Securing Ad-hoc On-Demand Distance Vector Protocol in Wireless Sensor Networks

Working with What the Node Can Offer

Shaikha Alkhuder
Computer Engineering
Kuwait University
Kuwait City, Kuwait
s.alkhuder@ku.edu.kw

Abstract*—Wireless sensor networks (WSNs) are considered to be one of the most important technologies of the 21st century. As a result, WSNs have been used in numerous applications in industry, health monitoring, environmental monitoring, and other related fields. However, the unprotected nature of WSN protocols such as the Ad-hoc On-Demand Distance Vector (AODV) Protocol makes them prone to malicious attacks. One such attack is the replay attack. A single sensor node has limited computation and communication capabilities, but processing routing information through data structures with acceptable time and space complexity can lead to secure data acquisition and sensing. Sensor nodes have limited energy resources, so this attack can have a serious impact on network functionality. In this work, Bloom filters are used to identify the legitimacy of a packet. Sensor nodes will be able to distinguish between legitimate and replayed packets along a path, thus improving a node's decision as to whether it should transmit or discard the received packet.

***Keywords*—WSN; protocol; security; bloom filters; energy; data structure; complexity; reliability;**

I. INTRODUCTION

Networks have been rapidly developing throughout the last few decades, varying from simple host-server structures to a thousand-node WSN architecture. In parallel, networking risks have grown as well. The more sophisticated the network, the wider the range of potential attacks that could be performed to sabotage the dedicated functionalities of that network. Replay attacks are among the most common and easily performed attacks. This line of research studies the efficiency of Bloom filters in addressing replay attacks performed on wireless sensor networks and how they impact energy, throughput, and numbers of packets exchanged in the network. WSNs, AODV routing protocol, replay attacks, and Bloom filters are all concepts that make up the building blocks of this work, so they are discussed in detail in the following subsections to give readers a better understanding of the overall ideas and approaches.

A. Wireless Sensor Networks (WSNs)

WSNs are the result of advancements in the technology fields of micro-electro-mechanical systems (MEMS) and other related areas of research, such as communication networks and embedded systems. A low-cost, less power demanding, space-efficient network structure is available for multiple uses and purposes [9]. The field of WSNs is a fertile source of applications in industry, military, health practices, scientific research, and other sectors. Composed of inexpensive sensors triggered by the surrounding environment, WSNs can collect meaningful data, which can then be analyzed for deployment purposes. Data collected by sensor networks are basically what the sensor nodes detect. Sensor nodes, as illustrated in Fig. 1, can detect light, heat, humidity, sound, weight, or any other measurable form of data that can be translated to parameter values according to what type of sensor network application is in use. Mobilizers and power generators are not always utilized in sensor nodes [1]. Thus, it is crucial to have WSNs engineered in a matter that takes into consideration the previously mentioned elements in order to maintain an acceptable level of performance. WSN nodes can be deployed in random or fixed positions to serve the purpose of deployment. Unlike ad-hoc networks, WSNs are more densely used, but at the same time, they are prone to failures in higher frequencies than ad-hoc distributions. In addition to that, WSNs are based on broadcast communication as shown in Fig. 2. Nodes gradually build their routing tables and start transmitting data based on the routes they discover, whereas ad-hoc networks are more based on point-to-point communication [1]. As for the size of WSNs, this type of network consists of node numbers ranging from tens to thousands in order to observe specific real-time on-location alterations that may be on land, beneath the ground, or in wet environments, such as underwater or in humid habitats [14].

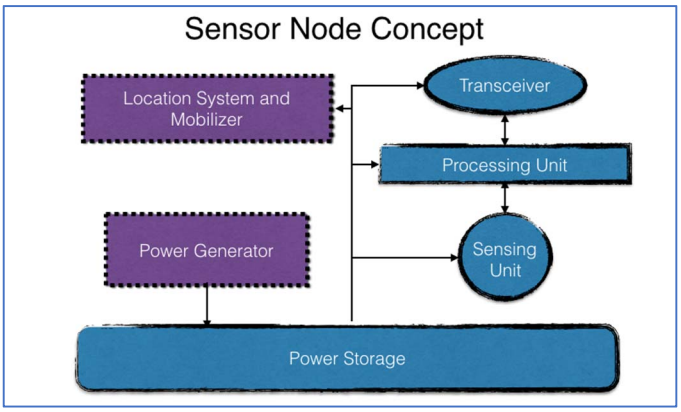

Fig. 1. Simplified components of a sensor node

978-1-5386-4680-9/18/$31.00 ©2018 IEEE

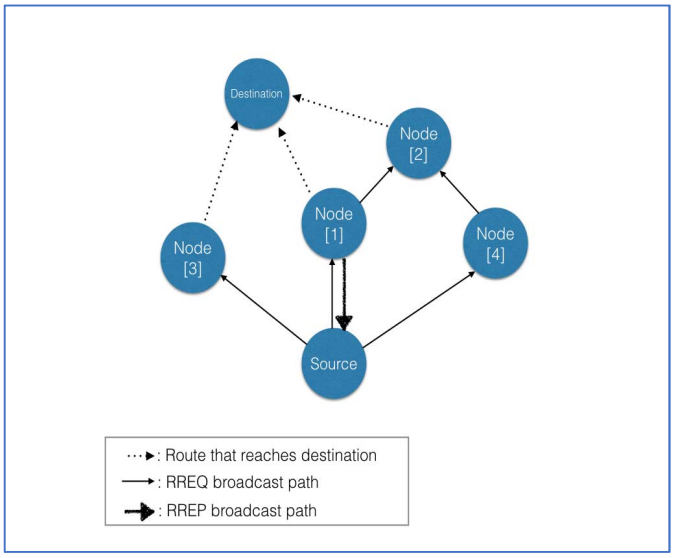

Fig. 3. RREQ and RREP in an AODV network

WSNs can be deployed in terrains and disaster locations that are absolutely inaccessible. Thus, another important feature of WSNs is self-organization, which allows the nodes to form ad-hoc networks. Nodes in WSN are also capable of doing minor processing of data to minimize the size of data messages needed to be exchanged. For example, having sensor nodes close to hurricanes can improve environmental research. Monitoring a patient from a distance for any abnormal symptoms is not only relieving for the patient's mind, but also more comfortable for the doctor in performing his/her duties more efficiently. These are some of the science and humanity serving purposes of WSNs [1].

Although WSNs deliver many beneficial services, they are composed of sensor nodes, which have limited resources: battery life, transmission bandwidth, and RAM space, which are all defined to a certain amount. Factors like hardware constraints, production cost, scalability, and fault tolerance can impact the design of such networks [1]. This explains why security measures must be considered carefully lest the nodes be drained sooner than expected. Additionally, WSNs, in standard scenarios, are designed to perform while unattended. This makes energy the most important factor, and it is the main reason why WSN communication protocols are designed to value energy and transmission rates the most [10].

Unlike ad-hoc networks, nodes in sensor networks are vulnerable to failures because they are frequently deployed in massive numbers over wide areas, and sometimes with mobility features. Although sensor nodes are prone to physical damage, fault tolerance solutions are not always trivial and easy to apply. This is crucial because the WSN must sustain its levels of performance in the event of partial failure [1].

In typical networks, security is directly concerned with transmitting untampered, authenticated, confidential, and available data to the receiver host. Normally, this is performed by a higher layer schema, such as SSL or SSH. This seemed suitable for typical networks because most end-to-end communications are carried out without the intermediate nodes needed to access the content of transmitted data. However, WSNs have a different concept of transmission: the perspective of security changes, because contents of data sent from one node to the other need to be accessed by the intermediate nodes; therefore, security must be implemented at multiple layers, and routing protocols should be designed in a way that adopts these criteria [10].

B. Ad-hoc On-Demand Distance Vector (AODV) Routing Protocol

There are many routing techniques and protocols used in ad-hoc networks in general, like: flooding, gossiping, directed diffusion, low-energy adaptive clustering hierarchy (LEACH), sensor protocols for information via negotiation (SPIN), and Ad-hoc On- Demand Distance Vector (AODV), which is the routing protocol used to carry out the simulation in this research [13] [1]. AODV routing protocol is one of the common reactive ad-hoc routing protocols used in WSNs [5]. The AODV network depends on a request-reply (RREQ/RREP) mechanism to discover routes leading to other nodes in the AODV network as shown in Fig. 3. A source node broadcasts an RREQ packet and then waits within a specific time frame for any other AODV node to reply with a route to the destination node. The node that has a route to the destination, which can also be the destination node itself, can reply with an RREP through a unicast to the source AODV node where it shares the route for the destination AODV node. Once one node has received the RREP packet, it can start transmitting the data packet according to the route information it received. To assure freshness of the routing information, AODV uses sequence numbers. Initially, every AODV node creates a UDP socket with its kernel to start sending and receiving packets with other AODV nodes through a particular port, normally port 654. Other ports are used for multicast and routing table updates, which occur periodically or on specific events like node death [13].

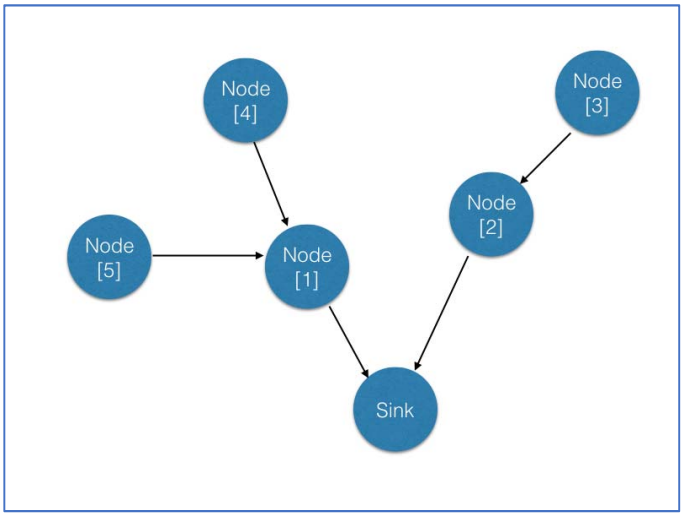

Fig. 2. Data Aggregation in a WSN

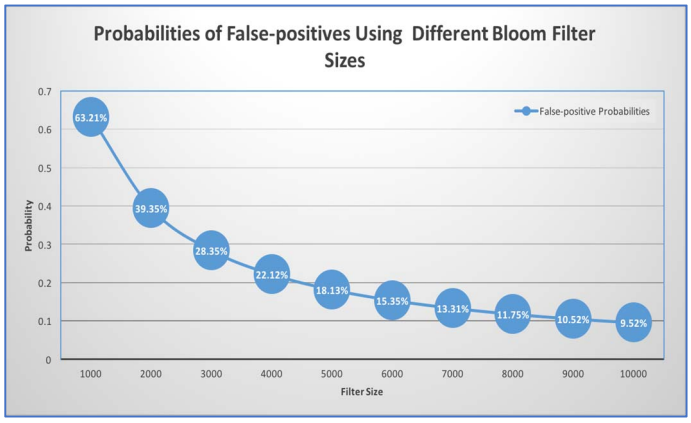

Fig. 4. Probabilities of false-positives using different bloom filter sizes

AODV routing uses HELLO messages to test node links for any potential link breaks. With the assumption that an active node would send periodic HELLO messages, if a single AODV neighbor node fails to receive HELLO messages multiple times, then that AODV neighbor is assumed to have a broken link [5]. Data in this simulation is transmitted over a radio medium, and AODV nodes are assumed to have fixed (immobile) positioning to avoid unexpected link breakage.

C. *Replay Attack*

A replay attack, also called a copycat attack, in sensor networks is a common attack that aims to fool the MAC recipient into receiving and forwarding the replayed packet to the following sensor network node and then persuade the destination node to take in the replayed packet as a legitimate one. This can be done within the same protocol run or in another run in a different protocol. Replay attacks can be accomplished by manipulating packet contents like headers to hide the actual sender and impersonate the original source, or without changing any of packet contents and retransmitting a copy of the packet in a different time frame, in which case the packet is sent as is.

An example of manipulating packet contents to perform a replay attack is to replace the node (A) address field in the header of the packet with a spoofed MAC address of a legitimate sensor network node that is not part of the original node (A)-node (B) connection route. Then, it sends the replayed packet from the sensor node with the replaced address. Having Bloom filters in the model of sensor networks can be used to detect duplicate nodes that are replayed without modifications [7]. However, this line of defense is bypassed by manipulating the contents of the packet. Signatures step into the picture at this point.

D. *Bloom Filters*

Most known for being utilized in databases, Bloom filters are randomized structures that test the membership of entries that filter through them. Only recently have Bloom filters been utilized with increasing popularity in computer networking, although the use of these logical structures stretches back to the 1970s when Burton Bloom introduced them [3]. Bloom filters were used for various applications, such as dictionaries, UNIX spell checkers, and storing insecure passwords to prevent them. Although Bloom filtering may produce false-positives, its space efficiency compensates for that precision deficiency [4].

To mathematically explain Bloom filters, the following is noted: A Bloom filter is represented as a set B that translates to an array a that contains n elements $B = \{ e1, e2, e3, e4, ..., en\}$. All elements are initially set to *0*. Array indices here are also referred to as bits. The Bloom filter then uses a number of m hash functions $h1(x), h2(x), ..., hm(x)$, that produces result values ranging from *0* to n. For the purpose of easier mathematical comprehension, hash function results are assumed to be uniformly distributed. To examine whether x is an element of set B, x is hashed m times. If the resulting hash, which happens to indicate an array bit, carries a value of *1*, then x is an element of set B. If the array bit indicated by the hash result carries a value of *0*, then x is not a part of the set B. The Bloom filter then changes those bits to *1* in order to make a record of the new element. The rate of false-positives can be estimated using mathematical formulas of that matter. The following equation describes the approximated rate of false positives P with respect to the number of hash functions K, number of set elements N, and filter size M in bits:

$$\mathbf{P = (1- e (-KN/M))}$$

However, the larger the Bloom filter array size, the less the rate of false-positives it may produce. Fig. 4 shows how different bloom filter sizes can produce varying false-positive probabilities considering a constant number of hash functions K=5. The simple structure of the Bloom filter makes it easy to implement and access [4]. The time complexity of using Bloom filters is estimated to be $O(1)$, assuming fixed values for K and P [6]. In Fig. 5 (a), two elements x and y are filtered to see if they belong to a certain set S or not. Element x is a member of the set because all corresponding hash result bits bear a value of *1*. On the other hand, y is not a member of the set because one of the hash result bits holds a *0* value in the array of the resulting hash bits as shown in Fig. 5 (b).

II. Related Work

Literature shows varying methods to avoid or defend against replay attacks. In an effort to encompass most of the possible current solutions suggested for mitigating the effect of replay attacks, a number of counter schemes are discussed in the following subsections. Many approaches had crypto-based security design principles and features, while other schemes mainly utilized hashed data digests to make use of both random numbers and the identities of network nodes in order to improve the information protection concept.

A. *Design Principles Applied to Cryptographic Protocols*

The protocols' design principles that are used to avoid replay attacks depend upon processes like using cryptographic functions for type-tagging exchanged messages, whole information hashes, and unique session key production without mutual trust. This approach compensates for the decreased security with low-cost robustness. It focuses on the structural flaws of the suggested cryptographic protocols, and it emphasizes the importance of the implementability of any

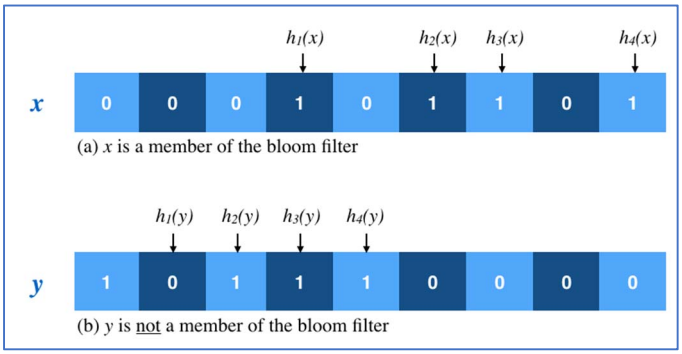

Fig. 5. Example of Bloom filtering elements x and y

technique that can support the robustness of the cryptographic protocol. Type tagging, which is attached to the message, indicates identifiers for: protocol, transmission step, message subcomponent, primitive data type, and the protocol run. A good example of how this approach works is the X.509 messages, which contain all the mentioned identifiers except for the protocol run identifier. The X.509 standard also includes the public key certificates but not the protocol authentication messages. This is where the protocol-related unique hash function is used to implicitly type-tag exchanged messages [2].

Another approach to defend against replay attacks was an introduced scheme [11] that guarantees the following two statements: preventing the original network nodes from accepting packets that are not as claimed, and the fact that if a packet is claimed to be legitimate, it cannot be replayed as having the legitimate properties. Ref. [11] proposed this scheme and then proved its correctness. If one node accepts a replayed packet, both receiver and sender are compromised, so replay attacks can exist on both nodes or on none. This approach also considers cryptography and type-tagging as well in the preceding approach.

Both of these proposed solutions discuss the route of packets within the process of securing the connection or connection start-up. In these two approaches, replay processes happen after the communication is secure and done, not before. In wireless networks, replay attacks are addressed by different solutions, but most of them neglect data-packets replay attacks although they tackle control-packet replay attacks [7]. Ongoing literature reviews show that research in the field of security principle design has been continuously developing general schemes to lower incident rates of impersonation and replay. The proposed techniques aim to enhance the robustness perspective against several types of network attacks, including replay attacks.

B. Hashing Chains to Defend Against Replay Attacks

TESLA, on the other hand, is a technique that addresses replay attacks in broadcast communications. The TESLA protocol is an abbreviation for Timed Efficient Stream Loss-tolerant Authentication protocol. This protocol uses chains of one-way hash functions and deferred traffic authentication key releases. The one-way hash functions operate as self-authenticating values. Because TESLA applies a low significance to time synchronization between nodes, this technique treats time intervals as cryptographic keys, and this is how receiver nodes authenticate the sender. A receiver node directly knows if the sender produced a message or not, because it can identify if the sender has reached that time value or if it hasn't possibly reached it yet. Because TESLA does not depend on exact time synchronization, it has no demand for the highly precise time synchronization requirement that other protocols may not function properly without [12].

Yet, this approach bears the potential problem of what might happen if the hash key was disclosed prior to the packet authentication done by the forwarding nodes in which case destination nodes will accept the lately received packets signed by the attacker node that, with good chances, may have obtained the hash key to produce authenticated packets. This approach may also suffer a replay attack produced by monitoring the clock of a particular sender node and syncing with it to produce packets within parallel time intervals.

C. Hop-by-Hop Authentication Mechanisms

Another method to counter replay attacks through secure data transmission was a simple protocol called ALPHA [8]. ALPHA abbreviates as Adaptive and Lightweight Protocol for Hop-by-hop Authentication. Using ALPHA, an initial packet is sent to the destination node to have all nodes on the way check the integrity of the large data packets delivered. The way that ALPHA is executed allows creating protected paths between sender and receiver through multiple hops, and with the authentication of each hop, early detection of replayed packets has been made trivial. ALPHA also operates in three modes to cover scenarios of sending control packets or large data. The three modes of operation can also be combined in operation to create better performance. A performance test of ALPHA was carried out on different types of networks, and one of those was WSN. Public key cryptography showed poor performance in terms of delay. Symmetric keys showed better performance but would not prevent a replayed packet from going further into the network [8]. In this approach, some replay attacks were established without changing packet contents, so checking the integrity of the packets would not detect the replay attack. In the Copycat Online Prevention System (COPS) design [7] replay attacks were assumed to be executed in a way that the replayed packet header was not changed or marginally manipulated, and the packet was sent as is. This type of attack is also called a copycat attack. The approach then presents a combination of

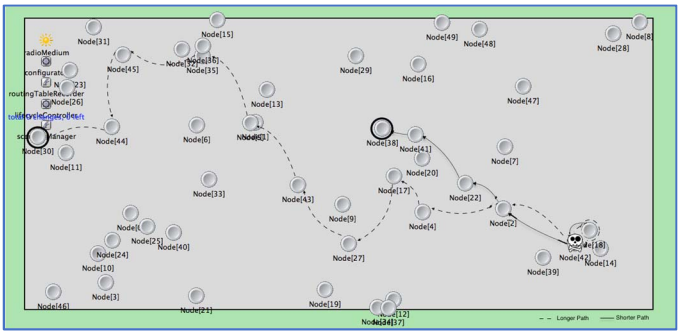

Fig. 6. Simulated AODV network with N=50 Nodes

digital signatures and Bloom filters within the node's soft structure to counter the attack. The testing of this approach is done to prove lightweight functionality, and it was applied to a testbed of less than 50 nodes. It was observed that with the increase of the path length between event sources and sinks, the degradation caused to the network by the replay attack increased as well. The replayed packet in this case traveled further in the network using extended numbers of hops across WSN nodes.

Simple time-stamping and packet counters can also be used against replay attacks, but they require highly accurate synchronization between network nodes, which is hard to obtain in a dynamic environment without dedicated pieces of hardware.

III. Methodology

OMNet++ is used to simulate an AODV WSN in three different stages: a normal AODV network, an AODV network suffering replay attacks, and an AODV network enhanced with Bloom filters to mitigate the effect of replayed packets. The implementation of the transport layer mounted on each AODV node was modified to include one Bloom filter for each node structure in the simulated WSN. The Bloom filter class was designed to receive a string (Message), filter it, and return a result that confirms that the message belongs to the set of the previous elements added to the filter, hence it is a replay packet, or it is a newly introduced element, in which case it is added to the set for future message matching. The Bloom filter class was implemented using simple C++.

The simulation can be launched with initially stated values, because the initial energy, transmission range, sending interval, and other values were required to start the interaction between the AODV nodes. These values reflect on the result readings and plots produced after each simulation regardless of their precision or resolution. Nodes in all simulation runs are randomly placed in fixed locations where they are assigned zero-mobility to have minimum link breakage. The sink node is presumed to have an extended energy storage as it is expected to stay alive and receive messages from any other node in the WSN. For each run, a group of sample nodes, including sink, source, and some intermediate sensor nodes, are chosen to measure their level of average residual energy and received packet count.

Fig. 6 is a random placement for 50 AODV sensor nodes in a 1500 x 700 m^2 simulated area. For this run, node [38] is set as the sink and node [18] as the source sensor node sending periodical messages every 10 seconds. The typical energy behavior for this AODV WSN collecting data through a short path is indicated by the solid line as shown in Fig. 6. The original AODV WSN behavior is captured to observe comparison at later stages. For a typical AODV network composed of N = 50 nodes with no replay attacks in execution, and with a relatively short path of 4–5 hops, Fig. 7 illustrates the average energy values for nodes along the path. Excluding the sink that has a larger energy storage set to it, the other intermediate nodes seem to lose energy exchanging control messages and forwarding UDP data packets. Each node starts by exchanging AODV- RREQ and AODV-RREP with neighbors within its transmission range to build up routing tables. Normally, a node-like node [41] would lose a significant amount of energy due to the higher number of neighbors (5 nodes) in its range that it needs to be aware of. The source node loses energy the most because it does not only participate in the network convergence, but also sends periodic readings to the sink.

Each node in this simulation has been equipped with its own Bloom filter. Bloom filters in these nodes are designed to hash packets with 5 hash functions and then place the results in the cells of a 4000-byte array. The sensor node can then categorize each UDP packet as either legitimate or replayed using these Bloom filters, as explained earlier. If replayed, the node immediately drops the packet without processing it. If not, the node forwards the packet to its next hop. Although the Bloom filters are able to identify replayed packets, in some cases, where the number of packets generated is large enough to occupy all filter entries, false positives may occur. False positives are events when legitimate packets are detected as replayed, and consequently dropped. The dropping of legitimate packets can decrease throughput, disconnect the stream of useful data, and weaken the performance of this WSN. To have a tolerable false-positive rate, the Bloom filter needs to flush its entries in a frequent manner. Flushing is simply the process of returning all filter entries back to initial 0's. In order to maintain an approximate 20% probability of false-positives, the Bloom filter of a single node needs to flush at least once after 200 different packet recordings given that each packet will occupy 5 indexes out of the total 1000.

The detection may be delayed in alternating paths, but still accomplished by another node down the path to the sink. In the worst-case scenario, at the sink itself before the packet is processed, the sink's Bloom filter is flushed due to being fully occupied as well.

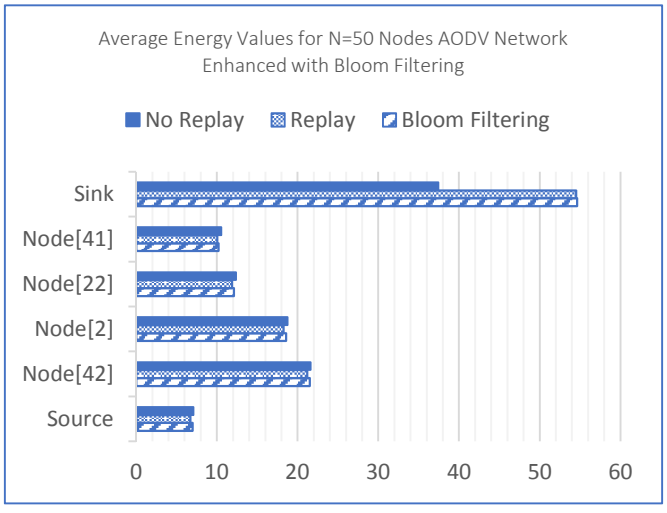

Fig. 7. Energy levels in different nodes in no-replay, in replay, and with Bloom filtering

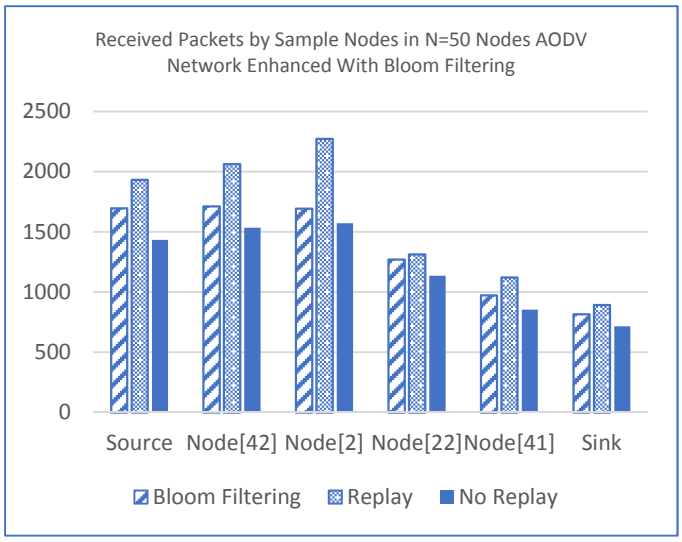

Fig. 8. Number of received packets in different nodes in no-replay, in replay, and with Bloom filtering

IV. Results

Fig.7 shows how integrating Bloom filters into the current WSN can mitigate the impact of replay attacks on average residual energy. Nodes that are no longer required to process the replayed packets, after they are detected and dropped earlier in the transmission path, can now conserve their energy to perform better for longer. In terms of energy preservation, filters can also save energy in nodes preceding the malicious node in the path (the malicious node itself and the source) when it discards extra control packets needed to update routing tables instead of broadcasting those for duplicate packets in an unessential, repeated manner.

The filters also decrease the number of packets needed to be processed by other sensor nodes. The number of packets accepted and processed is significantly moderated for all nodes in the packet transmission path, as shown in Fig. 8. This is most obvious in node [2], the first hop after the malicious node, which discards processing any duplicate message. The replayed packets previously transmitted through the whole path are now dropped before getting halfway through it. Adding Bloom filters to this simulation throws the throughput value back to 132.7 bps. This throughput is less than the throughput of the previously healthy WSN due to the dropped packets mid-path. Instead of receiving three packets, two valid and one duplicate in a specified time interval, the sink receives only two, while one is dropped earlier in the path, meaning fewer bits as a result of receiving two valid packets in the same time interval of an expected three packets.

V. Conclusion

Replay attacks have an inevitable impact on energy storage in WSN nodes. They decrease residual energy and increase exchanged protocol and data messages. Bloom filters can salvage energy storages in different levels according to how big the WSN is and how long the path of transmission is.

Acknowledgement

Thanks and gratitude for Dr. Tassos Dimitriou, the associate professor at the computer engineering department of Kuwait University, and my graduate supervisor, for sparking this thought of research and assisting through the rough turns. This research is a major step in my master thesis which will be the base for further research development in the field of WSN protocol security.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey. *Computer Networks"*, *38*(4): 393–422. http://doi.org/10.1016/S1389-1286(01)00302-4, 2002

[2] T. Aura, "Strategies against Replay Attacks". *Proceedings of Computer Security Foundations Workshop,* 59–68. Rockport, MA: IEEE http://doi.org/10.1109/CSFW.1997.596787, 1997.

[3] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors". *ACM Communications*, 13(7): 422—426, 1970.

[4] A. Broder, and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey". In *Internet Mathematics* 1(4): 485–509. A K Peters, Ltd. http://doi.org/10.1080/15427951.2004.10129096, 2004.

[5] I. D. Chakeres, and E. M. Belding-Royer, "AODV Routing Protocol Implementation Design". In *24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04),* 698–703. IEEE. http://doi.org/10.1109/ICDCSW.2004.1284108, 2004.

[6] F. Deng, and D. Rafiei, "Approximately detecting duplicates for streaming data using stable bloom filters". In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data - SIGMOD '06*, 25-36. http://doi.org/10.1145/1142473.1142477, 2006.

[7] Z. Feng, J. Ning, I. Broustis, K. Pelechrinis, S. V. Krishnamurthy, and M. Faloutsos, "Coping with packet replay attacks in wireless networks". In *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 368–376. http://doi.org/10.1109/SAHCN.2011.5984919, 2011.

[8] T. Heer, S. Götz, O. G. Morchon, and K. Wehrle, "Alpha: An adaptive and lightweight protocol for hop-by-hop authentication". In *Proceedings of the 2008 ACM CoNEXT Conference*, 1-12. http://doi.org/10.1145/1544012.1544035, 2008.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion". In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking - MobiCom '00*, 56–67. http://doi.org/10.1145/345910.345920, 2000.

[10] C. Karlof, and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures". *Ad Hoc Networks*, *1*(2–3): 293–315. http://doi.org/10.1016/S1570-8705(03)00008-8, 2003.

[11] S. Malladi, J. Alves-foss, and R. B. Heckendorn, "On Preventing Replay Attacks on Security Protocols". In *International Conference on Security and Management*, 77–83, 2002.

[12] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA Broadcast Authentication Protocol". *RSA CryptoBytes Technical Newsletter*, *5*(2): 2–13, 2002.

[13] E. M. Royer, and C. E. Perkins, "An implementation study of the AODV routing protocol". In *IEEE Wireless Communications and Networking Conference(WCNC)*, 1003–1008. http://doi.org/10.1109/WCNC.2000.904764, 2000.

[14] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey. *Computer Networks"*, *52*(12): 2292–2330. http://doi.org/10.1016/j.comnet.2008.04.002, 2008.