NEW CONCEPTUAL REPRESENTATION OF COLLISION ATTACK IN WIRELESS SENSOR NETWORKS

S. AL-FEDAGHI
Department of Computer Engineering, Kuwait University, Kuwait.

ABSTRACT
Diagrammatic methodologies for modeling information security attacks have been developed in various forms (e.g. attack trees, use cases, and misuse cases) and applied for many purposes (e.g. security requirements specification and identification of commonly occurring attack patterns). They play an important role in the development of more effective communication between technical and nontechnical participants than that made possible by text. Recently, Unified Modeling Language (UML) sequence diagrams have been used to model security attacks (e.g. collision attacks and unintelligent replay attacks) in wireless sensor networks (WSNs). WSNs require protection to preserve the confidentiality and integrity of sensitive information as well as availability of the system. This is an important research issue because WSNs are used in critical applications such as military battlefield surveillance, industrial process monitoring and control, and machine health monitoring. This paper describes an alternative flow-based approach for visualizing security attacks in terms of depiction of behavioral interactions. It models security attacks in WSNs and contrasts this method with the sequence-based diagrammatic method. The comparison provides an initial appraisal of the technique with reference to a well-known process modeling methodology. The results indicate that the method can capture the interweaving of attack events to achieve a more complete and detailed picture necessary for better understanding.
Keywords: Collision attacks, conceptual model, information security, security requirements, UML sequence diagram, visualization, wireless sensor networks.

1 INTRODUCTION

The process of modeling involves artificial language used to express systems in an organized and methodologically interpreted way to capture the structural meaning of a system. The language can be in the form of diagrammatic representation of concepts and their relationships, with the representation built on visual depictions of activities, events, flow controls, functions, applications, actors, and their interactions.

It has been shown that such visualization offers significant benefits by providing an instrument for documentation, communication, and management purposes. In software engineering, visual notations are used extensively, in specifying requirements, design, and implementation. They play an important role in the development of more effective communication between technical and nontechnical participants than is possible with text [1]. Extensive research has been conducted in computer science with respect to the field of visualization; nevertheless, deficiencies still exist in the tools and methods used for capturing complex processes diagrammatically.

According to Moody [2],

> Visual notations are pervasively used in software engineering [SE], and have dominated both research and practice since its earliest beginnings. The first SE visual notation was Goldstine and von Neumann's program flowcharts, developed in the 1940s, and the ancestor of all modern SE visual notations […] This pattern continues to the present day with UML, the industry standard SE language, defined as "a visual language for visualising, specifying, constructing and documenting software intensive systems" [3].

ISSN: 2041-9031 (paper format), ISSN: 2041-904X (online), http://journals.witpress.com
DOI: 10.2495/SAFE-V3-N4-307-317

Unified Modeling Language (UML) [4,5] has been utilized in modeling systems. In this regard, Moody and Hillegersberg [4] caution that

> Both academic analyses and official revisions to the [UML] standard have focused almost exclusively on semantic issues […] The UML visual notations were developed […] by reusing and synthesising existing notations, […] based on expert consensus […] This is an inappropriate basis for making visual representation decisions […] The analysis reveals some serious design flaws in the UML visual notations.

This paper introduces a foundation for a general diagrammatic apparatus for modeling systems [referred to as the Flowthing Model (FM)] and, without loss of generality, contrasts it with the diagrammatic tools used in UML; however, in contrast to Moody and Hillegersberg's [4] criticism of UML which is based on the criteria of visual notation, this paper claims that UML additionally suffers from lack of an underlying unifying notion that ties together the rhythm and continuity of the narratives embedded in the sequence of modeled events. FM is built on the basic concept of flow that interweaves various streams to maintain continuity across parts and along the conceptual representation. 'Flow' here refers to the flow of things, as in the specification of flows of electricity, water, gas, and signals (e.g. telephone lines), added to the blueprint of a high-rise building.

Furthermore, in this paper, contrasting FM with UML is achieved by concentrating on modeling of security attacks. Information security attack modeling has been developed in various forms (e.g. attack trees, use cases, misuse cases, scenarios, and asset analysis [6]) and has been applied for many purposes (e.g. security requirements specification and identification of commonly occurring attack patterns). As a sample method of description, attack trees are used to analyze attacks through identification of security vulnerabilities and of compromises caused by attackers. An attack tree represents a damaging event. Branches of the tree elaborate the methods by which that event could occur.

In many attack-related studies, the focus of analysis is on improving decision making. Information is analyzed and combined with existing knowledge to produce a model for decision making. Decision making is an important factor in the construction of a rationality-based model that allows choosing from alternatives by moving through a series of steps. In contrast, the approach in this paper provides a descriptive model encapsulating structured knowledge captured from the phenomenon of attacking. 'Descriptive' here refers to identifying the progression of an attack through its various phases. This type of model facilitates understanding of and communication about the notion of *attack*.

Also, because of the generality of the problem of diagrammatically representing security attacks, this paper concentrates on the specific problem of modeling security attacks in wireless sensor networks (WSNs) medium access control (MAC) layer [7]. This field is selected because of the availability of recently published (2012) research that models attacks using a UML sequence diagram [8], indicating that UML methodology is the current method for this type of problem.

Additionally, WSN security is more complex in comparison with traditional network security, because of computational constraints of the nodes, conservative energy requirements, and an unpredictable communication channel and unattended operations [9]. Developing security mechanisms to protect WSNs requires understanding attacks through some type of modeling [10].

While maintaining general applicability, this paper focuses on a specific research study that utilizes only UML *sequence diagrams* to represent security attacks in WSNs. Such a

particular example of attacks in a specific field using a definite diagrammatic method provides an opportunity to contrast the features of FM against sequence diagrams representative of UML.

Section 2 describes the focus of this paper, modeling of security attacks, specifically the problem of modeling security attacks in WSNs. This is followed in Section 3 by a summary of the main features of the FM as applied to modeling of attacks in WSNs. In Section 4, a collision attack in such a network is represented in FM. Section 5 provides general conclusions and further work in this area.

2 WSN: PROBLEM

A WSN consists of specially distributed nodes with sensors, which can perform the monitoring of physical or environmental conditions (e.g. temperature, sound, pressure) by communicating with each other [11]. They are used in applications such as military applications (battlefield surveillance), industrial process monitoring and control, machine health monitoring, home automation, and intelligent transportation [12].

WSN requires protection to preserve the confidentiality and integrity of sensitive information, and availability of the system. Pawar *et al*. [7] used sequence diagrams to study WSN security by addressing the behavioral modeling of MAC security attacks in order to make the design of the layer protocols more efficient and secure. The MAC layer is important for the operation of a WSN since it regulates energy consumption, channel utilization, and network delay [13].

> Current research in WSN security has focused less on security on the MAC security. However, understanding the behaviour of MAC security attacks is important in order to develop secure mechanisms for the MAC layer […].
> Little research has been done in UML modeling of a WSN environment especially concerning the security [7].

According to Pawar *et al*. [7], UML has been chosen for analysis of security attack behavior [8] because it represents a well-known and standard methodology for modeling real-world objects. It also reflects the best engineering practices that have been utilized in modeling large and complex systems. Pawar *et al*. [7] then use the sequential diagram approach of UML to depict six known types of attack: collision attack, unintelligent replay attack, unauthenticated broadcast attack, full domination attack, exhaustion attack, and intelligent jamming attack.

This paper proposes to redraw these attacks in terms of FM in order to demonstrate that the proposed FM diagrammatic method can capture the interweaving of different events using the notion of *flow*. FM can be applied to most current diagramming methodologies as a tool for high-level specifications, as demonstrated in several previous publications (e.g. [14–18]).

For the sake of a self-contained paper, the next section briefly reviews FM; meanwhile, the example using the RTS/CTS protocol, also shown in the next section, is a new contribution.

3 FLOWTHING MODEL

The FM was inspired by many types of flows that exist in diverse fields, such as, for example, supply chain flows, money flows, and data flows in communication models. This model is a diagrammatic schema that uses ‘flowthings’. A flowthing is a thing that can be released, transferred, received, processed (in form), and created. For example, in communication, data are a flowthing that is released and transferred by the source, received in the destination, and

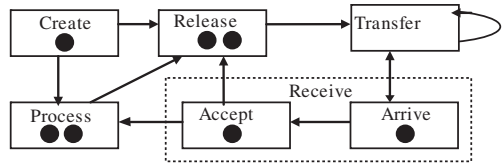

Figure 1: Flowsystem: The dark dots denote things at different stages of the flowsystem. The figure can be considered net marking (instantaneous location of all tokens in the net Petri net terminology).

processed (translated from one format to another), and this causes the creation of another piece of data (e.g. response). FM depicts processes using 'flowsystems' (Fig. 1) that comprise six stages, as follows:

Arrive: A flowthing reaches a new flowsystem (e.g. a data arrival buffer in a router).
Accepted: A flowthing is permitted to enter the system (e.g. no wrong address for a delivery); if arriving flowthings are also always accepted, Arrive and Accept can be combined as a **Received** stage.
Processed (changed): The flowthing goes through some kind of transformation that changes its form but not its identity (e.g. compressed, colored, etc.).
Released: A flowthing is marked as ready to be transferred outside the system (e.g. airline passengers waiting to board).
Created: A new flowthing is born (created) in the system (a data mining program generates a conclusion).
Transferred: The flowthing is transported to or from somewhere outside the flowsystem (e.g. packets reaching ports in a router, but still not in the arrival buffer).

These stages are mutually exclusive (i.e. a flowthing in the Process stage cannot be in the Create stage or the Release stage at the same time). An additional stage of 'Storage' can also be added to any FM model to represent the storage of flowthings; however, storage is not a generic stage because there can be stored processed flowthings, stored created flowthings, etc. Hereafter, a thing means a flowthing.

A flowsystem depicts the internal flows of flowthings in a system with the six stages and transactions among them. FM also uses the following notions:

Spheres and subspheres: These are the environments of the flowthing. A sphere can have multiple flowsystems in its structure if needed. A sphere can be an entity (e.g. a company, a customer), a location (a laboratory, a waiting room), and communication media (a channel, a wire). A flowsystem is a subsphere that embodies the flow and has no subsphere itself.

In this paper, it is assumed that the control of the movement of flowthings is embedded inside the stages (e.g. in Process: *if a flowthing fits a certain criterion, then it flows to Release*). In principle, there should be no difficulties in conceptualizing such a control at the edges, in the manner of Petri nets.

Triggering: Triggering is activation (denoted in FM diagrams by a **dashed arrow**) of one flow by another, initiating a different flow. It is a (causative) dependency among flows and parts of flows. A flow is said to be triggered if it is created or activated by other flows (e.g. the flow of electricity triggers the flow of heat), or activated by another point in the flow [e.g. *processing* logical formulas x and y triggers the *creation* of z (x AND y $\geq$ z) in the flowsystem of true formulas].

A flowsystem may not need to include all the stages; for example, an archiving system might use only the stages Arrive, Accept, and Release. Multiple systems captured by FM can interact with each other by triggering events related to one another in their spheres and stages. Triggering can also be used for events such as starting a flow system (e.g. outside start-up signal).

Example: The MAC layer in 802.11 standards, where transmission in a wireless node performs the following sequence, summarized from [19]:

Step 1: The sender checks whether the medium is idle or not; if so, after the Distributed Inter Frame Space (DIFS) units of time, it broadcasts a Request-to-Send (RTS) frame to the receiver address.

Step 2: The receiver waits for a Short Inter Frame Space (SIFS) unit of time; then it responds to the sender with a Clear-to-Send (CTS) frame.

Step 3: The sender receives the CTS frame; then it waits for another SIFS unit of time before sending the data frame to the receiver.

Step 4: Finally, when the receiver successfully receives the data frame, it waits for an SIFS unit of time and also returns an Acknowledgement (ACK) message to the sender.

Figure 2 shows how data are exchanged using RTC/CTS protocol. Figure 3 shows the corresponding FM representation. There are two spheres: sender and receiver. Each sphere

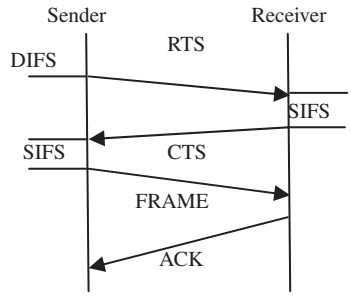

Figure 2: How data are exchanged using RTC/CTS (adapted from [19]).

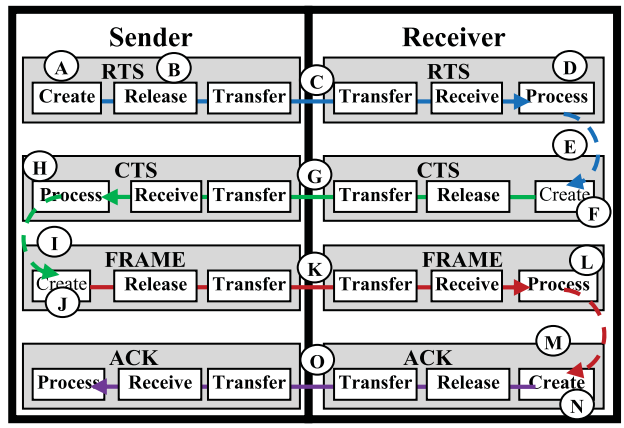

Figure 3: FM representation corresponding to Fig. 2.

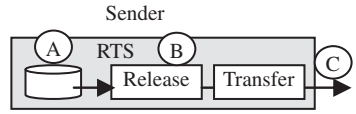

Figure 4: Retrieval of a stored RTS.

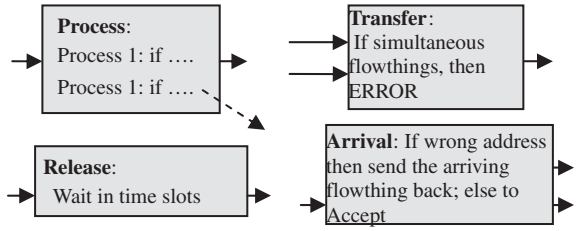

Figure 5: Sample rules and constraints.

includes four flowsystem subspheres: RTS, CTS, FRAME (data), and ACK; thus, the sender and receiver handle these four flowthings by creating, receiving, processing, releasing, and transferring. It is possible to replace 'Create' with retrieval of RTS from storage if RTS is not constructed (Fig. 4).

In Fig. 3, the sender creates an RTS (circle A in Fig. 3), then releases it (B). Releasing means being marked for transmission, but the RTS is not actually transferred; thus, this is the suitable place for DIFS (time delay not shown in the figure). The RTS flows to the receiver (C), where it is received and processed (D) to trigger (E) the creation of a CTS (F). The CTS flows to the sender (G) to be processed (H), triggering (I) the creation (J) of a FRAME. The FRAME flows to the receiver (K), where it is processed (L) and triggers (M) the creation of an ACK (N), which flows to the sender (O).

Each stage in the flowsystem can include all types of features such as constraints synchronization and operational aspects, as illustrated with some examples in Fig. 5.

In the FM-based depiction, a complete semantic picture of the communication phenomenon, the sender and receiver can generate (create), process, release, receive, and transfer data, not merely send and receive. Messages trigger each other and are 'physically' connected (solid and dashed arrows), not just with implicit time-sequenced connections. Additionally, the flows of flowthings (RTS, CTS, FRAME, and ACK) are clearly separated or connected by dashed arrows. The representation also embeds a great deal of semantic material that can be used to describe dynamism and control (e.g. constraints, Fig. 5). Yet, the FM representation is characterized by simplicity provided by the repeated application of flowsystems.

4 COLLISION ATTACK

In this type of attack, a malicious node causes collisions with the transmissions of neighboring nodes by sending a short noise packet, thus causing a great deal of disruption in the network operation [20]. Pawar *et al*. [7] describe the events of this type of attack as follows:

- An external attacker initiates a collision attack through malicious node 3.
- Once the attack is initiated on node 3, it will start to send noise packets to all nodes in the network. It will increase traffic in the network, causing the channel to become busy as it performs this activity.

- Node 1 detects an event and sends an RTS packet to node 2. At the same time, the malicious node 3 also generates a noise packet and forwards it to node 2. Both packets will reach node 2 simultaneously and cause a collision.
- Again, node 1 detects the event and checks channel availability by exchanging RTS (request to send) and CTS (clear to send) with node 2. Once node 1 receives a CTS from node 2, node 1 starts to send data packets toward node 2. If, at the same time, malicious node 3 also sends noise packets toward node 2, collisions will happen in the network.
- Malicious node 3 is continuously generating noise packets that try to use the channel so a collision will take place. This collision of packets leads to retransmission of the packets, which in turn leads to increasing energy consumption [7].

Figure 6 shows a partial picture of the corresponding sequence diagram given by Pawar *et al*. [7]. Notice how semantically disturbing the sequence diagram is; for example, the same type of a symbol – a solid arrow – simultaneously represents (1) an attack, (2) different types of flows, and (3) detection of events. Also, the sequence diagram mixes different types of collisions, as will be clear next, thus making it difficult to understand the attack.

Figure 7 shows the FM-based logical map of flows in the four-node situation in this example. The following discussion aims at furthering understanding of the events related to the attack.

Figure 7 includes the four nodes and the flowthings that are transmitted between them: FRAMES (noise or data), RTSs, and CTSs. It is assumed that the noise is of type 'data' (noise FRAME) because it is generated by a user (the hacker), not by the system, as in the case of RTS and CTS. Figure 7 depicts only the flows described in the sequence diagram of Fig. 6. Later in this paper, a more complete representation will be developed.

- Node 3 creates and sends noise to nodes 1 (circle A), 2 (B), and 4 (C).
- Node 1 sends RTS to node 2 (D).
- A collision occurs between RTS (D) and noise passing from node 3 to node 2 (B).

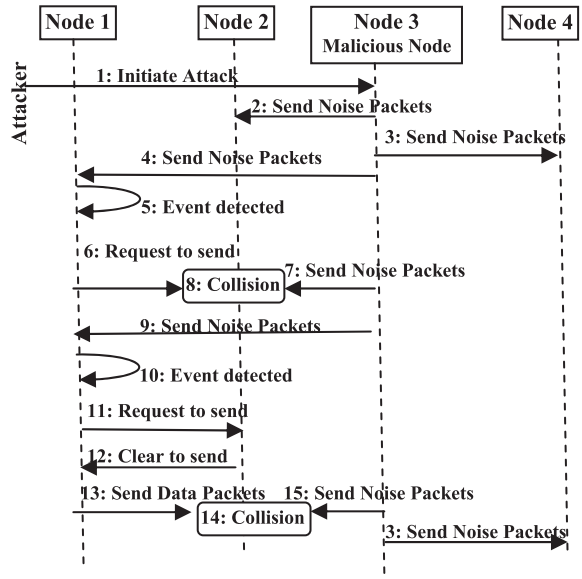

Figure 6: Sequence diagram of collision attack.

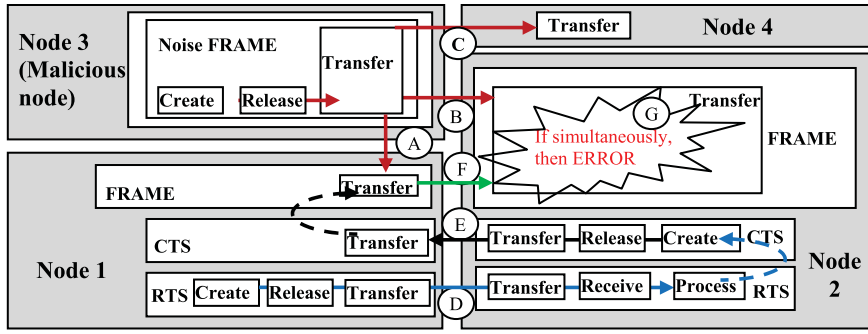

Figure 7: Logical map of flows in the example.

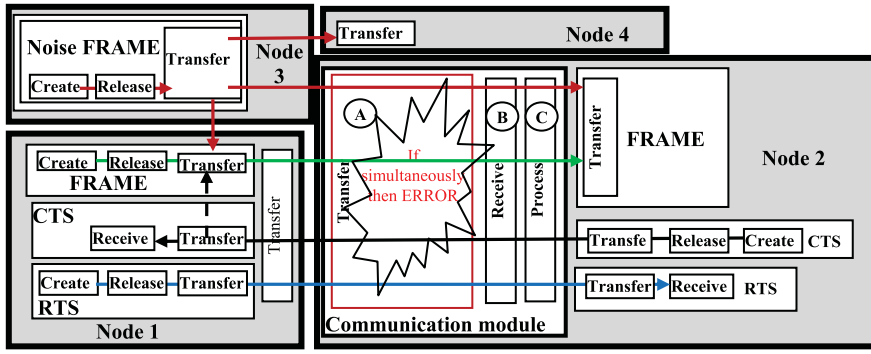

Figure 8: Collisions among all types of messages.

- Node 1 sends RTS to node 2 (D) again.
- Node 2 sends CTS to node 1 (E).
- Node 1 sends FRAME: data packet (F).
- A collision occurs between FRAME (F) and noise passing from node 3 to node 2 (B). This collision is shown in Fig. 7 (G).

It can be observed that the sequence diagram in Fig. 6 is only a partial diagram, because collisions occur between all messages sent by all nodes to all other nodes. The scenario of Fig. 6 include only a few instances of these collisions. If all occurrences of collision were depicted, the sequence diagram would be very long.

Furthermore, the sequence diagram depicts a single type of collision (e.g. collision between RTS and noise) between noise and FRAME, rather than collisions among messages. Thus, in Fig. 7, collisions between different flowthings (e.g. FRAMES and RTS) do not appear. This does not help in understanding the total picture of the attack, especially the logical 'location' of the attack, as will become clear next.

The reason for treating all flowthings as a single type in the sequence diagram is that the communication system, for practical reasons, does not distinguish among different types of messages (flowthings). Accordingly, Fig. 7 is redrawn as Fig. 8 to distinguish between internal senders/receivers and the system that performs the actual communication. In the new Fig. 8, node 2 has a single (logical) interface (flowsystem) with the communication system (circles A, B, and C). Upon receiving a message (regardless of whether it is RTS, CTS, or FRAME)

from outside, the communication system of node 2 processes it and directs it to the appropriate application or system program. Similarly, upon receiving a message (regardless of whether it is RTS, CTS, or FRAME), it sends it to the appropriate outside party. While each internal flowsystem handles one type of flowthing, the communication module handles all types as *messages*; hence, collisions can occur among these messages regardless of their nature.

Note that the communication channel could be drawn as a sphere with flowsystems installed among nodes in the FM description, but those flowsystems would be irrelevant to the depiction of the attack under discussion.

Even though the FM in Figs 7 and 8 follows the sequence diagram in the given particular occurrence of an attack where a collision appears in the communication between nodes 1 and 2, this is not a general modeling of this type of attack. To make the model broader, it is assumed that the network consists of n nodes plus one malicious node. Figure 9 shows the resultant FM representation of the collision attack.

The hacker (A) sends noise to the communication module of the malicious node (it is possible to model the situation such that the hacker takes over the communication module). Accordingly, the malicious node starts bombarding other nodes with noise (B). Focusing on node 2, the figure shows this node receiving a stream of noise (C). With this constant noise reaching node 2 (D), there is a high probability of collisions (E), with messages arriving from node 1 (F), node 3 (G), and other nodes. Node 2, itself, contributes to the problem by sending messages (RTS, CTS, FRAMES, etc.) coming from different applications and system programs (H) to node 1 (I), node 3 (J), and other nodes.

Figure 9 certainly presents a more complete conceptual picture, for the purpose of understanding the collision attack, than a sketch such as a sequence diagram. Suppose it is required to model the event in the sequence diagram (Fig. 6), where *13: Send Data Packets* (denoted as FRAME), sent from node 1 to node 2, collides with *15: Send Noise Packets,* sent from the malicious node to node 2. Figure 10 shows the depiction of this event.

In the figure, node 1 sends RTS (A), which flows to node 2 (B) to arrive at its RTS flowsystem (C). The request is processed to trigger (D) the creation of CTS (E), which flows to node 1 (F) to be received by the flowsystem of CTS (G). There, it is processed to trigger (H)

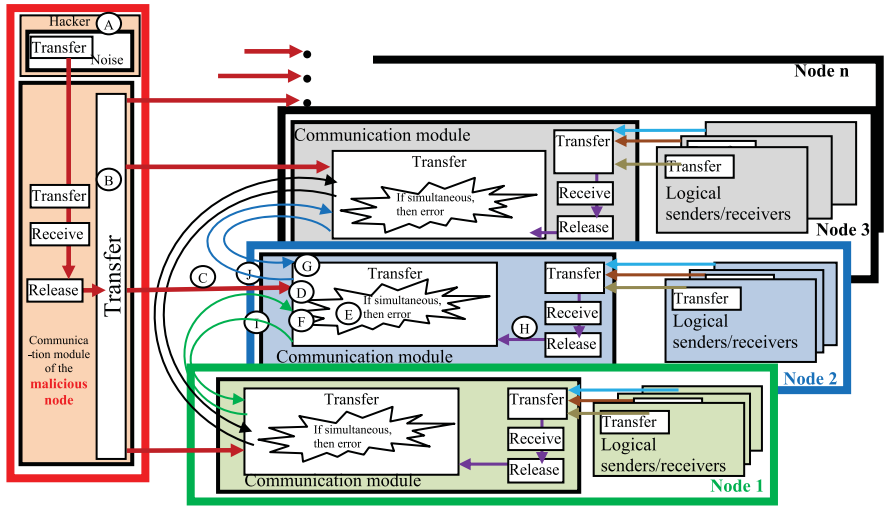

Figure 9: General model of the collision attack.

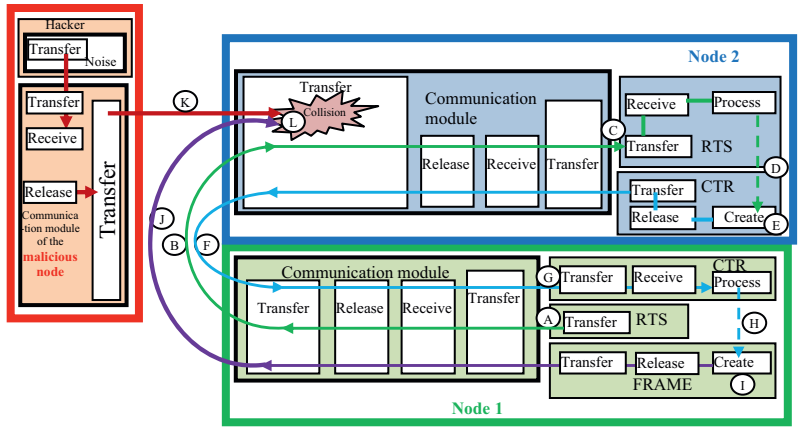

Figure 10: Modeling of collision of FRAME from node 1, with noise from the malicious node.

the creation of FRAME (I). The data frame flows to node 2 (J) to arrive simultaneously with the noise flowing from the malicious node (K), causing collision (L).

5 CONCLUSION

Recently, UML sequence diagrams have been used to model security attacks (e.g. collision attacks and unintelligent replay attacks) in WSNs. This paper describes an alternative flow-based approach (FM) for visualizing security attacks in terms of depicting behavioral interactions. It models security attacks in WSNs and contrasts this model with the sequence-based diagrammatic method.

The comparison provides an initial appraisal of the FM method with reference to the well-known UML process modeling methodology. It is shown that modeling using sequence diagrams is semantically disturbing, for example, where the same type of a symbol—a solid arrow—simultaneously represents (1) an attack, (2) different types of flows, and (3) detection of events detection. The sequence diagram also mixes different notions, such as types of collisions, thus making it difficult to understand the attack.

The results indicate that the FM diagrammatic method can capture the interweaving of different events in the attacks to achieve a more complete picture necessary for better understanding. The FM representation of attack progression can be applied to modeling different types of computer and communication attacks. Based on FM conceptualization, it is possible to characterize weak points and develop a map of vulnerabilities in the system. Such a methodology provides a base for analysis in the fields of threat modeling and secure software development.

Further work would model different types of attacks in FM and apply the resulting model to description of actual computer attacks. Another aim is to utilize the FM representation of attacks in other applications, such as in design of protection strategies and development of security policies.

REFERENCES

[1] Avison, D.E. & Fitzgerald, G., *Information Systems Development: Methodologies, Techniques and Tools*, 3rd edn., Blackwell Scientific: Oxford, UK, 2003.

[2] Moody, D.L., The physics of notations: towards a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, **35(6)**, pp. 756–779, 2009. doi: http://dx.doi.org/10.1109/TSE.2009.67

[3] OMG, *Unified Modeling Language Version 2.0: Superstructure*, Object Management Group (OMG), 2005. http://www.omg.org/cgi-bin/doc?formal/05-07-04

[4] Moody, D.L. & van Hillegersberg, J., Evaluating the visual syntax of UML: an analysis of the cognitive effectiveness of the UML suite of diagrams. *Proceedings of the 1st International Conference on Software Language Engineering* (SLE), Toulouse, France. Springer Lecture Notes in Computer Science, 2008.

[5] OMG, *Unified Modeling Language Specification*, *Version 1.5*, Object Management Group, March 2003.

[6] Gordon, D., Stehney, T., Wattas, N. & Yu, E., System Quality Requirements Engineering (SQUARE): Case Study on Asset Management System, Phase II. Carnegie Mellon University, available at: http://www.cert.org/archive/pdf/05sr005.pdf, 2005.

[7] Pawar, P.M., Nielsen, R.H., Prasad, N.R., Ohmori, S. & Prasad R., Behavioural modelling of wsn mac layer security attacks: a sequential UML approach. *Journal of Cyber Security and Mobility*, **1(1)**, pp. 65–82, 2012.

[8] Hong, S. & Lim, S., Analysis of attack models via unified modeling language in WIRELESS SENSOR networks: a survey study. *IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS2010)*, pp. 692–696, 2010.

[9] Chen, X., Makki, K., Yen, K. & Pissinou, N., Sensor network security: a survey. *IEEE Communications Surveys & Tutorials*, **11(2)**, pp. 52–73, 2009. doi: http://dx.doi.org/10.1109/SURV.2009.090205

[10] Lopez, J., Roman, R. & Alcaraz, C., Analysis of security threats, requirements, technologies and standards in wireless sensor networks. In *On Foundations of Security Analysis and Design*, eds. A. Aldini & R. Gorrieri, Springer, LNCS 5705, pp. 289–338, 2009.

[11] Dargie, W. & Poellabauer, C., *Fundamentals of Wireless Sensor Networks: Theory and Practice*, John Wiley and Sons, ISBN 978-0-470-99765-9, 2012.

[12] Sohraby, K., Minoli, D. & Znati, T., *Wireless Sensor Networks: Technology, Protocols, and Applications*, John Wiley and Sons, ISBN 978-0-471-74300-2, 2007. doi: http://dx.doi.org/10.1002/047011276X

[13] Bachir, A., Dohler, M., Watteyne, T. & Leung, K., MAC Essentials for wireless sensor networks. *IEEE Communications Surveys & Tutorials*, **12(2)**, pp. 222–248, 2010. doi: http://dx.doi.org/10.1109/SURV.2010.020510.00058

[14] Al-Fedaghi, S., Scrutinizing the rule: privacy realization in HIPAA. *International. Journal of Healthcare Information Systems and Informatics (IJHISI)*, **3(2)**, pp. 32–47, 2008.

[15] Al-Fedaghi, S., Conceptualizing effects, and uses of information. *Information Seeking in Context Conference* (ISIC 2008), September 17–20, Vilnius, Lithuania, 2008.

[16] Al-Fedaghi, S., Software requirements as narratives. *Third International Conference on Information, Process, and Knowledge Management*, February 23–28, Gosier, Guadeloupe, France, 2011.

[17] Al-Fedaghi, S. & Al-Babtain, B., Modeling the forensics process. *International Journal of Security and Its Applications*, **6(4)**, pp. 97–108, 2012. doi: http://dx.doi.org/10.4018/jhisi.2008040104

[18] Al-Fedaghi, S., Annotations of security requirements. *International Review on Computers and Software* (IRECOS), **7(4(A))**, pp. 1470–1477, 2012.

[19] Sakshat Virtual Lab, Simulating a Wi-Fi Network, available at http://virtual-labs.ac.in/cse28/ant/ant/5/theory/ (accessed 2013).

[20] Reindl, P., Nygard, K. & Du Xiaojiang, D., Defending malicious collision attacks in wireless sensor networks. *IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC)*, December 11–13, pp. 771–776, 2010.