Selective and Secure Over-The-Air Programming for Wireless Sensor Networks

Nils Aschenbruck *•, Jan Bauer°, Jakob Bieling°, Alexander Bothe°, and Matthias Schwamborn*

*University of Osnabrück - Institute of Computer Science
Albrechtstr. 28, 49076 Osnabrück, Germany
{aschenbruck, schwamborn}
@informatik.uos.de

°University of Bonn - Institute of Computer Science 4
•Fraunhofer FKIE
Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
{bauer, bieling, bothea}
@cs.uni-bonn.de

***Abstract*—The growing range of Wireless Sensor Network (WSN) applications, their long-life and large-scale design, as well as various deployment fields necessitate the feasibility of remote maintenance and reprogramming of in-situ sensor nodes. The network-wide dissemination of program code is not appropriate in every WSN due to the heterogeneity of sensor hardware, the diversity of sensing tasks, and the event and location dependency of software. Thus, a flexible and group-wise selective Over-The-Air Programming (OTAP) is required in these scenarios. Furthermore, securing the OTAP protocol is imperative in order to prevent unauthorized and malicious reprogramming attempts. In this paper, we introduce SenSeOP, a selective and secure OTAP protocol for WSNs. For this purpose, the proposed protocol uses multicast transfer supported by asymmetric cryptography. We evaluate the performance of our approach in real testbeds, compare it with state-of-the-art protocols, and show that this approach enables efficient and reliable wireless reprogramming.**

***Index Terms*—Secure OTAP; Reprogram node attack; WSN**

I. Introduction

A Wireless Sensor Network (WSN) is composed of small and highly resource-constrained sensor nodes that monitor some measurable phenomenon in the environment, e.g., light, humidity, or temperature. WSNs are deployed in a steadily growing plethora of application areas. These range from military (e.g., security perimeter surveillance) over civilian (e.g., disaster area monitoring) to industrial (e.g., industrial process control). Application scenarios of WSNs typically involve monitoring or surveillance of animals or humans, infrastructure, or territories. Their long-life and large-scale design, various deployment fields, and changing environments necessitate the feasibility of remote maintenance and in-situ reprogramming of sensor nodes using a so-called Over-The-Air Programming (OTAP) protocol. In particular, if sensor nodes are inaccessible after deployment, a reliable OTAP is crucial.

We believe that in a plurality of WSNs, the network-wide dissemination of program code is not appropriate. Within a single WSN, the heterogeneity of sensor hardware, the deployment of manifold sensor technologies, the diversity of sensing and communication tasks, and possibly the event and location dependency of software require a flexible, group-wise selective OTAP approach in order to be able to efficiently reprogram a subset of nodes. Furthermore, securing the OTAP protocol is imperative in order to protect the OTAP from unauthorized reprogramming attempts, i.e., to prevent *reprogram node attacks*.

In this paper, we present SenSeOP, a *Selective 'n' Secure OtaP* protocol which is integrated in our intrusion detection system [1] and offers both, selective and secure reprogramming in WSNs. For our approach, we assume infrequent and non-regular software updates. On the one hand, these updates are supposed to be time- and energy-efficient. On the other hand, it is essential that the primary application running on the sensor nodes is interrupted by the OTAP process as little as possible. However, short interruptions of several seconds are inevitable due to the usage of strong cryptographic operations and must be tolerated by the application. Additionally, we assume that confidentiality is not required in these scenarios since no trusted information, e.g., symmetric keys, are exchanged using our SenSeOP protocol. However, if in a specific scenario confidentiality is demanded, it may be achieved by link-layer encryption, e.g., AES. Moreover, our SenSeOP protocol in its current version is tailored to scenarios with all nodes residing in communication range to each other.

The rest of this paper is organized as follows: We first discuss related work in Section II. A threat model leading to our security goals is presented in Section III. Details of the proposed secure OTAP protocol are described in Section IV and a performance evaluation is conducted in Section V. Finally, we conclude the paper and point out future work in Section VI.

II. Related Work

First research work related to OTAP can be found in the literature beginning with the approach of Reijers [18] and the In-Network Reprogramming tool XNP [4]. Whereas Reijers' approach essentially enables a *diff*-based incremental reprogramming scheme, entire program images are used in the so-called monolithic approach of XNP. Among others, the advantages and drawbacks of both approaches are discussed in [22] and [3] which, furthermore, provide a detailed survey of OTAP in WSNs.

In practice, incremental OTAP protocols based on Reijers (e.g., [9]) could not be established due to special requirements of hard- and software. In contrast, the research focused on the monolithic reprogramming family emerging with XNP and produced enhanced protocols and implementations. This family includes the well-known OTAP protocols MOAP [6], MNP [12], and Deluge [7]. The latter protocol, presented in 2004 by Hui and Culler, introduces a fragmentation scheme of the program image and supports a self-organizing multihop reprogramming realized by a sophisticated negotiation-based dissemination mechanism using Trickle [15]. Deluge is widely used and still state-of-the-art although there are many extensions proposed.

In the second half of the last decade, beginning with the distribution of WSNs and the deployment of OTAP, first questions concerning security arose (cf. e.g., [13]). In 2004, research efforts applying cryptography to sensor platforms (e.g., TinySec [10] and TinyECC [16]) enabled asymmetric cryptography based on Elliptic Curve Cryptography (ECC) in WSNs. However, asymmetric cryptography is expensive in terms of memory and computational runtime and must only be used sparsely. As a result, based on Deluge and the new opportunities, three OTAP protocols [5], [11], [14] were proposed in 2006 followed by [8] in 2008. These protocols leverage asymmetric cryptography in order to protect the OTAP from being exploited. Given the image fragmentation of Deluge, Sluice [14] uses hash chains for authentication based on pages, whereas Dutta et al. [5] apply hash chains on packet basis. In [11] and [8] a hybrid combination of hash trees and hash chains is used to enable a more flexible dissemination of the program code. Moreover, Seluge [8] and the confidential dissemination protocol proposed in [20] take specific signature-based Denial of Service (DoS) attacks explicitly into account. However, since each of the secure protocols mentioned above is based on Deluge, none of them supports a selective, group-wise OTAP.

III. Threat Model and Security Goals

Corresponding to [5], [8], [11], [14], we define the following generic *threat model* and *security goals* demanded by our approach.

- Sensor nodes deployed in the WSN are non-tamper-proof and may be compromised by an attacker.
- The base station acting as the OTAP gateway is assumed to be a trusted device which cannot be compromised.
- We assume an inside attacker who is able to *eavesdrop on*, *inject*, and *manipulate* packets in the WSN.

Arising from the threat model, the *security goals* stated below represent the essential requirements for secure OTAP protocols.

- *Reprogram node attacks resilience:* The main goal of the SenSeOP protocol is the resilience against *reprogram node attacks* with malicious code. Thus, the *authenticity* of each received program image has to be verified. It must be guaranteed that all devices solely reprogram authenticated images of *authorized* entities. Additionally, prior to reprogramming, the *integrity* of received images must be guaranteed since the assumed attacker may be able to manipulate data.
- *Replay attacks resilience:* A secondary goal is the resilience against *replay attacks*. It must be guaranteed that an attacker cannot replay eavesdropped program images. Furthermore, the attacker must not be able to exploit eavesdropped images to reprogram uninvolved nodes which are not addressed by this image.
- *Mitigation of DoS attacks:* An additional goal is the mitigation of *DoS attacks*. Using cost-intensive ECC in terms of computational power, it is possible that an attacker aims for a DoS attack by forcing the verification of malicious or faked images. This may lead to disruptions of the primary application and to the depletion of constrained energy resources.
- *Compromise-tolerant:* Also relevant in terms of security is compromise tolerance. A single compromised device must not allow the attacker to compromise other devices in the WSN.
- *Light-weight security solution:* Finally, the proposed security solution has to be light-weight in terms of required computational power and memory.

IV. Design and Implementation

In this section, design decisions based on the threat model and on the security requirements as well as details of the implementation are presented. Similar to the approaches [5], [8], [14] discussed in Section II, we decided to rely on TinyOS and asymmetric cryptography using public/private key pairs. The TinyECC library enables us to execute these cryptographic operations in a sufficient way (runtime of a few seconds).

Following the light-weight approach of NWProg [21] offered by BLIP, a collection of several IP-based protocols in TinyOS, our implementation is based on Deluge [7] and supports full and monolithic software updates. In order to make it light-weight, we replaced the complex dissemination algorithm with simple propagation. The reason for this replacement is that the dissemination algorithm is not feasible for scenarios we consider. Furthermore, it implies a huge memory demand which is adverse for the primary application with regard to the constrained memory resources of sensor nodes.

Similar to NWProg, our approach adopts several features of Deluge, e.g., the fragmentation of the image into a stream of several pages subdivided into packets, the insertion of metadata to the program image, the error detection mechanism, and the flash memory layout using multiple slots.

For the packet-based delivery of the program image, we adapted the message format from NWProg and extended the defined commands in order to enable the intended security mechanisms presented below. *Request* packets are generated by the OTAP operator using a python script running on the server. These packets contain a specific command (CMD), a slot number (SN), additional information (DATA), and optional payload. For instance, a WRITE command packet is provoking the receiver to write the data block included in the payload to the specified position (DATA) of the image slot defined in the SN field. An overview of all commands and the corresponding allocations is provided in Table I.

Packet header (1+1+2 byte)			**payload**	**remark**
CMD	**SN**	**DATA**	optional	
ERASE	*x*	–	–	Erasing flash memory slot *x*.
WRITE	*x*	*position*	*data*	Writing *data* into specified memory *position* of slot *x*.
BOOT	x	*time*	–	Booting of program code stored in slot *x* with a certain delay defined in *time*.
REBOOT	–	*time*	–	Rebooting of sensor node with a certain delay defined in *time*.
WRITE_ BEGIN	*x*	*counter*	–	Initialization of a WRITE stream with a predefined image *counter* to be stored in slot *x*.
WRITE_ END	*x*	–	*signature*	Conclusion of a WRITE-packet stream associated with slot *x* and delivery of the corresponding *signature*.

TABLE I
SENSEOP PROTOCOL COMMANDS AND THE CORRESPONDING REQUEST PACKET FORMAT.

Beside the request packets, *reply* packets (6 bytes) are used as acknowledgements (ACKs). Containing the header of the corresponding request packet (4 bytes), these packets signal the success of requests with an error field. Since we apply an Automatic Repeat reQuest (ARQ) mechanism with *stop-and-wait* strategy, sequence numbers are required, in particular for WRITE packets. The memory position stored in the DATA field of WRITE packets in combination with the slot number offers a unique identification and, thus, is leveraged as an implicit sequence number.

A. Security Concept

Our approach is enhanced by digital signatures based on ECC. Leveraging digital signatures, the protocol is able to ensure the *integrity* and *authenticity* of disseminated code images. Thus, the SenSeOP protocol has the ability to detect manipulated and malicious images and prevents their installation.

Using asymmetric cryptography, our approach is secure against *access node memory attacks* since only the public key is stored on each mote. The only instance disposing of the private key used to sign new code images is the legitimate OTAP gateway which is assumed to be tamper-proof. Hence, the protocol is *compromise-tolerant*. Based on performance evaluations of TinyECC, we decide to use a key size of 192 bits since we believe the stronger encryption justifies the low increase of the computational runtime. However, other key sizes are supported. Nonetheless, applying signatures to every packet is impractical due to the computational runtime of cryptographic operations. Instead, similar to [14] and [5], our protocol computes a single signature using the SHA-1 hash of the entire program image. In contrast to others, in our protocol, the cost-intensive cryptographic operation required by the verification is performed after receiving the entire packet stream whereas this is done prior to the image transmission in [14] and [5], causing additional effort.

Via a python script, the WSN operator is able to initiate an OTAP process indicated by a WRITE_BEGIN packet. Therefore, on a server, the program image is prepared and fragmented into a stream of packets. Moreover, the image is hashed and encrypted with the operator's private key in order to create the corresponding signature. To prevent *replay attacks*, a *version counter* is used which allows each client to distinguish between recent and obsolete versions. For this purpose, the version counter in combination with the destination address (either a specific node ID or the broadcast address) are considered in the hash computation as well. In the next step, subsequent to the initial WRITE_BEGIN, the packet stream is transmitted in several WRITE packets from the OTAP gateway to its client(s). While receiving the packet stream, each client reassembles the packets and instantly performs an iterative hash computation. Finally, the gateway sends the signature within a concluding WRITE_END message. After receiving and reassembling all packets and determining the hash value, the client verifies the *authenticity* and *integrity* of the delivered program code using this signature and the preinstalled public key. If the verification is successful, the receiver may boot the program code as soon as the corresponding BOOT packet is received. Otherwise, if the verification fails, the received data is deleted immediately since it is potentially malicious. Moreover, the SenSeOP module is locked for a predefined interval of time (currently 5 min) as a DoS countermeasure against possibly following attacks.

B. Selective Reprogramming

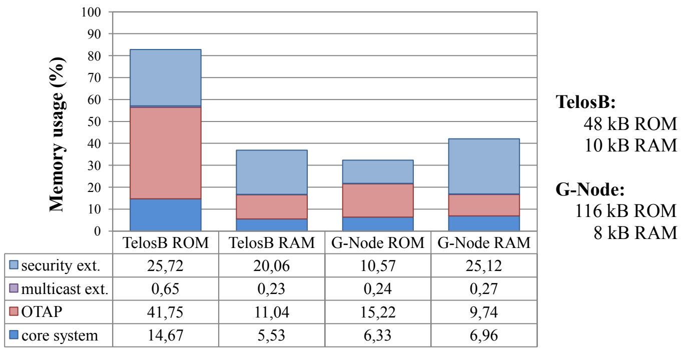
Neither Deluge nor NWProg support group-wise selective reprogramming of a subset of sensor nodes within a WSN. However, as mentioned in Section I, we believe this to be an inherent requirement of a proper OTAP protocol. For this reason, we extended our protocol with the ability to perform efficient selective reprogramming.

Since multicast is not supported natively in IEEE 802.15.4, we realized selective OTAP by using a specific multicast mode based on the explicit multicast (Xcast) approach presented in [2]. While sending a packet to the broadcast destination, an explicit list containing the unique addresses of the intended receivers is inserted into the packet header. In order to be able to distinguish between Xcasts and broadcasts, we replace this list with the broadcast address `0xffff` in case of an actual broadcast.

The explicit receiver list is stored in the optional payload field of the original NWProg frame and, thus, implies no revision of the chosen packet format. Indeed, there are two request packet types already occupying this payload field, namely WRITE and WRITE_END (cf. Table I). However, since both request types necessarily follow a corresponding WRITE_BEGIN packet and each receiver has to maintain the address list obtained by the WRITE_BEGIN anyway, the additional transmission of the address list would be redundant and can be omitted.

V. PERFORMANCE EVALUATION

In this section, the code size and the performance of the implemented SenSeOP protocol is evaluated for the two sensor platforms TelosB [17] and G-Node [19] using the 2.4 GHz and the 868 MHz frequency band, respectively. The goal of



	TelosB ROM	TelosB RAM	G-Node ROM	G-Node RAM
🟦 security ext.	25,72	20,06	10,57	25,12
🟪 multicast ext.	0,65	0,23	0,24	0,27
🟥 OTAP	41,75	11,04	15,22	9,74
🟦 core system	14,67	5,53	6,33	6,96

Fig. 1. Memory usage of a TinyOS core application enhanced with the SenSeOP module, depending on the hardware platform (TelosB vs. G-Node).

the performance evaluation is to study the reliability of the reprogramming operation and to demonstrate the benefit of selective OTAP using multicast mode. First, we present the memory footprint. Then, the evaluation setup is described and the metrics are introduced. Next, we present and discuss the results obtained by the evaluation. Finally, we compare our approach with both protocols NWProg and Deluge.

A. *Memory Footprint*

The code size resulting from our modular SenSeOP protocol integrated into an exemplary core application is summarized in Figure 1. For both platforms TelosB and G-Node, the memory usage is visualized, separated into the basic OTAP component and optional extensions which are successively added to the core application. One can observe that the optional but efficient multicast extension yields only minimal additional cost. In contrast, the OTAP component as well as the security extensions cause an inevitable memory demand. However, in particular for the G-Node platform, which offers more program memory (ROM), the achieved code size is appropriate for a deployment in practice since there is enough memory left for a reasonable application.

B. *Setup*

In the evaluation, two one-hop topology testbeds are deployed. One testbed is composed of five TelosB nodes in communication range of each other, whereas the second testbed consists of G-Nodes. In each testbed, one specific node attached to the PC and running the TinyOS `BaseStation` application is acting as an OTAP gateway. The other four nodes represent the clients which are intended to be reprogrammed. Therefore, a test application is preinstalled on these nodes. This application contains the realized SenSeOP module with all extensions and a simple blink component. The purpose of the blink application is to be able to distinguish between the current and the new application which is supposed to be installed via OTAP. Therefore, there are two alternative versions of this test application compiled in advance. Each version using a different LED color for its blinking. Using SenSeOP, in each step of the evaluation, the current version is replaced by its counterpart and vice versa.

As performance metrics, we consider the reliability of the SenSeOP operation as well as the latency and the overhead introduced by this protocol:

- *Reliability:* Aside from the robustness against *reprogramming node attacks*, the reliability of the SenSeOP protocol is an important requirement. The reliability is defined as the success probability of each reprogramming operation, i.e., whether the reprogramming operation was successful or not. In particular, a successful reprogramming operation includes the complete delivery of the new program image and its signature as well as the verification of its integrity and its authenticity.
- *Latency:* The latency of the reprogramming operation is defined as the period of time between the initial WRITE_BEGIN packet sent by the OTAP base station and the reply of the concluding WRITE_END packet sent by the receiver. As a result, the time required for the verification and the rebooting of the corresponding sensor node(s) is not considered by the latency metric.
- *Overhead:* In order to determine the protocol overhead, the overall traffic on application layer induced by reprogramming operations is considered. This is done by summarizing the size of all packets sent during these operations. This includes all request packets of the base station and the replies of every receiver. Furthermore, it is distinguished between control overhead, which is the traffic resulting from control packets and protocol headers, and the payload, which is associated with the program image. The latter includes all possible retransmissions belonging to the program image as well.

C. *Results*

For the performance evaluation of the SenSeOP protocol, three independent series of experiments (with ten replications each) are realized to study the performance of three available transmission modes. In the unicast mode, the OTAP aims for reprogramming one specific client, whereas all four clients are intended to be reprogrammed in the multi- and broadcast mode. The resulting standard deviations are shown in the figures using error bars. It is important to note that the supported data rate as well as the maximum transmission unit (i.e., the available payload size) depends on the frequency band and the hardware platform used. In the 2.4 GHz frequency band (TelosB) the nominal data rate accounts for 250 kB/s and with TinyOS the maximum payload size is 114 B, whereas using 868 MHz (G-Node), a nominal data rate of 38.4 kB/s (with Manchester encoding) and a maximum payload size of 52 B is possible. Furthermore, the binary size of the application image compiled for each platform differs slightly as well (TelosB: 46.752 kB, G-Node: 44.544 kB). Due to these differences, the image is fragmented into 487 packets in the TelosB scenario, whereas 928 packets are necessary using G-Nodes.

Reliability: We studied the probability of a successfully performed OTAP operation for both platforms TelosB and G-Node using each available transmission mode (uni-, multi-, and

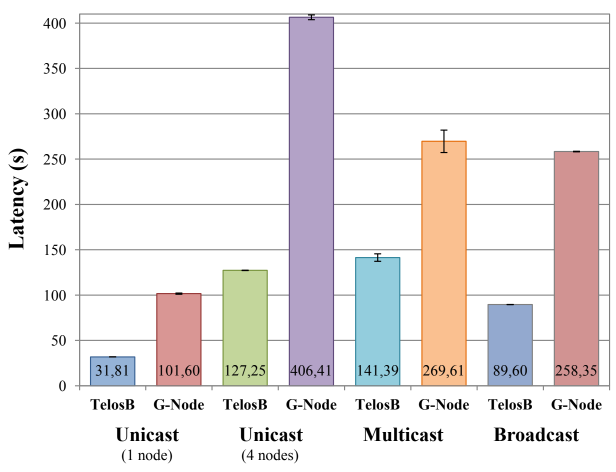


Fig. 2. Latency of SenSeOP operations using different transmission modes (TelosB vs. G-Node).

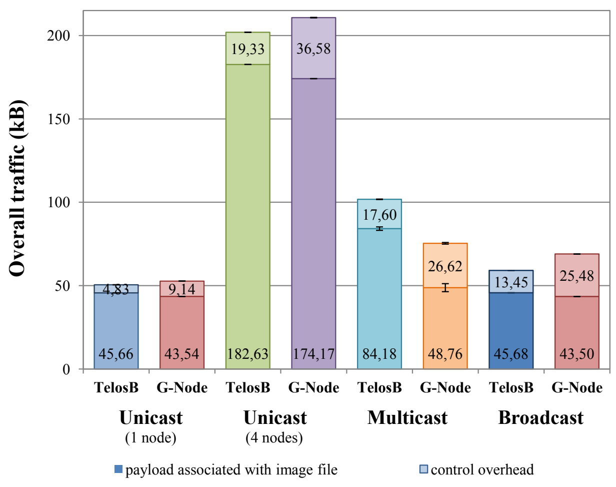


Fig. 3. Overall traffic produced by SenSeOP operations using different transmission modes (TelosB vs. G-Node).

broadcast). The evaluation confirms that the protocol is 100% reliable using the unicast or the multicast mode due to the ARQ mechanism. In contrast, using the broadcast mode, the possible number of receivers and, thus, the number of expected replies is unknown. Therefore, a packet retransmission is not assumed to be required as long as at least one reply message is received at the OTAP gateway. As a result, if merely a single packet of the image stream could not be received correctly at a destination while there exists another destination which returns a reply, no retransmit is performed. Thus, the integrity of the image cannot be guaranteed. This situation leads to the poor reliability of the broadcast mode (TelosB: $52\% \pm 13.27\%$, G-Node: $26\% \pm 12.81\%$) and is the reason for the high variance. Moreover, the success probability obtained by both platforms reflects the ratio of packets required for the image delivery. In other words, since in the G-Node scenario nearly twice the amount of packets are transmitted, the error probability concerning the entire packet stream is higher and, thus, the success probability is accordingly lower.

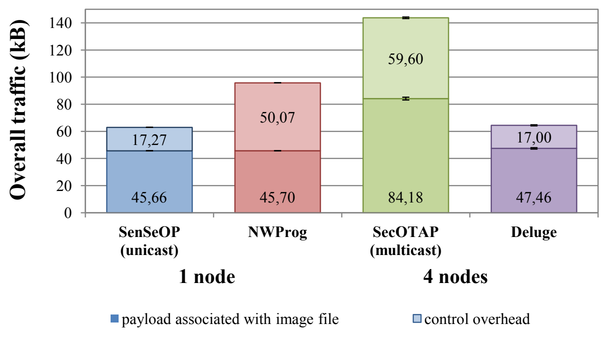
Latency: The latency induced by the OTAP operation (visualized in Figure 2) reflects the difference between the available data rates. Independent of the transmission mode, the higher data rate of the 2.4 GHz band (TelosB) allows for a significantly faster OTAP. The results show that the multicast mode causes a higher latency than the unicast mode *(1 mote)*. However, with regard to the number of nodes reprogrammed by the corresponding operation, the multicast mode clearly outperforms the unicast OTAP. For illustration, we determine the latency of reprogramming every node in the testbed using four successive unicast operations, denoted as *Unicast (4 motes)* in the figure. Taking the poor reliability of the broadcast OTAP into account, the multicast mode also outperforms this mode since a broadcast operation usually has to be performed several times due to its lower reliability.

Overhead: Finally, the overhead of the realized SenSeOP protocol was evaluated considering the overall traffic produced in this setup. The results are presented in Figure 3. Since in the G-Node testbed more packets are required for the OTAP, a larger control overhead is implied due to more reply messages. Furthermore, the adverse ratio of packet header and payload of the G-Nodes results in a higher control overhead as well. Both effects lead to the differences between the control overhead of TelosB and G-Node shown in Figure 3. It is worth to be noted that the traffic occurrence corresponding to the payload which is associated with the image file does not significantly increase if the multicast or broadcast mode is used instead of unicast OTAP. Hence, both modes significantly reduce the network load as well as the energy consumption. The only exception is given by the results of the multicast mode using the TelosB hardware. Here, a huge amount of retransmits is required to achieve the desired reliability. This could be explained by interference with other communication standards, e.g., WLAN and Bluetooth operating in the 2.4 GHz frequency band, as well. Similar to Figure 2, Figure 3 depicts the overall traffic of four successive unicast operations (*Unicast (4 motes)*) in order to emphasize the benefit obtained by the multi- and broadcast mode.

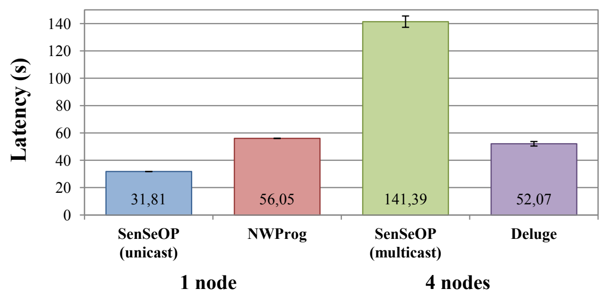
D. Comparison to related work

In order to compare SenSeOP with NWProg and Deluge, we consider the overhead and the latency of the reprogramming again. However, instead of measuring the overhead on application layer, the measurement is carried out on the physical layer taking the link layer ACKs used by NWProg into account. We apply the same setup described in Section V-B, the maximum payload size, and the default configuration of the protocols. Using NWProg and *SenSeOP (unicast)*, we reprogram one node, whereas all nodes are reprogrammed by Deluge and *SenSeOP (multicast)*. Due to space limitations, we just present results for the TelosB testbed.

The results are depicted in Figure 4 and show that in the unicast case, our secure approach achieves better performance than NWProg which uses link layer ACKs, resulting in a higher overhead and a higher latency in this scenario. Since our



(a) Overall traffic.



(b) Latency of OTAP operations.

Fig. 4. Comparison of SenSeOP, NWProg, and Deluge using the TelosB platform and different transmission modes.

approach cannot guarantee the success of the reprogramming using broadcast transmission, we use the multicast mode in the comparison with Deluge. Due to the usage of reply messages instead of selective negative ACKs, a higher overhead as well as an increased latency must be tolerated with our approach, as can be seen in Figure 4. However, considering that our SenSeOP is secure and offers a selective group-wise reprogramming, the additional costs are acceptable.

VI. Conclusion

In this paper, we presented the *Selective 'n' Secure OtaP* protocol (SenSeOP) leveraging ECC as a feasible countermeasure against wireless *reprogram node attacks*. After discussing related work, introducing the threat model, and presenting our security goals, details about the design and the implementation were provided. The rendered performance evaluation demonstrated the 100% reliability of our SenSeOP approach using the uni- and multicast mode. Besides the tolerable increase of latency, the multicast mode offers an appropriate and very efficient selective OTAP approach. The comparison with NWProg and Deluge showed that our approach outperforms NWProg and has acceptable additional cost compared to Deluge. In the future, we plan to extend our approach with multi-hop support. Due to constrained memory, an efficient multi-hop dissemination is very challenging, in particular in combination with the selective OTAP.

References

[1] N. Aschenbruck, J. Bauer, J. Bieling, A. Bothe, and M. Schwamborn, "A Security Architecture and Modular Intrusion Detection System for WSNs," to be published in: *Proc. of the 9th International Conference on Networked Sensing Systems (INSS '12)*, Antwerp, Belgium, 2012.

[2] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms, "Explicit Multicast (Xcast) Concepts and Options (RFC 5058)," http://tools.ietf.org/html/rfc5058 (November 2007).

[3] S. Brown and C. Sreenan, "Updating Software in Wireless Sensor Networks: A Survey," *Dept. of Computer Science, National Univ. of Ireland, Maynooth, Tech. Rep*, 2006.

[4] Crossbow Technology, Inc., "XNP: Mote In-Network Programming User Reference," http://www.tinyos.net/tinyos-1.x/doc/Xnp.pdf (2003).

[5] P. Dutta, J. Hui, D. Chu, and D. Culler, "Securing the Deluge Network Programming System," in *Proc. of the 5th Int. Conference on Information Processing in Sensor Networks (IPSN '06)*, Nashville, TN, USA, 2006, pp. 326–333.

[6] J. Heidemann, T. Stathopoulos, and D. Estrin, "A Remote Code Update Mechanism for Wireless Sensor Networks," Technical report, Center for Embedded Networked Sensing (CENS), University of California, CA, USA, Tech. Rep., 2003.

[7] J. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," in *Proc. of the 2nd Int. Conference on Embedded Networked Sensor Systems (SenSys '04)*, Baltimore, MD, USA, 2004, pp. 81–94.

[8] S. Hyun, P. Ning, A. Liu, and W. Du, "Seluge: Secure and DoS-Resistant Code Dissemination in Wireless Sensor Networks," in *Proc. of the 7th Int. Conference on Information Processing in Sensor Networks (IPSN '08)*, St. Louis, MO, USA, 2008, pp. 445–456.

[9] J. Jeong and D. Culler, "Incremental Network Programming for Wireless Sensors," in *Proc. of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON '04)*, Santa Clara, CA, USA, 2004, pp. 25–33.

[10] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," in *Proc. of the 2nd Int. Conference on Embedded Networked Sensor Systems (SenSys '04)*, Baltimore, MD, USA, 2004, pp. 162–175.

[11] I. Krontiris and T. Dimitriou, "Authenticated In-Network Programming for Wireless Sensor Networks," in *Proc. of the 5th Int. Conference on Ad-Hoc Networks & Wireless (ADHOC-NOW '06)*, Ottawa, Canada, 2006, pp. 390–403.

[12] S. S. Kulkarni and L. Wang, "MNP: multihop network reprogramming service for sensor networks," in *Proc. of the 25th Int. Conference on Distributed Computing Systems (ICDCS '05)*, Columbus, OH, USA, 2005, pp. 7 – 16.

[13] P. Lanigan, R. Gandhi, and P. Narasimhan, "Disseminating Code Updates in Sensor Networks: Survey of Protocols and Security Issues," Technical report, School of Computer Science, Carnegie Mellon University, PA, USA, Tech. Rep., 2005.

[14] ——, "Sluice: Secure Dissemination of Code Updates in Sensor Networks," in *Proc. of the 26th Int. Conference on Distributed Computing Systems (ICDCS '06)*, Lisboa, Portugal, 2006, pp. 53–53.

[15] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *Proc. of the 1st Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, CA, USA, 2004, pp. 15–28.

[16] A. Liu and P. Ning, "TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 2.0)," 2011, http://discovery.csc.ncsu.edu/software/TinyECC.

[17] MEMSIC, "TelosB Data Sheet," 2011, http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152.

[18] N. Reijers and K. Langendoen, "Efficient Code Distribution in Wireless Sensor Networks," in *Proc. of the 2nd ACM Int. Conference on Wireless Sensor Networks and Applications (WSNA '03)*, San Diego, CA, USA, 2003, pp. 60–67.

[19] SOWNet Technologies, "G-Node G301 Wireless Sensor Node," 2011, http://www.sownet.nl/download/G301Web.pdf.

[20] H. Tan, D. Ostry, J. Zic, and S. Jha, "A Confidential and DoS-Resistant Multi-hop Code Dissemination Protocol for Wireless Sensor Networks," in *Proc. of the 2nd ACM Conference on Wireless Network Security (WiSec'09)*, Zurich, Switzerland, 2009, pp. 245–252.

[21] TinyOS Community, "BLIP Tutorial - TinyOS Documentation Wiki," 2011, http://docs.tinyos.net/tinywiki/index.php/BLIP_Tutorial#Network_Programming.

[22] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming Wireless Sensor Networks: Challenges and Approaches," *Network, IEEE*, vol. 20, no. 3, pp. 48–55, 2006.