A Novel Approach for Security and Robustness in Wireless Embedded Systems

Mohammad Iftekhar Husain, Shambhu Upadhyaya, Madhusudhanan Chandrasekaran

Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY USA 14260
{imhusain, shambhu, mc79}@cse.buffalo.edu

Abstract: Security and robustness are paramount in wireless embedded systems due to the vulnerability of the underlying communication medium. To institute security and reliability, most of the existing schemes perform periodic re-establishment of authentication credentials and share secrets among various participating nodes. However, such measures result in overheads in an energy-constrained wireless environment. To alleviate this problem, we propose a software approach that exploits the features of the underlying communication protocol and uses the concept of steganography and covert channels. The highlight of our approach is that it does not require any changes to the protocol and relies only on the modification of frame contents without degrading the protocol performance. We argue that our covert-channel based communication scheme provides security and robustness at low cost and it neither requires centralized authority nor does it disrupt the overall network operation. We evaluate the security benefits of our proposed method in terms of the difficulty of detecting the covert channel by the adversary and compare our technique with other existing schemes. Performance evaluation is done by determining the bandwidth efficiency of the channel, backward compatibility with the standard MAC as well as the ease of implementation.

Keywords: Covert channel, Embedded systems, Media Access Control (MAC), RTS/CTS, Security, Wireless networks

1. Introduction

Covert channels [1] are communication channels that are neither designed nor intended to transfer information. Covert channels usually exploit the legitimate use of shared resources and operations of a system to leak sensitive information to someone who is not authorized to access it. In the literature, two types of covert channels exist: storage and timing channels. A storage channel involves direct or indirect writing of a storage location by one process (sender) and direct or indirect reading of the storage location by another process (receiver) [2]. A timing channel involves a sender process that signals information to another by modulating its own use of system resources (e.g., CPU time) in such a way that this manipulation affects the real response time observed by the second process [2]. Also, there are hybrid channels where time and storage information are used together. Unlike traditional communication channels, a covert channel does not need to have a high capacity or transmission rate to be useful. In contrast, the difficulty of detection and resilience are much more important issues for covert channels. From an adversary's point of view, it should be hard for a monitor to discover the existence of the covert channel. Network covert channels leak information across the network through unused fields of network packets (storage channel) as well as the timing of sending and receiving packets (timing channel).

Steganography, on the other hand, refers to concealing the existence of a message when secret information is hidden in an innocuous cover object. From a security analyst's point of view, covert channels can therefore also make use of network packets as the cover object and thus can be used for secure communication. There are several research works in the literature [4], [5] that have studied data hiding techniques in the TCP/IP protocol suite.

As wireless embedded systems like sensors are gaining ground these days in mission critical applications, so does the need for effective security mechanisms for their operation. Because wireless networks comprising of embedded nodes might not have any infrastructure and/or operate in hostile and unattended environments, it is imperative that these security concerns be addressed for robust and dependable operation. However, due to inherent resource and computing constraints, security in wireless networks poses different challenges than traditional wired network security. A desirable feature in wireless embedded systems is that the security solutions be lightweight.

Cryptographic techniques in wireless networks such as WEP [17], [18] and its successors use lightweight methods based on sharing cryptographic credentials among the participating nodes or stations. If this information is compromised or revealed to an adversary, the older secrets must be revoked, followed by the reestablishment of the credentials. Even when the credential is not compromised, sometimes the nodes might need to update or verify the credentials periodically to maintain the security association. In a wired network, it is comparatively easier to update, verify or rebuild the credentials in a secure way. But in the wireless domain with bandwidth and energy constraints and due to the infrastructure-less open mode, re-establishing the security association is a difficult task. So, an important security problem in wireless domain is how to effectively and efficiently enable a node to communicate with its peer securely to update, verify or re-distribute security credentials without using the standard security mechanisms or cryptographic primitives. More precisely, the focus is to ensure the confidentiality and integrity of sensitive data that the node sends to its peer with minimum or no overhead.

To address this security issue, we propose a covert channel based communication mechanism using the concept of steganography. In other words, we use the technique of data hiding using normal network packets in the wireless domain. The reason for hiding data will become apparent shortly. The idea of the protocol is to share a pre-deployed secret and verify the secret using covert channels in an innocuous way. We refer to our protocol as Opportunistic Secure Communication (OpSeCom). This will allow us to detect and isolate compromised nodes in a wireless network. The protocol uses the technique of Bit Commitment [22] as a means to implement challenge-response (handshake). The characteristic of Bit Commitment is that it shares a pointer to the data first and reveals the actual data later, making it a good choice in our protocol for secret sharing. As IEEE 802.11 MAC is widely used in the wireless domain, we choose the two control packets, viz. RTS (request to send) and CTS (clear to send) to covertly (to defeat the adversary) communicate and verify the shared secret. As RTS and CTS are optional control packets in IEEE 802.11 MAC, appearance of these packets in the network is a normal phenomenon. Though, we are using bit commitment to share secret and RTS/CTS for covert communication to demonstrate the protocol in this paper, these are open for the user to choose. In other words, our proposed idea is generic and does not dependent on the choice of the algorithm for secret share or network packet for covert communication.

So, we design a storage based network covert channel for opportunistic secure communication in wireless domain for updating, verifying or re-establishing security credentials without using standard security mechanisms. Our proposal is neither dependent on a centralized authority nor it re-initializes the whole network to achieve the goal which makes it unique from other security mechanisms. We will demonstrate the security of our technique through the analysis of the covert channel and performance in terms of bandwidth efficiency, co-existence with standard MAC and the ease of deployment.

The rest of the paper is organized as follows: In the next section we will formulate the problem. Section 3 discusses our proposed solution: Opportunistic Secure Communication (OpSeCom). Sections 4, 5 and 6 deal with the security analysis, the robustness of the covert channel, and the performance analysis and comparison with related work, respectively. Section 7 concludes the paper.

2. Problem Definition and Related Work

2.1 Assumption

In this paper we assume wireless embedded systems network as a collection of decentralized embedded devices that use IEEE 802.11 MAC for shared medium access and RTS/CTS for virtual career sensing [17]. The stations (embedded devices) are subject to energy and computational constraints. They are pre-configured for wireless communication prior to deployment in the infrastructure-less mode.

We now discuss several application scenarios and available solutions to highlight the problem domain and to put our research in perspective.

2.2 Scenario 1: Verification of Existence of Malicious Nodes

Detecting malicious nodes like phantom nodes [8] and false beacon nodes [9] is important in wireless sensor networks because it hinders the normal operation of the network. There are several approaches [16] in the literature addressing detection of malicious nodes. These techniques can raise alerts against a suspicious node but they cannot exactly pinpoint the malicious nodes in a decentralized environment. They also produce false positives. There is still lack of effective techniques to accurately verify malicious nodes after an alert is raised so that such nodes can be excluded from the normal network operation once and for all.

2.3 Scenario 2: Re-establishing Security Credentials

Usually, in a wireless domain, a network-wide shared key is used for cryptographic operations in the network. Although, this makes the deployment and key management easier, a big disadvantage of this technique is, if the shared key is revealed, the whole network security is compromised. To restore network security, distribution of some sort of secret information is needed among the nodes. Several key management proposals [18], [19], [20] address this issue in an ad hoc manner by proposing to re-initialize the whole network operation with new keys or security credentials. But, this is not a viable idea in wireless domain because of the existence of energy and computational constraints as well as lack of infrastructure in most cases.

2.4 Scenario 3: Updating Trust and Security Credentials

Again, when a network-wide shared key is used, it is important to update the keys periodically for enhanced security. Though, the periodicity of update depends on the type of network, in wireless domain, a feasible solution will be something which is efficient in computation and communication as well as one that does not allow the malicious nodes to know that the credentials are being updated. Proposals addressing these issues [14], [15] use central authority based complex cryptographic operations which are computation-intensive. In addition, they involve exchange of multiple messages among the nodes to successfully update the credentials which incurs significant communication overhead. So, these ideas are not feasible for resource constrained wireless embedded systems communication.

2.5 Summary of Problem Domain

All the above mentioned scenarios in wireless embedded networks demand for careful communication and data transfer. Such communication should be lightweight, easy to implement and efficient. To address these issues we propose to exploit opportunities that may exist in the form of storage based network covert channel (example: RTS/CTS) as well as the idea of secret sharing (example: Bit Commitment). The next section presents the details of our protocol as well as the basics of RTS/CTS and the bit commitment protocol for a better understanding of the solution.

3. Proposed Method (OpSeCom)

3.1 Basics of RTS/CTS [17]

RTS/CTS (Request to send / Clear to send) is primarily a channel reservation mechanism used by the IEEE 802.11 MAC protocol to reduce frame collisions introduced by the hidden terminal problem and exposed node problem [17].

RTS Packet Format:

Frame Control (2)	Duration (2)	Transmitter Address (TA) (6)	Receiver Address (RA) (6)	FCS(4)

Fig. 1. RTS Packet (20 Bytes) Format

The RA field of the RTS frame is the address of the intended immediate recipient of the data. The TA field is the address of the node transmitting the RTS frame. The duration value is the time, in microseconds, required to transmit the pending data or management frame, plus one CTS frame, plus one ACK frame, plus three SIFS (Short

Inter Frame Space) intervals. If the calculated duration includes a fractional microsecond, that value is rounded up to the next higher integer.

CTS Packet Format:

Frame Control (2)	Duration(2)	Receiver Address (RA) (6)	FCS(4)

Fig. 2. CTS Packet (14 Bytes) Format

The RA field of the CTS frame is copied from the TA field of the immediately previous RTS frame to which the CTS is a response. The duration value is the value obtained from the Duration field of the immediately previous RTS frame, minus the time, in microseconds, required to transmit the CTS frame and its SIFS interval. If the calculated duration includes a fractional microsecond, that value is rounded up to the next higher integer.

3.2 Bit Commitment Protocol

Bit commitment is a simple and straightforward cryptographic primitive to enable secret sharing between two mistrusting agents, say, Alice and Bob. This is how the protocol works. Alice puts her secret in a locked box and gives it to Bob. So, neither Alice nor Bob can change the secret as Alice does not have the box and Bob does not have the key. To reveal the secret, Alice simply sends the key later. This ensures secret sharing with cheat prevention. This protocol can be used in our scenarios to implement a challenge and response between peer nodes. Details of Bit Commitment protocol can be found in [22].

3.3 OpSeCom in Action

In order to illustrate our security protocol, we consider two wireless embedded devices Alice and Bob. Alice is the node under suspicion of a compromise and Bob is the verifying node. We assume that, there are some alarm mechanisms at Alice which will be triggered when Alice is under attack. For example, if Alice is a wireless sensor node, she can use photo sensors or motion sensors to detect an attack and trigger an alarm. Also, we assume that Alice can use some software mechanisms such as anomaly detectors to trigger an alarm if sensitive files are accessed or altered at the application layer. When an alarm is triggered, the system will automatically set the corresponding bit string in the CTS packet at the MAC layer. These changes at the MAC layer occur momentarily and the adversary at the application layer will be oblivious of such changes as it takes place as a triggered system task.

The problem here is to verify that Alice is indeed compromised. The protocol works in three phases as illustrated in Figure 3. Phase 1 and Phase 3 are mandatory whereas Phase 2 is optional since verification may be needed even when there is no attack. In the figure, horizontal arrows represent a communication while vertical arrows indicate progression in time.

1. **Secure Parameter Agreement Phase:** When the network is established, Alice

sends two code words *m* and *n* to Bob, committing to the bit strings corresponding to normal situation and under-attack situation respectively. They will also agree upon the challenge sequence that Bob will send to verify the node condition. By default, Alice will have a special CTS packet set at the MAC layer containing the bit strings corresponding to normal situation as the covert data. This is illustrated in Fig. 3(a). There are several bit commitment protocols [22] in the literature and the implementer is free to choose one.

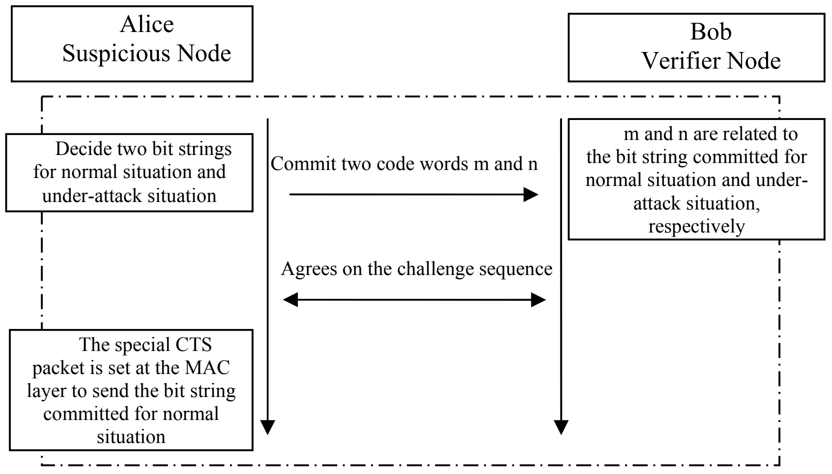
Alice
Suspicious Node
Bob
Verifier Node
Decide two bit strings for normal situation and under-attack situation
Commit two code words m and n
m and n are related to the bit string committed for normal situation and under-attack situation, respectively
Agrees on the challenge sequence
The special CTS packet is set at the MAC layer to send the bit string committed for normal situation

Fig. 3(a). Secure Parameter Agreement Phase

2. **Node under Attack Phase:** When Alice is under attack, she will trigger an alarm which will alter the bit string in the CTS packet at the MAC layer and set it to the bit string committed for under-attack situation as described earlier. At the same time, Bob can also sense some abnormality in Alice's behavior, for example too much deviation in data, activity/inactivity, excessive dropping packets, etc. This is illustrated in Fig. 3(b).

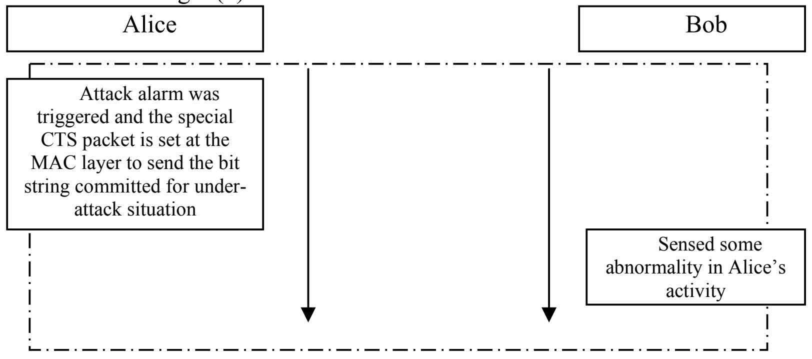
Alice
Bob
Attack alarm was triggered and the special CTS packet is set at the MAC layer to send the bit string committed for under-attack situation
Sensed some abnormality in Alice's activity

Fig. 3(b). Node under Attack Phase

3. **Covert Verification Phase:** When Bob suspects that Alice is under attack or when Bob desires to verify his security association with Alice, he will send an RTS packet containing the challenge sequence as the covert data. Upon receiving the special RTS packet, Alice will respond with special CTS packet already set at the MAC layer. This response is generated at the MAC layer and is beyond the notice of application layer. So, it is expected that, even if Alice is compromised or captured, the special CTS packet will contain correct bit string corresponding to the situation. So, Bob will verify the bit string and decide whether Alice is in normal operation mode or under-attack as shown in Fig. 3(c).

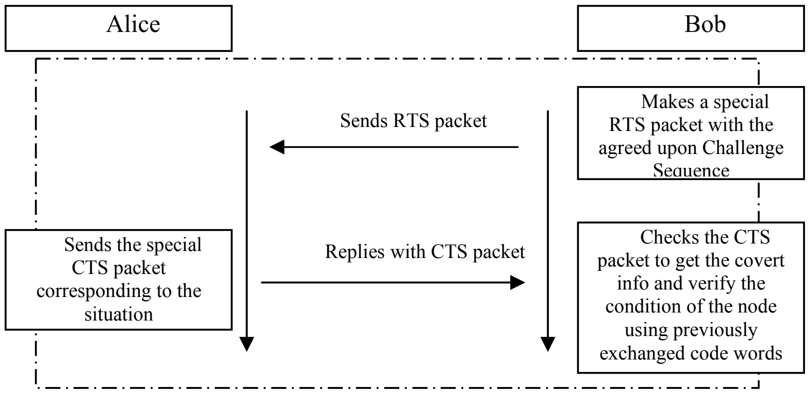
Alice
Bob
Makes a special RTS packet with the agreed upon Challenge Sequence
Sends RTS packet
Replies with CTS packet
Sends the special CTS packet corresponding to the situation
Checks the CTS packet to get the covert info and verify the condition of the node using previously exchanged code words

Fig. 3(c). Covert Verification Phase

The covert operation of RTS and CTS packets is described below. In this particular example we use TA and RA fields to hide data in a manner that is similar to [21]. However, the implementer can choose other network packets and fields to use as network covert channel. For example, there are lots of unused bits in the Frame Control Sequence field and the Acknowledgement frame fields in the IEEE 802.11 MAC protocol, which can be used to carry covert data.

3.4 RTS/CTS Covert Channel

In the RTS packet, there are five fields as described in Sec. 3.1. Out of these, Frame Control, Duration, RA and FCS are necessary for the proper performance of the protocol. TA field is basically the RTS-sending station's MAC address. This information gets copied in the CTS RA field. So, we modify both RTS TA field and CTS RA field in such a manner that there remains a connection between a RTS and the corresponding CTS.

As both TA and RA fields should contain the MAC addresses, the modified information in those fields should look like MAC addresses. This will keep the RTS and CTS packets out of suspicion from non-OpSeCom nodes. This part is crucial because if non-OpSeCom nodes suspect and start brute force analysis of the packet, they can bring down the protocol within polynomial time. According to IEEE standard, universally administered and locally administered MAC addresses are distinguished by setting the second least significant bit of the most significant byte of the address. If the bit is 0, the address is universally administered. If it is 1, the address is locally administered. Again, if the least significant bit of the most significant byte is set to 0, the packet is meant to reach only one recipient (unicast). If the least significant bit of the most significant byte is set to a 1, the packet is meant to be sent only once but still reaches several stations (multicast). In our design, we will put it as a unicast locally administered MAC address. So, the first two least significant bits of the most significant byte will be set to 01 in our design.

Since we have used the most significant byte, we are left with 40 bits. Among these 40 bits, we will use 8 bits for randomly generated RTS ID. Remaining 32 bits will be the challenge sequence. So, the RTS TA field will appear as shown in Fig. 4.

02H(8)	RTS ID (8)	Challenge Sequence (32)

Fig. 4. OpSeCom RTS TA (6 Bytes) storage channel

The station receiving CTS will check the challenge sequence, put the bit strings corresponding to the situation as Response Sequence and create the CTS RA storage channel as shown in Fig. 5.

02H(8)	CTS ID=RTS ID (8)	Response Sequence=Bit String (32)

Fig. 5. OpSeCom CTS RA (6 Bytes) storage channel

As OpSeCom is a two way handshake, we need to modify the duration field of RTS/CTS also. The duration field of RTS will be the time, in microseconds, required to transmit one CTS frame, plus one SIFS interval. CTS duration value will be zero.

4. Security Analysis

Here we briefly discuss how OpSeCom can address the various security concerns described in the scenarios of Section 2.

4.1 Verifying Existence of Malicious Nodes

In order to verify whether a device is malicious or not, a verifier device can just generate and send an OpSeCom RTS packet to it. By malicious, here we refer to non-OpSeCom nodes or captured/compromised OpSeCom nodes. The verifier can then check whether the suspicious device can reply back with a proper OpSeCom CTS packet containing the committed bit string. If it is a non-OpSeCom node, the CTS-RA field will simply contain the copy the RTS-TA field. However, OpSeCom doesn’t address malicious insider attack.

4.2 Re-establishing or Updating Credentials

For re-establishing or updating the credential, nodes can covertly exchange the credential after loosely authenticating the nodes to each other using OpSeCom (repeat phase 3). The capability of generating an OpSeCom RTS packet and responding back with an OpSeCom CTS packet itself provides a means of authenticating the nodes to each other. The data being hidden in network packets can be treated as confidential.

4.3 General Discussion

The core security of our protocol lies in the fact that the covert operation is done at the MAC layer and it is independent of the control of application layer. When an OpSeCom node gets a special RTS packet, it replies back to it with corresponding bit strings in the CTS packet without consulting the upper layer which ensures the integrity of the bit string.

In the secure parameter agreement phase of OpSeCom, the nodes exchange bit commitment code words and the challenge sequence. Though, we assume that this phase is secure, it is possible that this phase could come under attack. For example, a malicious outsider could sniff these packets. However, having these packets will not be of much use. Because of the property of bit commitment, even if the malicious outsider has the bit commitment code words, they won't give it any information about the bit strings committed. Again, knowing only the challenge sequence is of no use if the attacker does not know where the covert channel exists. As we have described earlier, selection of shared secret algorithm and covert channel is open to the implementer. So, the covert verification phase is also secure as the existence of the covert channel is obscured to the outsider.

5. Covert Channel Analysis

We analyze the OpSeCom Storage Covert Channel according to the framework proposed by [11].

5.1 Generation of the Covert Channel

The widespread use of RTS/CTS protocol in wireless embedded systems communication lends itself to use for covert channels. In our particular example, the unique existence of RTS/CTS and its capability to reuse RA/TA makes it a good choice as a covert channel. Strictly speaking, the legal definition of the protocol is broken as it is no longer guaranteed that all RTS packets can be uniquely distinguished based on the RTS ID field. However, in practice the probability of a collision is low. When we are using 8 bits for randomly generating the RTS ID field, the probability that two simultaneously generated RTS have different ID is $(1-1/2^8)$ which is very high.

Setting up the OpSeCom channel is simple and straightforward. A modified version of MAC with the desired properties is all what is needed to be implemented on the embedded devices. The node can then decide when to trigger the channel according to the scenarios described in Section 2.

5.2 Detection of the Covert Channel

As the OpSeCom storage covert channel is a noisy channel, it will be very difficult to detect. Specially, we are using some usual values in RA/TA field when replacing our own data with the MAC address. The uniqueness of our scheme is that the communicating devices just need to know the challenge sequence and the method of generating response sequence (bit strings).

6. Performance Analysis

6.1 Recognizing OpSeCom Devices

The receiver device identifies OpSeCom senders by inspecting RTS TA field. If the sender implements OpSeCom, the most significant byte will be 02H and the lower 32-bits will contain the challenge sequence.

As explained earlier, a receiver implementing OpSeCom extension will copy the 16 most significant bits from RTS TA to its CTS RA, and set the remaining 4 bytes of CTS RA to the corresponding bit string. In contrast, an IEEE standard implementation will respond with CTS RA exactly the same as RTS TA. When the sender receives a CTS packet, it inspects the CTS RA field. If the lower order 32 bits are still set to the challenge sequence, the sender can determine that the receiver station does not implement OpSeCom, and act accordingly.

6.2 Bandwidth

The RA/TA field is 48 bits long. We are using 32 bits of it to carry cover data. A bandwidth of 32 bits per packet can therefore be achieved. RTS and CTS packet size is 20 bytes and 14 bytes respectively. So, In the case of RTS this approximates to one fifth of the bandwidth of the MAC layer packet flow. For CTS, this is about one fourth of the flow.

6.3 Backward Compatibility with standard IEEE 802.11 MAC

The modification to the RTS/CTS frame that we have proposed is compatible with the IEEE 802.11 MAC standard. The standard specifies the content of the CTS RA header to be the MAC address of the receiver. However, only the intended recipient of the frame needs to recognize that the packet is intended for it. Other stations use only the duration field of RTS/CTS frames to refrain from initiating communication. It follows that nodes implementing OpSeCom can coexist with standard IEEE 802.11 MAC terminals without any problem.

6.4 Effect on Overall Network Performance:

Existence of OpSeCom devices does not impact the normal network operation for two reasons. First, this method works at the MAC layer. Non-OpSeCom nodes will either discard the special RTS packet or reply with an invalid CTS packet. Either way, it is limited to one hop neighbors only. Second, the necessary information is being exchanged covertly in network control packets. So, there won’t be much communication overhead to hinder normal network functionality.

6.5 Comparison with Standard Cryptographic Solution:

We now address the practicality of our protocol by comparing the performance of OpSeCom with public key cryptographic methods [23] that are enhanced for embedded systems. We use energy consumption as the evaluation metric. Existing cryptographic methods such as RSA and ECC incur both computation and communication overhead. However, OpSeCom has a very negligible computation overhead because it uses bit commitment realized through simple exclusive OR operation which is a single CPU cycle instruction. This computation overhead is trivially small compared to modular exponentiation or point multiplication used by

RSA and ECC respectively. Therefore, we will calculate energy consumption due to communication.

For public key systems like RSA and ECC, one of the necessary operations is the public key transmission. Similarly, in OpSeCom, we need to have RTS/CTS transmission. For comparison purposes, we consider Imote2 sensor [24] clocked at 13 MHz as our target embedded system and calculate the energy consumption for these transmissions. Further, we assume its radio device (Chipcon's CC2420) to be operated with 0dB. At a supply voltage of 4.5V, the Imote2 draws a current of 31mA when the CPU is operating. From [24] we see that the current drawn for both sending and receiving is 44mA. Again, the 802.15.4 radio has an effective data rate of 250kb/s. So, the energy consumption E_{rs} for sending or receiving is: E_{rs}= (44mA×4.5V) / 250kb/s = 792μJ/b. The size of RTS and CTS packets together is 34 bytes. Therefore the energy consumption for transmitting (i.e., sending and receiving) RTS/CTS is 0.4J. For RSA with 1024 bit key, the energy consumption will be 1.6J. Similarly, 160 bit ECC transmission will cost 0.25J. However, as we mentioned earlier, apart from the communication overhead, RSA and ECC also have high computation cost. For example, in case of Imote2, energy consumption for *t* sec CPU operation is: E_{cpu} (t) =31mA×4.5V×*t* =139.5*t* mJ which will be quite significant.

7. Conclusion and Future Work

In this paper, we proposed a method of opportunistic secure communication in wireless embedded systems using shared secret and storage based network covert channels. We have compared the energy consumption of our protocol with existing cryptographic solutions using the Imote2 platform. Our proposed method enables neighboring nodes to achieve security goals without the help of a centralized authority or impacting the overall network operation. These characteristics make OpSeCom a good choice to enhance security in wireless embedded networks. Also, this method has flexibilities to choose the secret sharing algorithm and the network packets to be used as the cover object. In the future, we plan to implement it on different types of wireless networks and do more rigorous performance analysis such as interoperability with the standard MAC protocol, quantification of OpSeCom nodes' impact on network performance, and so on. We also have plans to do test bed experiments to determine the efficacy of the new protocol on a real system.

Acknowledgments. This research is supported in part by U.S. Department of Defense Grant No. H98230-07-1-0243.

References

1. Gligor, V.D.: A Guide to Understanding Covert Channel Analysis of Trusted Systems. Technical Report NCSC-TG-030, National Computer Security Center, Maryland (1993)
2. U.S. Department of Defense: TCSEC. DoD 5200.28-STD Washington (1985)
3. Gray, J.W.: Countermeasures and Tradeoffs for a Class of Covert Timing Channel. Technical Report, HKUST (1994)
4. Ahsan, K.: Covert Channel Analysis and Data Hiding in TCP/IP. Master's Thesis, University of Toronto (2000)

5. Ahsan, K., Kundur, D.: Practical Data Hiding in TCP/IP. In: Proc. Workshop on Multimedia Security at ACM Multimedia, Juan-les-Pins on the French Riviera (2000)
6. Virendra, M., Jadliwala, M., Chandrasekaran, M., Upadhyaya, S.: Quantifying Trust in Mobile Ad-Hoc Networks. In: Proc. Int. Conf. Integration of Knowledge Intensive Multi-Agent Systems (KIMAS), Waltham (2005)
7. Zhang, Q., Yu, T., Ning, P.: A Framework for Identifying Compromised Nodes in Sensor Networks. In: Securecomm and Workshops, Baltimore (2006)
8. Hwang, J., He, T., Kim, Y.: Detecting Phantom Nodes in Wireless Sensor Networks. In: 26th IEEE International Conference on Computer Communications, pp 2391--2395. IEEE Press, Anchorage (2007)
9. Liu, D., Ning, P., Du, W.: Detecting Malicious Beacon Nodes for Secure Location Discovery in Wireless Sensor Networks. In: 25th International Conference on Distributed Computing Systems, pp 609--619, Ohio (2005)
10. Wang, Z., Deng, J., Lee, R. B.: Mutual Anonymous Communications: A New Covert Channel Based on Splitting Tree MAC. In: 26th IEEE International Conference on Computer Communications, pp 2531--2535. IEEE Press, Anchorage (2007)
11. Llamas, D., Miller, A., Allison, C.: An Evaluation Framework for the Analysis of Covert Channels in the TCP/IP Protocol Suite. White Paper, ZDNet (2003)
12. Marti, S., Giuli, T. J., Lai, K., Baker, M.: Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In: Proc. of International Conference on Mobile Computing and Networking, pp 255-265, Boston (2000)
13. Yi, S., Naldurg, P., Kravets, R.: A Security-Aware Routing Protocol for Wireless Ad Hoc Networks. In: Proc. of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing, pp 299--302, Long Beach (2002)
14. Dini, G., Savino, I.M.: An Efficient Key Revocation Protocol for Wireless Sensor Networks. In: Proc. of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks, pp 450--452, Buffalo (2006)
15. Hoeper, K., Gong, G.: Key Revocation for Identity-Based Schemes in Mobile Ad Hoc Networks. In: Ad-hoc, Mobile and Wireless Networks, LCNS, vol. 4104, pp 224--237, Springer, Heidelberg (2006)
16. Zhang, Y., Lee, W., and Huang, Y.: Intrusion Detection Techniques for Mobile Wireless Networks. Wireless Networks, vol. 9, pp 545--556, Kluwer Academic Publishers, Hingham (2006)
17. ANSI/IEEE Std 802.11, http://ieeexplore.ieee.org/xpl/standardstoc.jsp?isnumber=30234
18. Serge, V.: On Bluetooth Repairing: Key Agreement based on Symmetric-Key Cryptography. In: Information Security and Cryptology, LCNS, vol. 3822, pp 1--9, Springer, Heidelberg (2005)
19. Hegland, A.M., Winjum, E., Kure, Ø., Mjølsnes, S.F., Spilling, P.: Key Management in Ad Hoc Networks, Survey and Evaluation, UniK report, Oslo (2005)
20. Damodaran, D., Singh, R., Phu D. L.: Group Key Management in Wireless Networks Using Session Keys. In: Proceedings of the Third International Conference on Information Technology: New Generations, pp 402--407, Las Vegas (2006)
21. Eriksson, J., Krishnamurthy, S. V., Faloutsos, M.: TrueLink: A Practical Countermeasure to the Wormhole Attack in Wireless Networks. In: Proc. of the 2006 IEEE International Conference on Network Protocols, pp 75--84, Santa Barbara (2006)
22. Schneier, B.: Bit Commitment. Applied Cryptography, Second Ed. pp 133--217, John Wiley and Sons, Inc. (1996)
23. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. Proc. of Workshop on Cryptographic Hardware and Embedded Systems, pp 119--132, Boston (2004)
24. Imote2 Datasheet: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2 _Datasheet.pdf