This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

TIme-Based DynamiC Keying and En-Route Filtering (TICK) for Wireless Sensor Networks

A. Selcuk Uluagac* Raheem A. Beyah† John A. Copeland*

*CSC Lab, School of ECE
Georgia Institute of Technology
Atlanta, GA 30332, USA
{selcuk,jcopeland}@ece.gatech.edu

†CAP Group, Dept. of CS
Georgia State University
Atlanta, GA 30303, USA
rbeyah@cs.gsu.edu

***Abstract*—Given that transmission cost is significant in a Wireless Sensor Network (WSN), sending explicit keying control messages significantly increases the amount of energy consumed by each sensing device. Thus, in this paper, we address the issue of security for WSNs from a completely novel perspective. We present a technique to secure the network, *without* the transmission of explicit keying messages needed to avoid stale keys. Our protocol, the TIme-Based DynamiC Keying and En-Route Filtering (TICK) protocol for WSNs secures events as they occur. As opposed to current chatty schemes that incur regular keying message overhead, nodes use their local time values as a one-time dynamic key to encrypt each message. Further, this mechanism prevents malicious nodes from injecting false packets into the network. TICK is as a worst case twice more energy efficient than existing related work. Both an analytical framework and simulation results are presented to verify the feasibility of TICK as well as the energy consumption of the scheme under normal operation and attack from malicious nodes.**

***Index Terms*—WSN Security, Time-based Dynamic Keying and En-route Filtering, TICK, Wireless Sensor Networks**

I. Introduction

In the last decade, the introduction of wireless sensor networks (WSNs) to the networking field has gathered the attention of academia and industry. Today, WSNs are no longer a nascent technology and future networks, especially Cyber-physical systems (CPS) [1] will require the integration of more sensor-based systems into a variety of application scenarios such as in medical systems, aerospace systems, transportation vehicles and intelligent highways, robotic systems, process control, factory automation, building and environmental control and smart spaces. Providing security to this diverse set of sensor-based applications is necessary for the healthy operations of the overall system because adversaries may target the proper functioning of applications and disturb the critical decision-making processes by injecting false information into the network. Therefore, protocols should be resilient against false data injected into the network by malicious entities.

One way to eliminate injected malicious data from WSNs is to utilize an *en-route-filtering* scheme as in [2], [3], [4]. The en-route-filtering schemes generally utilize keys generated by either *static* [5] and *dynamic* [6] key management schemes [7]. Although dynamic schemes are more attack-resilient than static ones, one significant issue with these schemes is that they increase the communication overhead due to keys being refreshed or redistributed from time to time in the network [8]. Moreover, the common observation with current en-route-filtering schemes [2], [3], [4] are as follows: (1) They are complicated for resource-constrained sensors as they utilize many keys; (2) they transmit many keying messages in the network, which increases the energy consumption of WSNs; and (3) the energy cost, especially the communication cost, associated with the operations of the protocols are often not discussed by researchers when building secure WSN protocols.

Motivated with these points and the fact that the communication cost is the most dominant factor in a sensor's energy consumption [9], [10], we tackle the problem of providing security to sensor-based applications with a new approach. As opposed to other "chatty" dynamic en-route filtering schemes, we focus on eliminating specific control messages for keying or rekeying in the network so that some of the energy savings from transmission cost can be utilized for the computation of local security operations. Thus, in our work, we present the *TIme-Based DynamiC Keying and En-Route Filtering (TICK)* framework that addresses the aforementioned concerns for WSNs. Specifically, TICK uses the local time value of the node, where data is originated, as the dynamic key to encrypt the messages. Then, the receiving nodes on the path to the sink use their local time to successfully decode the timing key of the source node and verify the security of the packet. As time progresses, the subsequent transmissions use different time values to derive the key for the encryption mechanism, which increases the resiliency of the network against adversaries. Thus, the protocol avoids extra overhead of control messages. The nodes forwarding the data along the path to the sink are able to filter out the malicious data verifying its authenticity and integrity with the provision of non-repudiation. With TICK, our main goal is to send events to the sink as energy-efficient and surreptitious as possible to reduce the likelihood of interception by an adversary. More importantly, we seek to minimize the overhead associated with refreshing keys to avoid them becoming stale.

To the best of our knowledge, our approach has not been taken by earlier security studies in the WSN domain. Our novel approach using local clocks is well suited for both WSNs and sensor-based CPS applications where utmost silence is

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

necessary, like in military scenarios, as TICK is not "chatty" in nature. For instance, radio silence is very important for military operations as any radio transmission may reveal troop positions; so, restrictive EMCON[1] orders may be in effect [12]. Both analytical and simulation results verify the feasibility of the TICK framework. TICK is at least two times more energy efficient than other related schemes.

The paper proceeds as follows. Related work is presented in Section II. An overview of the TICK scheme is given in Section III. A performance evaluation with simulations, an analytical analysis, and a comparison with other schemes are presented in Sections III and IV. Section VI concludes the paper.

II. Related Work

En-route dynamic filtering of malicious packets has been the focus of several studies, including Dynamic En-route Filtering (DEF) by Yu and Guan [2], Statistical En-route Filtering (SEF), [3], and Secure Ticket-Based En-route Filtering (STEF) [4]. The brief details of these works as well as their performance are discussed in Section V. However, the common downside of all these schemes is that they are complicated for resource-constrained sensors and they either utilize many keys or they transmit many keying messages in the network, which increases the energy consumption of WSNs. Another significant observation with all of these works is that a realistic energy analysis of the protocols was not presented.

Furthermore, two pertinent studies based on associating keys with time information available in sensor nodes are presented in [13], [14]. In [14], a broadcast authentication scheme, μTESLA, is introduced utilizing the notions of loose-time synchronization and delayed key disclosure. However, sending keys as a separate message is not cost effective and keys may be lost due to communication errors. In fact, another worthwhile study [15] shows how μTESLA would be vulnerable to attacks due to its delayed key disclosure and loose-time synchronization concepts. On the other hand, in Time information-based Pre-deployed Secure Key Distribution (TPSKD) [13], time is used to create session keys between the communicating nodes. Several disadvantages exist in this study. First, the nodes still exchanges Δ_i (drift) values when establishing a pairwise session key with each other; thus, the communication cost of the nodes is increased. Second, the scheme loads the sensors with a randomly chosen fixed Δ_i value initially and assumes the sensors will always drift with this static value. However, in reality, nodes may have different drift values due to the effects of different environmental conditions.

In short, TICK is different from earlier studies in several ways: (1) TICK is a dynamic en-route filtering scheme that does not exchange explicit control messages for rekeying; (2) instead of using the same key multiple times, it provides one-time keys for each packet transmitted and hence avoids stale keys; (3) TICK has a modular and flexible security architecture with a simple technique for providing authenticity, integrity and non-repudiation of data.

[1]EMCON: "The selective and controlled use of electromagnetic, acoustic, or other emitters to optimize command and control capabilities while minimizing, for operations security: a. detection by enemy sensors; b. mutual interference among friendly systems; and/or c. enemy interference with the ability to execute a military deception plan" [11].

III. Overview of TICK

There are three main components of the TICK protocol: Time-Based Key Managemet (TKM), Crypto (CRYPT), and Filtering-Forwarding (FFWD) Modules. The TKM module is responsible for creation of the keys that will be used by the crypto module. The CRYPT module addresses the security part of the problem. Finally, the FFWD module filters the incoming decoded packet out of the network if it is classified as a bad packet or otherwise forwards it to the upstream nodes. The relevant modules are explained in the order they function in the TICK protocol.

A. Threat Model and Assumptions

Source nodes are synchronized and loaded with a network-wide initialization vector (IV) pre-deployment. The IV and local time information will be used to generate the initial and subsequent dynamic keys. Note that the sensor nodes *do not* have perfect clocks and over time the sensors' clocks gradually diverge from the real clock value due to changes in the environmental conditions such as temperature, humidity, pressure, and vibration. In the worst case, they can accumulate up to several seconds of error per day [16]. Thus, the dynamic keys generated in TICK will change as a function of time *and* random drift. As such, the same event reported by different sources located nearby or separate events reported by different sources located elsewhere in the deployment region use different keys. Hence, this situation will increase the effort of brute-forcing by the adversary. Nonetheless, using real clocks requires designing both a flexible and an error-cognizant scheme that would compensate for drifting clocks. This issue is investigated more in Section IV.

Similar to [3], we mainly consider the false injection and eavesdropping of messages from an outside malicious node; hence the insider attacks are outside the scope of this paper. Moreover, attacks on clocks (e.g., pulse-delay (replay) and wormhole) are detected by the extra delay they will introduce into the network as in [17], [18], [19].

The sink is the ultimate terminating point and decision maker. Nodes are statically deployed with same communication ranges. Note that more than one sink may exist in the network and more resources are available to the sink. Finally, the report (packet) size exchanged between the nodes is assumed to be fixed.

B. Time-Based Key Management (TKM) Module

One of the primary contributions of TICK is the generation of keys dynamically using local time. This is addressed in the Time-Based Key Management (TKM) module. When a source node has data to send to the sink due to either an external stimulation by the sink [20] or a self-initiated periodic report, it uses its local clock value as the key. Specifically, the keys are a function of the current time value (t_l) and an initialization

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

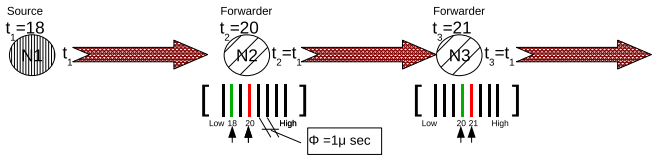


Fig. 1. An illustration of packet delivery path.

vector (IV) (i.e., $K_j^t \leftarrow F(t_l, IV)$). Note that nodes do not need to go through the shared-key discovery phase. This, in return, brings an extra energy savings in the communication cost.

For example, assume in Figure 1 the source node is $N1$, and the forwarder nodes $N2$, and $N3$ are on the path to the sink that the report by $N2$ will traverse. Note that $N1$ inserts a copy of its ID and a local counter value inside the report (packet) sent to the sink. The counter serves as a protection against replay attacks. It is increased each time a packet is sent from the source. The ID is used to verify the integrity of the packet. As in Figure 1, $N1$ uses its local clock value *18* as the key. This key is used by the CRYPT module to perform the desired cryptic operations depending on the security service (e.g., encryption, authentication, integrity) provided by the WSN application. When $N2$ receives the report from $N1$, it tries to find the value of the time at $N1$. First $N2$ substracts the approximate packet flight time ($\Theta = \rho + \tau + \varphi + \varepsilon$) between itself and $N1$ from its local time in order to be closer to the local time at $N1$, where ρ is the propagation time, τ is the packet transmission time, φ is the packet processing time, and ε is the approximation of errors for variability in transmissions due to fading, obstructions, and software errors, etc[2]. Furthermore, in order for a forwarder node to find the local clock value at the source node easily, all nodes are associated with a window of values, which we refer to as the *tick window, (T_w)* and a tick value (ϕ). Thus, $N2$ will try all values inside its tick window beginning from its local clock value. Once $N2$ finds the correct key value associated with the time at $N1$, it will be able to perform other security actions on the packet in the crypto module and will also be able to compute the time offset from the sender. However, to combat against counterfeit values and to ensure a forwarder node does not futily attempt to brute-force all time-based keys, lower and upper bounds are associated with each node's tick window. Note that proper choice for the size of the tick window depends on, among other parameters, the tick value and it is explained more in the next section.

C. Crypto (CRYPT) Module

The CRYPT module obtains the dynamic key from the TKM module and performs the necessary security service. This is also the module where the key from the TKM is verified. If the key value received from the TKM module is not correct then a new key is obtained from the the TKM module. This process continues until the correct key is found or the packet is marked as malicious to be discarded in the filtering-forwarding (FFWD) module when all attempts to find the correct key are exhausted within the tick window, (T_w). The CRYPT module incorporates the RC4 algorithm into its body as the encryption mechanism. The rationale for choosing RC4 is due to its proven lightweight computational energy consumption on sensors [21], [22]. The risk of differential cryptanalysis is eliminated since a new time-based key is used as input for each RC4 block [23]. Moreover, since the key is generated in another module, any desired encryption (e.g., DES, 3DES), authentication, or integrity mechanism (e.g., HMAC, CMAC) can be implemented together or separately depending on the security service desired from the WSN application. After the correct value of the key used by the sender is determined by the current node, the offset value for the sender node is stored by the current node.

[2]A realistic analysis of the uncertainty associated with errors is presented in the next section.

D. Filtering-Forwarding (FFWD) Module

The filtering-forwarding (FFWD) module in TICK is the module that filters the packets from the network if the incoming packet is malicious or forwards the data toward the sink otherwise. Specifically, it receives the decision about the decrypted packet from the CRYPT module. If the packet is not malicious, then the original incoming packet is forwarded to the next hop sensor intact toward the sink. Note that the original packet is not enlarged in any way (e.g., with MACs) to keep the energy costs at a minimum.

IV. Computation of the Tick Window (T_w)

Uncontrolled environmental conditions such as changes of temperature, humidity, pressure, and sudden vibrations in the deployment area cause internal clocks to gradually diverge from the real clock. Moreover, channel access time (at the medium access control layer) and send-time (including the time for preparing the packet at the application layer and passing it to the lower layers), can be considered as contributing to the unpredictable [18] clocks. In TICK, the environmental factors are captured with the parameter δ, which is the daily value of the drift per sensor given a deployment area, while the software-based factors are captured with ε. We adopt the values reported in [16], [17], [18], [19] for ε and δ. Deterministic factors, on the other hand, depend on more predictable parameters. In TICK, as in [17], [18], these include the transmission time of one packet (τ), the propagation delay (ρ), the packet processing time (φ) (e.g., due to cryptographic operations), and the average period of data from sensors (λ). The effect of all the factors are captured by the tick window, T_w, and it is the most significant parameter in dealing with the uncertainty in TICK. It provides a window of time values. However, even though the TICK protocol is designed with a flexible T_w mechanism, a quantitative analysis is still needed. Therefore, in this section, first an analytical model is presented to investigate the relationship between the size of T_w and the tick value (ϕ). Then, a realistic tick window (T_w) value is derived considering the capabilities of today's wireless sensor devices.

As briefly mentioned previously, the tick window T_w is available for the receiver node to choose from to decode the

received packets. The window has upper and lower boundary values. The efficacy of the TICK protocol depends on the size of this window because the larger the size of T_w, the more time it takes for a sensor to find the key. In TICK, the T_w value is basically a function of the tick value (ϕ). The smaller the value of ϕ, the more keys could be tried by each sensor, hence T_w is larger and the accuracy of the scheme is increased. Also from the sender's perspective, as the system becomes more precise (i.e., the smaller the ϕ), the chance of using a different key per packet transmitted increases. As long as the frequency of the events (packets) is larger than $\frac{1}{\phi}$, the system will use a different key per packet. Hence, assuming that the sensor application sends its data periodically (or on the average) at certain time intervals to the sink [24], [20], T_w can be computed as

$$T_w = \frac{(\lambda + \rho + \tau + \varphi + \varepsilon) * \delta}{3600 * 24 * \phi} \quad (1)$$

where τ is the transmission time of one packet, $\tau = \frac{l}{R}$ with l and R being packet length and rate of the WSN link, respectively; ρ is the propagation delay, $\rho = \frac{\chi}{c}$ with χ and c being distance between the sensors and the speed of light in the medium, respectively; λ is the average period of data in between sensed reports sent from a sensor; φ is the packet processing time, ε is the physical transmission error; δ is the daily value of the drift per sensor given a deployment area; and ϕ is the desired tick value. Note that (1) governs all the uncertainty factors into its body. Also, χ is taken based on the possible maximum distance to consider the worst case scenarios although nodes may be located closer than the maximum distance. Several observations are possible with a close examination of (1). When sensors send less frequently to the sink, hence λ is larger, the value of the T_w becomes larger. This obviously increases the computational effort of the sensor to find the correct key. A similar remark can be made for ε as well. On the other hand, when λ is smaller (i.e., more frequent data), the T_w is smaller. Hence, the scheme does not spend too much time trying to find the correct key; and the computational effort is smaller.

However, a realistic key window (T_w) value can be derived considering the capabilities of today's wireless sensor devices. Assuming a sensor node with a microcontroller unit (MCU) of MSP430F16x [25] and a transceiver of CC2420 [26], [10], [27], and also assuming RC4 [21] as the encryption scheme, with $l = 32$ bytes, $R = 250Kbps$, $\lambda = 5$s, $\delta = 2$s, T_w can be found to be 16; hence, the tick value, ϕ, of 7.24μs. Thus, given the technical capabilities of sensors today, the value of T_w computed in this section is instrumental in making TICK a realistic protocol as much as possible and will be used in the performance evaluation section.

V. Performance Evaluation

In this section we evaluate the effectiveness of the TICK protocol both via simulations and analysis. First, a comparative study considering other similar works is given. Next, simulations results are presented to examine the energy efficiency of our scheme under normal operation and under attack. Note that an analysis for the filtering efficiency is not needed as in [28], [29] because in TICK, malicious packets are immediately taken out from the network at the next hop.

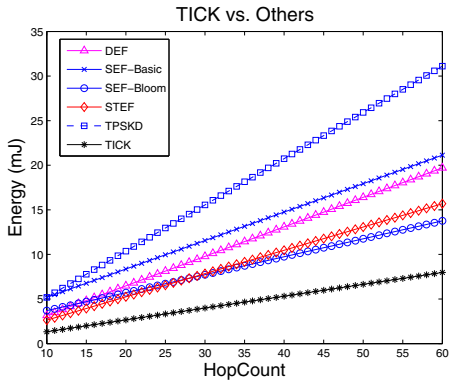
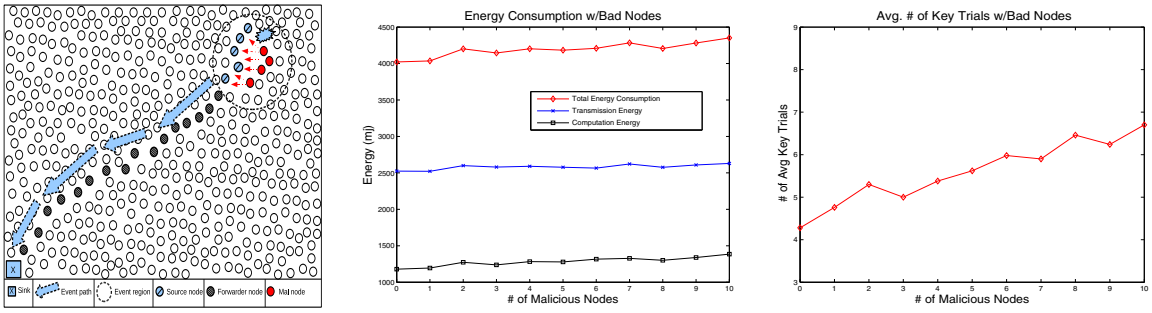


Fig. 2. Comparison of TICK, DEF, SEF, STEF, and TPSKD.

A. Comparison with Other En-route Filtering Schemes

In this sub-section, the energy performance of TICK is analytically compared with other relevant en-route filtering studies in the literature. Specifically, we compare the expected energy costs of dynamic en-route filtering (DEF) [2], statistical en-route filtering (SEF) [3], Secure Ticket-Based En-route Filtering (STEF) [4], and Time-based Predeployed Secure Key Distribution (TPSKD) [13] with that of TICK after very briefly summarizing each scheme. In DEF [2], a legitimate report is endorsed by multiple sensing nodes using their own authentication keys. It utilizes authentication keys and separate secret keys to disseminate those authentication keys; hence, it uses many keys. With SEF [3], each sensed report is validated by multiple keyed message authentication codes (MACs). Thus, the downside of SEF is that packets are enlarged by MACs. Although the authors suggest the use of bloom-filters to decrease the MAC overhead, SEF is a static key-based scheme and it inherits all the downsides of static key management schemes [8]. The STEF protocol [4] proposes using a ticket concept, where tickets are issued by the sink and packets are only forwarded if they contain a valid ticket. The downside of STEF is its one way communication in the downstream for the ticket traversal to the cluster heads. Finally, TPSKD [13] is essentially a time-based secure key pre-distribution scheme and it is not a en-route protocol per se. Its main purpose is to create static pairwise keys between the nodes. It is included in our comparative analysis here as it also uses time information to create keys. The scheme initially loads the sensors with fake clock drift values (Δ). These values are then exchanged by the nodes to create pairwise link keys in the clear.

A comparison of each scheme in terms of their energy consumption is presented in Figure 2. The results are generated for one round of communication from a source node to the sink, which is assumed to be located *n* hops away from the source node. The x-axis represents the hop count and is varied, while the y-axis is the energy. To simplify the comparisons, we assumed that all the nodes would have the keying material with probability of 1 to do the desired



(a) Simulation topology. (b) Total energy consumption. (c) Avg. number of key-trials.

Fig. 3. Simulation topology and computation, transmission, and total energy consumption under an attack scenario.

TABLE I
SIMULATION PARAMETERS

# of Nodes	500	SensSize	32 bytes	E_{ini}	5000 mJ	E_{dec}	3.3μJ
Area	1000x1000 m	RecvInterval	5s	E_{rx}	66.7μJ	E_{enc}	3.3μJ
Link Rate	250Kbps	SimTime	3000s	E_{tx}	59.6μJ	E_{mac}	8.6μJ
Range	75 m	#of Mal Node	(0..10)	E_{sens}	9μJ	E_{sa}	11.4μJ
# of Healthy Nodes	10	T_w	16	Time Offset	U$[-3,+3]\mu$s	Voltage	3V

security features imposed by the specific protocol in a benign environment (no malicious nodes). Without loss of generality, we assumed that all the schemes would use the same type of cryptographic mechanisms unless specified otherwise by the referenced work. Hence, we assumed that the protocols that use hashing and encryption mechanisms would use MD5 and RC4, respectively. The real sensor implementation values for these crypto mechanisms are taken from [21] and [22]. As can be seen, TICK is very energy-efficient compared to other schemes. The other schemes exchange keying messages and use many static keys. TICK eliminates these from its design and is able to save energy and reduce the opportunity for attackers to intercept packets.

B. Security and Energy Consumption Analysis

In this sub-section we evaluate the performance of the TICK protocol via simulations. We focus on the energy consumption of the TICK protocol while under attack.

1) Simulation Parameters and Assumptions: We use the Georgia Tech Sensor Network Simulator (GTSNetS) [30], which is an event-based sensor network simulator with C++, to perform the analysis of the TICK protocol. The topology and the parameters used are given in Figure 3(a) and in Table I. Nodes were located randomly in the deployment region and on average, source nodes were 25−35 hops away from the sink. The energy costs for different operations in the table are computed based on the values given in [25], [27]. However, the costs for encryption and decryption operations are computed based on the the reported values of the implementation of RC4 [21] on real sensor devices. E_{tx}, E_{rx}, and E_{sens} are the energy consumption of sending, receiving a packet and sensing an event, while E_{enc}, E_{dec}, and E_{mac} are the costs of encryption, decryption, and the message authentication code, respectively. We use 16 as the value of T_w as found in the previous sub-section. Due to the broadcast nature of the wireless medium used in WSNs, attackers may try to eavesdrop, intercept, or inject false messages. In this paper we mainly consider the false injection and eavesdropping of messages from an outside malicious node; hence similar to [3], the insider attacks are outside the scope of this paper. In our attack scenario, the total number of healthy source nodes that collect the event information and send it toward the sink is assumed to be fixed, whereas the number of malicious nodes are increased over time. As in Figure 3(a), the malicious sensors are randomly located inside the event collection region. Throughout this work, the following additional assumptions are made: each node has its local clock and its drift value from the real clock is generated using a uniform distribution between -3 and +3 μs similar to [19]. The Directed Diffusion routing protocol [20] is used, but others such as [31] can also be used. According to specifics of Directed Diffusion, after the sink asks for data via *interest* messages, a routing path is established from the sources in the event region to the sink. Thus, we assume that the path is fixed during the delivery of a particular sensed event report. The sensor network is densely populated such that multiple sensors observe and generate reports for the same event. Sensors are assumed to have the same communication ranges and may have different initial battery supplies. Finally, the simulation results presented in the figures are the average of 50 simulation runs for a specific analyzed parameter.

2) Simulation Results for Security and Energy Consumption: Figure 3(b) shows the results considering the aforementioned attack scenario. The x-axis represents the number of malicious nodes inside the region and y-axis respresents the energy consumption in mJ. We see that as the number of malicious nodes increases inside the network, nodes spend more computation energy. This happens because the number of nodes who use all their key-trial attempts and ultimately classifies a packet as malicious, increases with the increased malicious traffic.

3) Simulation Results for Key-trials: As explained in Section II, when a sensor receives a packet from another sensor, it tries to find the time-based key value associated with this packet used by the sender when encrypting the packet before sending. However, the total trial attempts is limited by the value of key window T_w, not to exhaust the resources onboard the sensor and the nodes immediately eliminate the malicious data from the network once they exhaust all their key-trial attempts. For this, we have used in our simulations a feasible value for T_w (16) given for today's sensor technology as we discussed in the previous section. With this in mind, it is also interesting to look at how many key-trial attempts on average that sensors uses in the simulations when attempting to decrypt the received packets. We generally observe in Figure 3(c) that the increase of malicious activity in the network increases the efforts of the sensors. Since packets are dropped immediately when nodes exhaust all of their key-trial attempts, the system does not allow a malicious packet to get through the network. Also, one interesting result is that nodes do not use all the attempts; the highest point for our attack scenario was around 6.9.

VI. Conclusion and Future Work

The communication cost is the most dominant factor in a sensor's energy consumption. Thus, in TICK, instead of explicitly sending keying or rekeying messages as opposed to current "chatty" schemes, sensor nodes use their local time values as a one-time dynamic key to encrypt each message. The receiving nodes use their local time to intelligently decode the timing key of the source node. As time progresses, the following transmissions use different time values. TICK is also an effective dynamic en-route filtering mechanism, where the malicious is filtered out from the network. To the best of our knowledge earlier dynamic en-route filtering schemes for WSNs have not taken this approach. Both analytical and simulation results verified the feasibility of the TICK scheme and presented that TICK was more energy-efficient than other comparable schemes. Our future work include improving TICK to provide loose-time synchronization for various WSN applications and stateful operations to recall previously seen nodes.

Acknowledgment

This work was partly supported by NSF Grant No. CAREER-CNS-0545667 844144.

References

[1] M. Iqbal and H. B. Lim, "A cyber-physical middleware framework for continuous monitoring of water distribution systems," in *Proc. of the 7th ACM SenSys*, 2009, pp. 401–402.

[2] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," *Proc. of IEEE INFOCOM*, pp. 1–12, April 2006.

[3] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE JSAC*, vol. 23, no. 4, pp. 839–850, April 2005.

[4] C. Kraub, M. Schneider, K. Bayarou, and C. Eckert, "Stef: A secure ticket-based en-route filtering scheme for wireless sensor networks," *The 2nd Int. Conf. on Availability, Reliability and Security (ARES)*, pp. 310–317, April 2007.

[5] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proc. of IEEE Symposium on Security and Privacy*, 2002, pp. 41–47.

[6] M. Eltoweissy, M. Moharrum, and R. Mukkamala, "Dynamic key management in sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 122–130, April 2006.

[7] S. Uluagac, C. Lee, R. Beyah, and J. Copeland, "Designing secure protocols for wireless sensor networks," *Lecture Notes in Computer Science, Wireless Algorithms, Systems, and Applications (WASA).*, vol. 5258, pp. 503–514, 2008.

[8] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 11-12, pp. 2314–2341, 2007.

[9] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[10] R. Roman, C. Alcaraz, and J. Lopez, "A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes," *Mobile Networks and Applications, Springer*, vol. 12, no. 4, pp. 231–244, August 2007.

[11] D. of Defense, *Department of Defense Dictionary of Military and Associated Terms, Joint Publication 1-02*, 2001.

[12] B. Nguyen and R. Rom, "Communication services under emcon," in *Proc. of the ACM SIGCOMM Conference*, 1986, pp. 275–281.

[13] J. Jeong and Z. Haas, "Predeployed secure key distribution mechanisms in sensor networks: current state-of-the-art and a new approach using time information," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 42–51, Aug. 2008.

[14] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Kluwer Wireles Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[15] D. Scott, "Relying on time synchronization for security in ad hoc networks," in *Proc. of 43rd ACM Southeast Conference*, March 2005.

[16] A. Boukerche and D. Turgut, "Secure time synchronization protocols for wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 5, pp. 64–69, October 2007.

[17] 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*. New York, NY, USA: ACM, 2006, pp. 264–277.

[18] S. Ganeriwal, C. Pöpper, S. Čapkun, and M. B. Srivastava, "Secure time synchronization in sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 4, pp. 1–35, 2008.

[19] K. Sun, P. Ning, and C. Wang, "Secure and resilient clock synchronization in wireless sensor networks," *IEEE JSAC*, vol. 24, no. 2, pp. 395–408, Feb. 2006.

[20] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. of ACM MOBICOM*, August 2002, pp. 56–67.

[21] R. V. et al., "Encryption overhead in embedded systems and sensor network nodes: modeling and analysis," in *Proc. of ACM CASES '03*, 2003, pp. 188–197.

[22] M. Passing and F. Dressler, "Experimental performance evaluation of cryptographic algorithms on sensor nodes," Oct. 2006, pp. 882–887.

[23] B. A. Forouzan, *Cryptography & Network Security (1st edition)*. McGraw-Hill, 2007.

[24] O. Akan and I. Akyildiz, "Event-to-Sink Reliable Transport in Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1003–1017, Oct. 2005.

[25] *MSP430x1xx Family User's Guide Rev. F*, Texas Instruments, November 2008. [Online]. Available: http://www.ti.com/msp430

[26] *CC2420DataSheet, 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver Rev. B*, Chipcon Products from Texas Instruments, November 2008. [Online]. Available: focus.ti.com/lit/ds/symlink/cc2420.pdf

[27] Xbow, "Crossbow technology," 2008. [Online]. Available: http://www.xbow.com/

[28] A. S. Uluagac, R. Beyah, and J. Copeland, "Virtual energy-based encryption and keying (vebek) for wireless sensor networks," *Accepted to Appear in IEEE Transactions on Mobile Computing,*, 2010.

[29] H. Hou, C. Corbett, Y. Li, and R. Beyah, "Dynamic energy-based encoding and filtering in sensor networks," in *Proc. of the IEEE MILCOM*, October 2007.

[30] G. T. S. N. Simulator, "Gtsnets," 2007.

[31] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Elsevier Ad Hoc Networks Journal*, vol. 3, pp. 325–349, May 2005.