclude in Section 7.

Throughout this paper, we use the terms *adversary* and *attacker* interchangeably.

2. BASIC ATTACKER MODEL

In this section we present our basic attacker model. This model can be used to formulate assumptions about the capabilities of the adversary according to its impact on the behavior of sensor nodes. That is, our model is *node-centered.* The attacker's effects on communication channels are attributed to its ability to influence input and output interfaces of the nodes. We later generalize basic attacker models to *general attacker models* in Section 3.

A *basic adversary model* is a triple (p, i, j) where the three parameters p, i and j represent values from three dimensions *Presence*, *Intervention* and *Insertion*, respectively. Roughly speaking, the *Presence* parameter describes *where* in the sensor field the adversary can influence the behavior of the sensor nodes. The *Intervention* parameter describes *what* the adversary can do with the sensor nodes. Finally, the *Insertion* parameter indicates whether the adversary is able to *add its own nodes* to the network.

In the following, we describe each parameter in detail.

2.1 Presence

Presence describes *where* the attacker acts. We present increasingly stronger levels of the adversary's presence in the sensor network. The levels are defined in an incremental fashion, such that stronger levels include the characteristics of previous levels. We consider local, distributed and global adversaries.

2.1.1 Local

A local presence is the weakest presence level. A local adversary can influence a small (relatively to the number of nodes in the network) spatially close subset of sensor nodes. Usually this subset would be connected with respect to the sensor network topology. For example, this can be a stationary attacker equipped with a simple receiver that can only eavesdrop in the range of a few meters, or an attacker that can only influence sensor readings or tamper with sensor nodes in a localized part of a building, e.g., in his own office.

2.1.2 Distributed

The distributed presence affects multiple small parts of the sensor network, i.e., it comprises multiple local adversaries. The total set of attacked nodes is usually unconnected with respect to the WSN topology, although there may exist out-of-band ways of attack coordination. An example can be a mobile attacker that can influence a small part of nodes in every location where it moves to. Another example is a spatially uniformly distributed adversary that affects some percent of sensor nodes. For small values of t, a "t-out-of-n" threshold adversary may also be modeled as a distributed adversary.

2.1.3 Global

The global presence is able to influence the whole sensor network, or a very large part of it. For example, a global adversary may be able to eavesdrop on the whole network using a very sensitive receiver or multiple eavesdropping stations, or it can jam the whole network using a powerful transmitter. The worst-case global adversary would be the one that exploits software vulnerabilities to injects self-propagating malicious code (a WSN worm) [23, 33].

2.2 Intervention

The intervention parameter describes *how* the attacker can influence the sensor nodes. We present the attacker's abilities as a partially ordered lattice where the stronger attacker ability includes the behavior of all lower levels. In Section 6 we give a theoretical justification of our partial order using the theory of trace-based system properties. We consider crashing, eavesdropping, x-raying, disturbing, modifying and reprogramming adversaries. The lattice is shown in Figure 1.

2.2.1 Crashing

Crashing intervention means that the adversary can irrevocably destroy sensor nodes without reading any internally stored data. This can be done, e.g., by physical destruction or by removing node's battery.

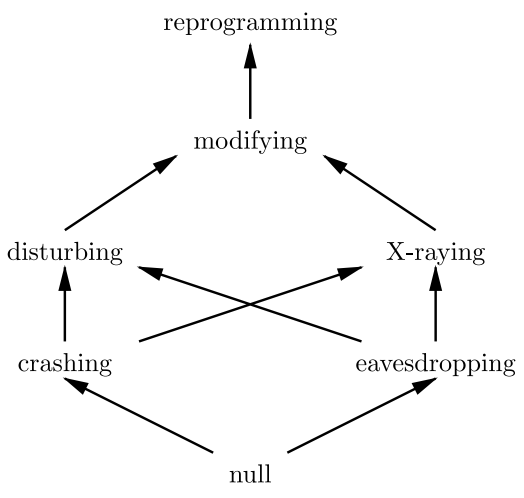

Figure 1: Lattice of intervention levels, where $X \to Y$ means Y includes all attacker behavior of X.

2.2.2 Eavesdropping

Eavesdropping intervention means that the adversary can hear all messages on incoming and outgoing channels of the attacked sensor node. This can be done, e.g., by placing a receiver in sensor node's vicinity.

2.2.3 X-raying

X-raying intervention denotes a combination of crashing and eavesdropping intervention plus some additional capabilities. This attacker can read full memory contents (data and code) of the sensor node. However, it cannot modify node's state. This can be done, for example, via physical access to node's hardware using the JTAG interface [5].

2.2.4 Disturbing

Disturbing intervention combines crashing and eavesdropping capabilities with the ability to partially modify data memory of the sensor node. For example, such an adversary can influence node's routing table by sending fake routing table updates, or influence temperature reading by warming the node with a cigarette lighter. The disturbing adversary can also delete, inject, replay and spoof packets, as the effect of these actions can also be modeled as changing the contents of node's memory (the communication registers). However, a disturbing adversary does not have any direct reading or writing access to node's other data or program memory.

2.2.5 Modifying

Modifying intervention combines x-raying and disturbing capabilities and can furthermore inspect and modify full data memory of the sensor node. However, it cannot change the node's program. This distinction is motivated, among other things, by the difference between data and program space in Harvard architecture devices such as MICA2 [13] and MICAz [14] sensor nodes.

2.2.6 Reprogramming

The strongest intervention level is reprogramming. A reprogramming adversary can make the attacked sensor node exhibit arbitrary behavior, i.e., run an arbitrary program. This adversary behavior naturally includes behavior of all previous adversaries and can be achieved, e.g, through full physical read/write access to the sensor node or through malicious software that spreads amoung the nodes through a software vulnerability.

2.3 Insertion

We now discsus the final parameter of our basic attacker model. As sensor networks operate unattended, it is possible for the adversary to insert its own nodes, or generally speaking, devices into the network. This called *insertion* in our model and can take two different values: *true* or *false*. Such an attacker assumption is also common in ad hoc networks [34]. Note that the adversary does not need to use exactly those nodes that are identical to the original nodes. Instead, the adversary can use PDA-class nodes, each of them simulating a small number of original sensor nodes.

Intuitively, Insertion refers to the *granularity* of the adversary's presence and it is *true* if the adversary can attack the network at the level of sensor nodes, and is *false* otherwise. If Insertion is *true*, the new nodes are distributed in the network according to the Presence parameter and possess the capabilities according to the Intervention parameter.

For example, the adversary may be able to add new nodes that are uniformly distributed in the sensor field and can selectively jam the communication channels. An eavesdropping adversary with Insertion will be able to recognize communication between a particular pair of nodes more easily than an adversary that has to listen to the communication of the whole network and then single out from noise the communication of individual nodes. A distributed disturbing adversary with Insertion can jam individual links selectively and thus remain undetected.

A reprogramming adversary that can insert its own nodes into the network can run a *node replication attack* [47]. If the adversary is precluded from doing so, it is restricted to reprogramming of the nodes already present in the network. This can make important difference if, for example, the network operation depends on some voting mechanism. In this case an adversary that can add its own specially programmed nodes into the network will be able to overrule the voting of the original nodes.

An adversary that cannot insert nodes is weaker than an inserting adversary. Thus, also the Insertion dimension is ordered with respect to the adversary's behavior.

In case insertion is false, we may also omit mentioning the third parameter, i.e., $\mathcal{A}(\text{local}, \text{reprogramming}, \text{false})$ can also be written as $\mathcal{A}(\text{local}, \text{reprogramming})$.

3. GENERAL ADVERSARY MODEL

A *general adversary model* is a set of basic adversary models $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. This definition makes it possible to define hybrid attacker assumptions. For example, an adversary can be global eavesdropping at the node level and local reprogramming at the same time. This is then specified as

$$\{\mathcal{A}(\text{global}, \text{eavesdropping}, \text{true}), \mathcal{A}(\text{local}, \text{reprogramming})\}.$$

To simplify the notation, we assume that the set of basic adversary models is closed with respect to the weaker-than relation on such models. For example, specifying

$$\mathcal{A}(\text{global}, \text{reprogramming})$$

as one of the basic adversary models implies that all weaker basic adversary models like

$$\mathcal{A}(\text{local}, \text{reprogramming}) \text{ or } \mathcal{A}(\text{global}, \text{disturbing})$$

are in the set too. Therefore, the general adversary model

$$\{\mathcal{A}(\text{global}, \text{reprogramming}), \mathcal{A}(\text{local}, \text{eavesdropping})\}$$

is equivalent to $\{\mathcal{A}(\text{global}, \text{reprogramming})\}$.

4. ADEQUACY OF OUR MODEL

4.1 Suitability for Sensor Networks

Our model is specifically tailored for sensor networks:

- We consider adversarial node insertion that is typical for WSNs.
- Existence of environmental sensors results in disturbing adversary that can influence sensor readings.
- If the adversary has physical access to sensor node (e.g., x-raying adversary), it can also crash the node.
- Difference between data and code in Harvard architecture motivates difference between modifying and reprogramming adversary.

Although some of these points are also typical for ad hoc networks, their combination is unique to WSNs.

4.2 Relation to Other Adversary Models

We now show that our model generalizes previous models, see Section 1.1.

Hu et al. [34] define the active $n - m$ attacker model for ad hoc networks where the attacker is a legitimate owner of m nodes and additionally compromises n nodes belonging to other users. This attacker model can be formalized as $\{\mathcal{A}(\text{distributed}, \text{reprogramming}, \text{true})\}$.

Ács et al. [2] present a cryptographic adversary model based on the simulation paradigm. Later the authors extend their model to routing in sensor networks [1]. The capabilities of the adversary are firstly described informally, and afterwards formalized in the simulation paradigm. The considered adversary model is $\{\mathcal{A}(\text{distributed}, \text{disturbing}, \text{true})\}$.

Another cryptographic model tailored specifically for data aggregation in sensor networks is presented by Manulis et al. [42]. This model describes a concrete adversary that is formalized with respect to a specific cryptographic model. The used model is

$$\{\mathcal{A}(\text{global}, \text{disturbing}), \mathcal{A}(\text{distributed}, \text{reprogramming})\}.$$

Ma et al. [40] present several adversary models for a special kind of sensor networks that they call unattended. The presented adversaries are mobile and can compromise certain number of sensor nodes in a time interval. The considered adversaries are

$\{\mathcal{A}(\text{distributed}, \text{x-raying})\}$ and $\{\mathcal{A}(\text{distributed}, \text{modifying})\}$.

Tangue et al. [51] present a model for the possible strategies of an adaptive adversary that wishes to learn by means of node capture attacks certain secrets that are kept by sensor nodes. The considered adversary model is

$\{\mathcal{A}(\text{global}, \text{disturbing}, \text{true}), \mathcal{A}(\text{distributed}, \text{modifying}, \text{true})\}$.

Benenson et al. [6] define a set of attackers for sensor networks that is similar to our work, although they do not consider node insertion and hybrid adversaries. They also use a slightly different terminology, e.g., their eavesdropping adversary can also destroy sensor nodes. Nevertheless, any attacker model that can be described using their terminology can also be described using ours. On the other hand, their model is not suitable to express many of the adversaries presented in this section. Thus, our model generalizes the model from [6].

Finally, Cordacso et al. [12] define a set of modular adversaries that is similar to ours. Their general adversary model consists of communication and computation dimensions. The computation dimension is not formalized. For communication dimension they present five increasingly strong adversary types with respect to their influence on node's communication. These types can be expressed as:
(1) $\mathcal{A}(\text{local}, \text{disturbing})$,
(2) $\mathcal{A}(\text{local}, \text{disturbing})$ & $\mathcal{A}(\text{global}, \text{eavesdropping})$,
(3) $\mathcal{A}(\text{distributed}, \text{disturbing})$,
(4) $\mathcal{A}(\text{distributed}, \text{disturbing})$ & $\mathcal{A}(\text{global}, \text{eavesdropping})$,
and (5) $\mathcal{A}(\text{global}, \text{disturbing})$.

Although it is very difficult to produce an exhaustive list of adversary models from the previous literature, we showed that all important adversary models we are aware of can be expressed using our adversary model.

5. BENEFITS

To show the benefits of our framework we investigated many papers on WSN security with respect to their adversary definitions. The field of WSN security started developing in the beginning of 21st century, such that most of pioneering works, e.g., works that for the first time define and solve a concrete WSN security problem, appeared in 2001-2005, with respective successor papers that developed more and more advanced solutions appearing in 2003-2009.

We considered early as well as more recent papers dedicated to security problems such as key management [4, 10, 18, 20, 38], secure node-to-node and broadcast communication [7, 9, 48, 57], secure data report [56, 58] and secure data aggregation [11, 49, 53, 55], secure routing [1, 15, 35], localization [39, 52] and time synchronization [25, 50].

Due to space limit we cannot present our full analysis here. Probably the most important concern revealed by our analysis is that in many cases it is exceptionally difficult to pinpoint the exact adversary model used in the protocols. This is especially true for pioneering works. More recent papers, mercifully, are usually more precise in their adversary definitions.

Whereas absence of an exact adversary model is understandable for early papers, problems arise when more recent papers use the early protocols as building blocks. In many cases some security solutions use as building blocks protocols that were developed for a weaker adversary model, and thus the whole solution becomes insecure. This kind of mistake is very dangerous, as any security analysis in this case is made invalid by the vulnerabilities introduced through the incorrect usage of previously developed protocols.

To illustrate how our model helps to determine the exact adversary model of a protocol, to establish relationships between the protocols, and to discover incorrect usage of protocol composition, we present our analysis of the pioneering work on key management by Eschenauer and Gligor [20] published in 2002, and of some protocols that use their key management scheme as building blocks. We call the key management scheme from [20] the *EG scheme* in the following.

5.1 Adversary Model in the EG Scheme

The first published key management scheme for sensor networks is by Eschenauer and Gligor [20]. This influential work was cited by more than 1500 papers since according to GoogleScholar [31], and was patented in 2009 [21]. The EG key management scheme was repeatedly used as a building block. The usual context of such usages is a "higher-level" secure protocol, such as routing, time synchronization, or localization that needs some kind of key management. In this case, the authors frequently say that one of possible key management schemes to use with their solution is the EG scheme.

In the EG scheme, the keys of each node are drawn from a large key pool such that any two nodes receive several keys before the deployment and share a key with a predefined probability. After the deployment, the nodes have to discover whether they share any keys with their neighbors. If some neighbors do not share any keys, a path key setup mechanism is used where two neighbors not in possession of a shared key use secure links to other nodes for a key establishment procedure.

What adversary model is used in the EG scheme? We list in the order of appearance some citations from the original paper that are relevant for determining the general adversary model, and also basic adversary models that fit these citations:

- "DSNs may be deployed in hostile areas where communication is monitored and nodes are subject to capture and surreptitious use by an adversary." Here the words "capture" and "use" enable ambiguous interpretations, from $\mathcal{A}(*, eavesdropping)$ to $\mathcal{A}(*, reprogramming)$. We note that very often in the subsequent literature, a reprogramming adversary is inferred. The following analysis shows that the EG adversary is in reality much weaker.
- The citation "[...] if an adversary captures a node he

can discover which key of that node is used for which link [...]" points to an x-raying adversary.

- "We distinguish between two levels of threats posed by node capture and potential countermeasures. The first is that of active manipulation of a sensor's data-inputs." Here, at least disturbing adversary is assumed.
- "The second level of threat materializes when a sensor node is under the complete physical control of the adversary." Here, the words "complete physical control" can be interpreted as x-raying, modifying or reprogramming.
- "[...] [The second level] includes the first, and in addition enables an adversary to mount attacks against other sensors of the DSN. For example, an adversary can launch a sleep-deprivation attack [...]" Here we note the modular approach to the adversary description that is also characteristic for our model. This citation clearly assumes a reprogramming adversary, as the sleep deprivation attack implies that a malicious sensor node engages legitimate nodes in extensive communication that depletes their battery.
- "Although we assume tamper-detection via sensor-node shielding that erases the keys of captured nodes, [...]" mentions very occasionally that the "real" EG adversary is disturbing after all.
- "[...] we note that our key-distribution scheme is more robust than those based on a single mission key or on pair-wise private sharing of keys even in the face of physical attacks against captured unshielded sensor nodes. [...] in our scheme only the $k \ll n$ keys of a single ring are obtained, which means that the attacker has a probability of approximately k/P to attack successfully any DSN link." This citation shows that the EG scheme possesses *gracefully degrading* security: Even if the basic adversary assumption (disturbing) is violated, the scheme remains partially resistant against the X-ray adversary. This adversary can use the keys of captured nodes in order to eavesdrop globally on the network communication.

In the subsequent parts of the original paper, the EG scheme is analyzed with respect to

$$\{\mathcal{A}(\text{global}, \text{eavesdropping}), \mathcal{A}(\text{local}, \text{x-raying})\}$$

adversary. By analysis and simulation the number of links is determined on which the adversary will be able to eavesdrop in case it knows the keys of one node.

We emphasize that the EG scheme was not analyzed with respect to reprogramming adversary neither in the original paper nor in the patent. The EG scheme is susceptible, for example, to the man-in-the-middle attack on the path key setup mechanism, and to node replication attacks. That is, one have to augment the EG scheme with additional security means if it is to be used with a reprogramming adversary.

5.2 Usage of the EG Scheme as Building Block

As already mentioned, best candidates for usage of a "foreign" key management protocol as a building block are works on such "higher-level" tasks as routing, localization or time synchronization.

We found at least three papers that present sophisticated solutions to secure positioning [52], localization [39] and time synchronization [26] and use the EG scheme as building block. All three works assume a reprogramming adversary with insertion, but propose to use, as one of possible variants, the EG scheme for key management. All three papers would exhibit serious security problems if the key management scheme they use is susceptible to node replication attacks.

Although the EG scheme is only mentioned as one of possible key management schemes in these works, one should not forget that the EG scheme could actually be chosen for key management in a sensor network, especially for industrial purposes. Probably the fact that the scheme is patented would even give the developers a kind of additional assurance in the security of the scheme.

6. THEORETICAL BASIS FOR OUR ATTACKER MODEL

Our attacker model is based on a theoretical model of system properties that has its roots in linear temporal logic [19]. Systems are modeled as finite state automata that proceed by executing actions thereby changing their state. System execution is modeled as a sequence of states or a sequence of pairs of states and actions. These sequences are often called *traces*. A system *property* is then defined to be a set of traces. A system satisfies a certain property if all traces that can be generated by the system belong to the property.

Using such a formal model, it is possible to distiguish different types of properties (safety, liveness and information flow properties) and reason about their composition [8, 24]. In this section, we explain the different classes of properties and their variants to show how the intervention aspect of our attacker model can be motivated. We also relate the variants of system properties to existing attack and failure models in the literature.

6.1 Safety and Liveness Properties

Lamport [36] termed two fundamental classes of trace properties *safety* and *liveness*. Briefly spoken, safety properties describe what is not allowed to happen. For example, the property that "if a result is received then it satisfies a certain condition" is a safety property. Integrity and authenticity of messages can also be formalized as safety properties. Safety properties are formalized through prefix-closure of trace sets, i.e., for any trace in the property, every prefix of that trace must also belong to the property. A safety property therefore rules out a set of "bad" trace suffixes that correspond to the "bad" event sequences that are not allowed to happen.

Liveness properties demand what eventually must happen, e.g., "every message which is sent is eventually received". Other examples of liveness properties are the termination of a protocol or the fact that messages are sent periodically. Formally, for any liveness property, any finite trace can be extended to belong to that property. This formalizes the intuition that the "good" thing remains always possible. Liveness properties can be used to specify weak timing conditions, namely the difference between finite and infinite time. Real-time properties, like "a message is received within 5 ms" are usually formalized as safety properties.

Later, Alpern and Schneider [3] provided formal definitions of safety and liveness. They proved a *decomposition theorem*, i.e., that every trace set property can be written as the intersection of a safety and a liveness property.

6.2 Information Flow Properties

In the area of specifying and verifying security properties, trace-based formalisms have been adopted, too (see for example Gray III. and McLean [32] or Mantel and Sabelfeld [41]). *Possibilistic security properties* [44, 45] use a particular type of trace-based formalization to specify the absence of information flow in multi-level security environments. Information flow properties can be used to specify who learns what information in a computer system, i.e., what remains confidential to who and what does not. Examples of such properties are "the contents of a message remains confidential to sender and receiver of that message". Another information flow property is "no other entity learns the value of a cryptographic key stored in the memory of some node".

The idea of possibilistic security properties is that information cannot be deduced by observing the system because the set of traces which might have generated this observation is too large. Interestingly, possibilistic security properties fall outside of the safety/liveness classification because they describe *properties* of trace sets, and hence must be formalized as a *set of sets of traces* [46]. Information flow properties are therefore orthogonal to safety and liveness properties.

6.3 Influencing Safety, Liveness, Information Flow

Considering an individial sensor node, we can ask: In what way does the attacker change the system properties of that node? We now describe general ways in which adversaries can change the behavior of sensor nodes and relate these variants to approaches in the literature.

6.3.1 Safety

Apart from changing nothing, we distinguish three incremental ways in which the attacker can influence the safety properties of a node:

- The attacker can change certain parts of the state of the node. We call this *limited state perturbation.* In sensor networks, this corresponds to manipulation of sensor readings, for example, by covering a light sensor or heating a temperature sensor.
- The attacker can change the entire state of the node. We call this *full state perturbation.*
- The attacker can change the entire state as well as the entire code of the node. We call this *full code perturbation.*

The idea of limited state perturbation is that the adversary is allowed to write certain "input registers" of the sensor node. For example, these can be the memory location associated with sensor readings or the input buffers of the wireless communication interface. Replaying messages or injecting fake queries can therefore also be regarded as limited state perturbation.

The full state perturbation model generalizes the limited state perturbation by allowing full modification of data. This is motivated from the volatility of main memory (RAM) which stores data (in contrast to the non-volatility of code often stored in a ROM). It is well-known from the area of fault-tolerance, that volatile memory can always suffer corruption from random bit flips or cosmic rays. The area of *self-stabilization* [16] explicitly has investigated algorithms that tolerate full state perturbation, including modifications of the program counter.

The assumption of full code perturbation usually implies full access of the adversary to the sensor node. This results in possibly arbitrary behavior, a worst-case assumption also made in the Byzantine failure model [37], allthough the original definition did not intend to model malicious activity [27].

6.3.2 Liveness

Apart from changing nothing, the attacker has, in principle, two different ways to influence the liveness properties of a node:

- The attacker can prevent the node from executing further processing steps *forever.* We call this behavior *crash.* Crash can be regarded as the effect of physically destroying the node or removing it totally from the sensor network.
- The attacker can *temporarily* prevent the node from executing steps. This means that he can crash a node and later on let it recover and have it continue executing steps. We call this behavior *crash-recovery.* This behavior includes *crash* and is therefore considered more severe.

6.3.3 Information Flow

Regarding the information flow properties of the node, we distinguish two different incremental types of information flow useful to the attacker:

- The attacker can learn the contents of the messages in transit from or to a node. This means that the attacker can observe the *network.*
- Apart from observing the network, the attacker can also learn the contents of the node's memory. Especially data can be interesting because it may contain the contents of cryptographic keys. Here, the attacker can look into the *participant.*

6.4 Three Dimensions of Intervention

The three distinct classes of properties (safety, liveness, information flow) form the basis of a three dimensional property space that can be used to define useful levels of the intervention aspect of our attacker model [8]. The different

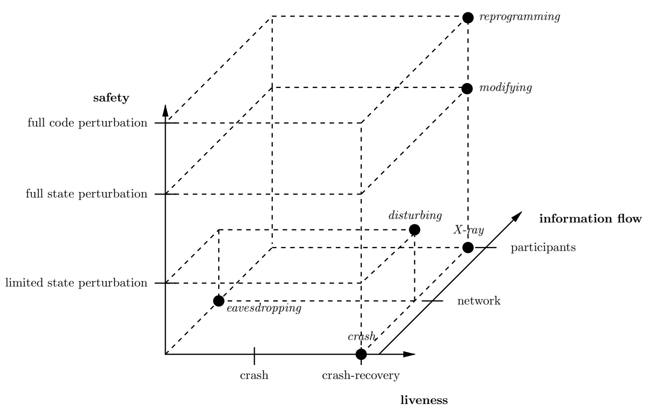

Figure 2: Embedding intervention levels in a general attacker model.

“grades” of these dimensions open up a lattice of intervention levels that can be used to motivate the structure of our attacker model in Section 2 (see also Fig. 2).

The *crash* adversary from Section 2 results from the following combination: there is no change in the safety properties, no change in the information flow properties, but a complete change in the liveness properties of the node. This means that the node may stop to execute steps at any time and at any time resume to execute steps. Nothing else is allowed as additional behavior.

The *eavesdropping* adversary results from no change in safety, no change in liveness, but network level information flow, i.e., the possibility for the attacker to learn the contents of the messages in transit. A global eavesdropping adversary is part of almost every adversary definition in cryptographic protocols.

The *X-ray* adversary combines crash-recovery with complete (participant) information flow to the attacker. This adversary comes close to what is usually termed *passive adversary* in many cryptographic protocols: such an adversary can globally eavesdrop on all communication and inspect the memory of all compromised nodes. However, a passive adversary usually cannot influence the liveness of the sensor node, which an X-ray adversary can. We think that an adversary who can inspect the memory of a sensor node must be phyisically present in the network. It is therefore reasonable to assume that he can also phyiscally destroy the sensor node, inhibiting its liveness properties.

The *disturbing* adversary is the combination of crash-recovery for liveness, eavesdropping for information flow and limited state perturbation for safety. Note that neither does a disturbing adversary imply an X-ray adversary nor vice versa.

The disturbing adversary comes close to the Dolev-Yao attacker model [17] mentioned in the introduction. Roughly speaking, the Dolev-Yao adversary has full control over the communication channel. The Dolev-Yao adversary can eavesdrop, delay, delete, inject, replay and modify messages, but it cannot compromise nodes. In our framework, Dolev-Yao combines a global eavesdropping with a local or distributed disturbing adversary. The limited state perturbation allowed by the safety aspect of the framework models messages injected into the communication buffers of sensor nodes (either those of outgoing messages or incoming messages). Dolev-Yao does not allow other data modifications of the sensor nodes themselves.

The *modifying* adversary covers the combination of X-ray and full state perturbation. In this model, the attacker can influence the data of a node but not the code.

Finally, the *reprogramming* adversary is the maximum of the lattice. The adversary can influence safety, liveness and information flow without restriction. The reprogramming adversary basically is another name for a Byzantine failure [37].

7. CONCLUSIONS AND FUTURE WORK

We presented a novel general framework for modeling adversaries in sensor network. Our framework unifies known attack and failure models from the previous literature and is based on theory of trace-based system properties such that it can potentially also be used for rigorous formal protocol analysis.

We showed that our model can be used to make attacker assumptions of the protocols explicit, to compare protocols with respect to their adversary models and to find out incorrect protocol compositions that may lead to unexpected security breaches.

The presented adversary model is very flexible and can be extended in many ways. For example, a time dimension may be added to the model in order to incorporate time-dependent development of basic attacker models, such as an adaptive adversary that compromises sensor nodes according to a sophisticated strategy [40, 51], or an adversary that is not present in the network initialization phase [4]. Another extension can be the modeling of compromised base stations.

We also believe that our model can be extended for usage in other types of future networks, especially in ubiquitous computing.

8. REFERENCES

[1] G. Ács, L. Buttyán, and I. Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *SASN ’06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 49–58, New York, NY, USA, 2006. ACM.

[2] G. Acs, L. Buttyan, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1533–1546, 2006.

[3] B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, 1985.

[4] R. J. Anderson, H. Chan, and A. Perrig. Key infection: Smart trust for smart dust. In *ICNP*, pages 206–215, 2004.

[5] A. Becher, Z. Benenson, and M. Dornseif. Tampering

with motes: Real-world physical attacks on wireless sensor networks. In *Security in Pervasive Computing, Third International Conference, SPC 2006*, Lecture Notes in Computer Science 3934, pages 104–118. Springer, 2006.

[6] Z. Benenson, P. M. Cholewinski, and F. C. Freiling. Vulnerabilities and attacks in wireless sensor networks. In J. Lopez and J. Zhou, editors, *Wireless Sensors Networks Security*, Cryptology & Information Security Series (CIS), pages 22–43. IOS Press, 2008.

[7] Z. Benenson, F. C. Freiling, E. Hammerschmidt, S. Lucks, and L. Pimenidis. Authenticated query flooding in sensor networks. In *Security and Privacy in Dynamic Environments, Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006)*, pages 38–49, 2006.

[8] Z. Benenson, F. C. Freiling, T. Holz, D. Kesdogan, and L. D. Penso. Safety, liveness, and information flow: Dependability revisited. In W. Karl, J. Becker, K.-E. Großpietsch, C. Hochberger, and E. Maehle, editors, *ARCS 2006 - 19th International Conference on Architecture of Computing Systems, Workshops Proceedings, March 16, 2006, Frankfurt am Main, Germany*, volume 81 of *LNI*, pages 56–65. GI, 2006.

[9] H. Chan and A. Perrig. Efficient security primitives derived from a secure aggregation algorithm. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 521–534, New York, NY, USA, 2008. ACM.

[10] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, May 2003.

[11] H. Chan, A. Perrig, and D. Song. Secure hierarchical in-network aggregation in sensor networks. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 278–287, New York, NY, USA, 2006. ACM.

[12] J. Cordasco and S. Wetzel. An attacker model for MANET routing security. In *Proc. WiSec*, pages 87–93, Zurich, Switzerland, may 2009.

[13] Crossbow, Inc. MICA2 data sheet. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.

[14] Crossbow, Inc. MICAz data sheet. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[15] J. Deng, R. Han, and S. Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *2nd IEEE International Workshop on Information Processing in Sensor Networks (IPSN 2003)*, April 2003.

[16] E. W. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

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

[18] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[19] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 997–1072. Elsevier, 1990.

[20] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM Press, 2002.

[21] L. Eschenauer and V. D. Gligor. Method and apparatus for key management in distributed sensor networks. US Patent No. US 7,486,795 B2, February 2009.

[22] M. Fitzi, M. Hirt, and U. M. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT '99: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 232–246, London, UK, 1999. Springer-Verlag.

[23] A. Francillon and C. Castelluccia. Code injection attacks on harvard-architecture devices. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 15–26, New York, NY, USA, 2008. ACM.

[24] F. C. Freiling and T. Santen. On the composition of compositional reasoning. In R. H. Reussner, J. A. Stafford, and C. A. Szyperski, editors, *Architecting Systems with Trustworthy Components*, volume 3938 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2004.

[25] S. Ganeriwal, C. Pöpper, S. Čapkun, and M. B. Srivastava. Secure time synchronization in sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4):1–35, 2008.

[26] S. Ganeriwal, C. Pöpper, S. Čapkun, and M. B. Srivastava. Secure time synchronization in sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4):1–35, 2008.

[27] F. C. Gärtner. Byzantine Failures and Security: Arbitrary is not (always) Random. Technical Report LPD-REPORT-2003-009, EPFL, 2003.

[28] V. Gligor. On the evolution of adversary models in security protocols: from the beginning to sensor networks. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 3–3, New York, NY, USA, 2007. ACM.

[29] T. Goodspeed. Machine code injection for wireless devices. Internet: `http://travisgoodspeed.blogspot.com/2007/08/machine-code-injection-for-wireless.html`, 2007. Blog entry.

[30] T. Goodspeed. Memory-constrained code injection. Internet: `http://travisgoodspeed.blogspot.com/2007/09/memory-constrained-code-injection.html`, 2007. Blog entry.

[31] GoogleScholar. http://scholar.google.de, accessed at 7. Sept. 2009.

[32] J. W. Gray, III. and J. McLean. Using temporal logic to specify and verify cryptographic protocols. In *Proceedings of the Eighth Computer Security Foundations Workshop (CSFW '95)*, pages 108–117.

IEEE Computer Society Press, June 1995.

[33] Q. Gu and R. Noorani. Towards self-propagate mal-packets in sensor networks. In *Proc. ACM Conference on Wireless Network Security*, 2007.

[34] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wirel. Netw.*, 11(1-2):21–38, 2005.

[35] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Network Journal, Special Issue on Sensor Network Applications and Protocols*, September 2003.

[36] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, Mar. 1977.

[37] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[38] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005.

[39] D. Liu, P. Ning, A. Liu, C. Wang, and W. K. Du. Attack-resistant location estimation in wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4):1–39, 2008.

[40] D. Ma, C. Soriente, and G. Tsudik. New adversary and new threats: security in unattended sensor networks. *IEEE Network*, 23(2):43–48, 2009.

[41] H. Mantel and A. Sabelfeld. A generic approach to the security of multi-threaded programs. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW 2001)*, pages 126–142, Cape Breton, Nova Scotia, Canada, June 11–13 2001. IEEE Computer Society Press.

[42] M. Manulis and J. Schwenk. Security model and framework for information aggregation in sensor networks. *ACM Trans. Sen. Netw.*, 5(2):1–28, 2009.

[43] U. Maurer. Secure multi-party computation made simple. *Discrete Appl. Math.*, 154(2):370–381, 2006.

[44] J. McLean. Security models and information flow. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 180–187, Oakland,CA, 1990.

[45] J. McLean. Security models. In J. Marciniak, editor, *Encyclopedia of Software Engineering*. John Wiley & Sons, 1994.

[46] J. McLean. A general theory of composition for a class of "possibilistic" properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, Jan. 1996. Special Section—Best Papers of the IEEE Symposium on Security and Privacy 1994.

[47] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2005.

[48] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.

[49] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SenSys 2003*, Nov 2003.

[50] K. Sun, P. Ning, and C. Wang. Tinysersync: secure and resilient time synchronization in wireless sensor networks. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 264–277, New York, NY, USA, 2006. ACM.

[51] P. Tague and R. Poovendran. Modeling adaptive node capture attacks in multi-hop wireless networks. *Ad Hoc Netw.*, 5(6):801–814, 2007.

[52] S. Čapkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *IEEE Infocom*, 2005.

[53] D. Wagner. Resilient aggregation in sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87. ACM Press, 2004.

[54] T. Warns. *Structural Failure Models for Fault-Tolerant Distributed Computing*. PhD thesis, University of Oldenburg, Department of Computer Science, Germany, Oct. 2009.

[55] D. Westhoff, J. Girao, and M. Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Transactions on Mobile Computing*, 5(10):1417–1431, 2006.

[56] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. *IEEE Jounal on Selected Areas in Communications, Special Issue on Self-organizing Distributed Collaborative Sensor Networks*, 2005.

[57] S. Zhu, S. Setia, and S. Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 62–72. ACM Press, 2003.

[58] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE Symposium on Security and Privacy*, pages 259–271, 2004.