Secure Key Renewal in WirelessHART

Shahid Raza*, Gianluca Dini†, Thiemo Voigt* and Mikael Gidlund‡
*Swedish Institute of Computer Science, Kista, Sweden
{shahid, thiemo}@sics.se
†Dipartimento di Ingegneria dell'Informazione, University of Pisa
via Diotisalvi 2, 56100 PISA, Italy
g.dini@iet.unipi.it
‡ABB Corporate Research Centre, SE-72178 Västerås, Sweden
Mikael.Gidlund@se.abb.com

***Abstract*—WirelessHART is a wireless extension to the HART protocol. Even though WirelessHART is designed to be a secure protocol, the loopholes in the key management system makes it vulnerable to security threats. The broadcast approach for key renewal mechanisms in WirelessHART is not secure enough to be used in sensitive industrial automation environments where breach of security may result in catastrophic results. Also, key distribution with unicast communication with each device requires $O(n)$ rekeying messages, where n is the size of the network. In this paper we provide a secure and scalable key renewal protocol for WirelessHART that reduces the communication overhead to $O(logn)$. Our protocol requires far less messages than the conventional unicast approach.**

I. Introduction

WirelessHART™ [1] is an open standard for Wireless Sensor Networks (WSN) designed primarily for wireless communication in industrial process automation environments. Security is built into the WirelessHART standard. It uses symmetric cryptographic mechanisms to secure messages between the WirelessHART devices, both the wired (Gateway and Network Manager) and the wireless ones.

The security of the data, however, lies in the security of cryptographic keys. WirelessHART uses seven security keys to secure communication among different devices [2]. The security focus of WirelessHART is primarily communication security and related security aspects. WirelessHART does not address other types of security, such as physical device security or security of data at rest. Hence a WirelessHART enabled sensor device can be captured, cloned, and the security *keys* can be revealed.

A wise threat model cannot exclude that one or more sensor nodes can be compromised and this way an insider becomes present in the system. If this happens, the intruder holds all the keys of the compromised node and can thus inject modified and faked packets. It follows that removing the adversary presence from the system is a crucial reactive action. Physically removing the compromised node could be a radical but not always an efficient solution. Actually, physical removal requires to send a human operator into the field. This may be expensive especially if the field is remote and/or presents a harsh or even hostile conditions for a human. It follows that *logically* removing the compromised node from the network would be a more efficient solution that, anyway, would not exclude a later physical removal. A compromised node can be logically removed by revoking all its keys, preventing it from accessing future traffic. This requirement is also called *forward security* [3]. It is worthwhile to notice that key revocation would be necessary anyway even in the case of physical removal because the keys of the compromised node are in the adversary's hands.

Revoking WirelessHART unicast keys simply requires the Network Manager and the Gateway to delete it. In contrast, revoking the broadcast keys or network key is much more complicated task because all the remaining nodes in the system have to delete it and receive a new one. Unfortunately, WirelessHART provides no efficient support to this revocation operation. Actually, the Network Manager should build the new broadcast key bk and network key nk, and transmit them to every remaining node by using the Join key uk of that node. This would require $O(n)$ rekeying messages, where n is the size of the network. In the case of a WSN composed of many nodes, a rekeying communication complexity of $O(n)$ could be too large. A crucial challenge is thus to devise a more efficient group rekeying scheme for the WirelessHART protocol.

In this paper we present an efficient rekeying system for the WirelessHART network. Our protocol make use of one-way function trees and a key graph [4], [5] to delimit the network overhead. For the authenticity of messages we use one-way hash function we suggested earlier [6]. We could use digital signature but the WirelessHART standard does not support asymmetric cryptography [7]. The key chain based mechanism does not enforce backward security [3]. Therefore, to ensure confidentiality of previous messages we recommend the rekeying of the Network key at each join operation.

II. Background

Cyber-attacks to process control systems are getting more and more frequent and this trend is deemed to increase for several reasons. First, process control networks are getting wireless so attacking them from "outside" the plant premises is simpler and simpler. Second, ready-made or so called Commercial off-the-shelf (COTS) protocols and components are more and more used, so systems may inherit their vulnerabilities. WirelessHART is a clean example of this: Wire-

lessHART inherits a poor key management from the protocols it derives. Third, process control networks are not closed systems anymore as they are integrated with the corporate network. Fourth, the presence of insiders cannot be excluded. See the case of the Maroochy Shire, Queensland computerized waste management system [8]. Last but not the least, poor management is always an issue. Consider a case of a plant having an IP-based CC-TV system. On the outside perimeter of the plant, one of the cameras is missing but the Ethernet socket is available to every one. By simply plugging in, one can get access to plant internal network. The Stuxnet worm (http://en.wikipedia.org/wiki/Stuxnet) is another clean example of cyber-attack to process control system.

A. WirelessHART

WirelessHART is an IEC [9] approved standard for industrial process automation and control systems. The WirelessHART standard complements the Highway Addressable Remote Transducer (HART) with wireless connectivity. WirelessHART is a combination of wired and wireless devices. Network Manager (NM), Gateway, and Security Manager (SM) constitute wired devices. Wireless devices include field devices (FD), routers, access points (AP), handheld devices, and adapters. The wireless devices are connected through a mesh network where each device can act as a routing device. Figure 1 shows a sample WirelessHART network. The WirelessHART standard is based on the OSI seven layer

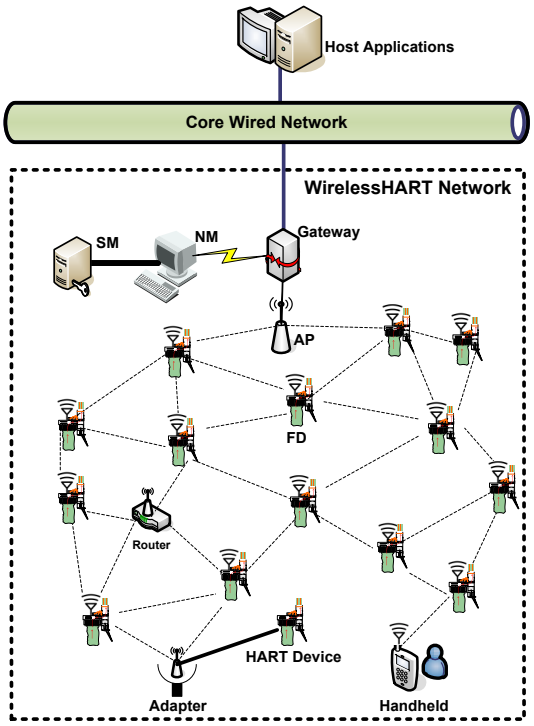

Fig. 1. A complete WirelessHART network with wireless and wired parts

architecture where session and presentation layers are not explicitly defined. The standard specifies completely new transport and network layers. The application layer is adapted from the HART protocol, and the data-link and physical layers are borrowed from the IEEE 802.15.4 standard. To provide frequency diversity and time diversity WirelessHART uses per message channel hopping using Frequency Hopping Spread Spectrum (FHSS)[1] and Time Devision Multiple Access (TDMA) in the MAC sub-layer at the data-link layer [10].

B. WirelessHART Security

WirelessHART is a secure and reliable protocol. Advanced Encryption Standard (AES-128) block cipher in Counter with CBC-MAC Mode (CCM*) [11] is used for encryption of messages and calculation of the Message Integrity Code (MIC). The WirelessHART standard provides end-to-end, per-hop, and peer-to-peer security. End-to-end security is enforced to secure the communication between the source and destination devices from malevolent insiders. The network layer provides end-to-end security. The data-link layer provides per-hop security between two neighboring wireless devices. Per-hop security is a defense against outsiders, i.e. devices that are not part of WirelessHART network. All traffic in the WirelessHART network flows through the Gateway; however, a Handheld device can create a secure and direct peer-to-peer session with a field device. We have earlier analyzed security in WirelessHART [2].

C. Key Management in WirelessHART

The WirelessHART standard specifies a security manager that is responsible for key management. The standard neither specifies the architecture of the security manager nor the functionalities for key management. However, the standard specifies some security related commands that can help to provide key distribution and key renewal. In WirelessHART networks all messages flow in the form of commands that are predefined in the standard. We have earlier designed and implemented a security manager for WirelessHART networks [12].

WirelessHART uses three group keys namely the Network, Broadcast Network Manager (BNM), Broadcast Gateway (BG), and four unicast keys namely the Join, Handheld, Unicast Network Manager (UNM), and Unicast Gateway (UG). Before joining the WirelessHART network each device must be provisioned with the Join key. A device uses the Device ID and the Join key to authenticate itself to the Network Manager. The Network Manager sends the verification request to the Security Manager that in turn authenticates the device with provided credentials. On successful authentication the Security Manager creates a Network key and four session keys (UNM, BNM, UG, and BG) and sends them to the wireless devices through the Network Manager. Now each device has session keys to establish end-to-end sessions with the destination devices and the Network key to secure links with the one-hop neighboring devices. The WirelessHART standard does not, however, provide a complete key management system hence

[1]DSSS is used for each message transmission and FHSS is applied on sequence of time slots.

an intelligent attacker can reveal a security key and launch an attack against the network; we have earlier identified security threats against WirelessHART [2].

The obvious counter measure against the compromise of keys is to change them regularly. However, the WirelessHART key renewal mechanisms do not provide strong protection mechanism against the known attacks. The lack of Public Key Infrastructure (PKI) in WirelessHART makes the key renewal process insecure. In the current WirelessHART standard, keys are interdependently used during the key renewal process and if one of the key is revealed the other is compromised as well. For example, the Join key is used to change the UNM key and the UNM key is used to change the Join key. The same is true for the Network key and the BNM. The problem is more severe in case of the Network key because the Network key is shared among all the devices and if revealed the attacker can send messages to all neighboring devices. A compromised node may also send messages carrying fake source addresses, in particular the Network Manager or the Gateway addresses. WirelessHART, secure key renewal is a challenge but necessary as the WirelessHART standard does not address physical device security and the security of data at rest.

III. An Efficient Group Rekeying Service for WirelessHART

We provide an efficient Rekeying Service that runs on top of WirelessHART and is composed of two components: the *Rekeying Server* running on the Security Manager and a *Rekeying Agent* running on each device of the network. The Rekeying Server's tasks consist in revoking the current network key, generating a new one and efficiently distributing it together with a proof of authenticity and freshness. In contrast, the Rekeying Agent's tasks only consist in verifying the authenticity and the freshness of the received keys and, if the verification succeeds, installing them.

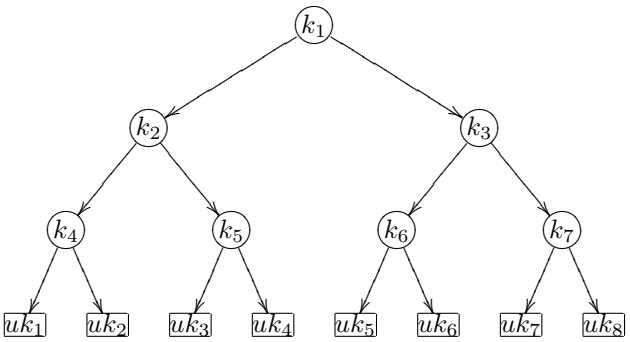

Fig. 2. A Logical Key Tree maintained by WirelessHART Security Manager (Rekeying Server).

To reduce the communication overhead, the Rekeying Server maintains a logical key tree (see Figure 2) where each internal node contains a *node key*, and each leaf is associated to a device and contains the unicast key uk of that device [4], [5]. We denote by k_ν and k_ν^+ respectively the current and next key of node ν.

Each device maintains a *key-ring* that contains every node key k_ν such that the sub-tree rooted at node ν contains the leaf associated with the node-key. Hence, with reference to Figure 2, the key-ring $\mathcal{KR}_4$ of device d_5 is $KeyRing_5 = \{k_1, k_3, k_6\}$. As it turns out, key k_1, associated to the key tree root, is shared by all devices and acts as the network key nk.

Let us now suppose that node d_5 leaves or is forced to leave the system. All keys in its key ring are considered compromised and the Rekeying Server has to distribute the respective next keys k_1^+, k_3^+, k_6^+ by means of the following rekeying messages

M1	$SM \rightarrow d_6$:	$E_{uk_6}(k_6^+)$
M2	$SM \rightarrow d_6$:	$E_{k_6^+}(k_3^+)$
M3	$SM \twoheadrightarrow \{d_7, d_8\}$:	$E_{k_7}(k_3^+)$
M4	$SM \twoheadrightarrow \{d_6, d_7, d_8\}$:	$E_{k_3^+}(k_1^+)$
M5	$SM \twoheadrightarrow \{d_1, d_2, d_3, d_4\}$:	$E_{k_2}(k_1^+)$

where $\rightarrow$ and $\twoheadrightarrow$ denote unicast and multicast communication respectively.

As it turns out the rekeying protocol requires $O(\log n)$ rekeying messages, where n is the number of devices. So the network and broadcast key distribution result highly scalable [4], [5].

A. Authentication or Integrity Service

It is important that when a node receives a rekeying message, after it has been properly decrypted, the authenticity of the key therein contained has to be verified. One possibility is to use conventional digital signatures. This means that the payload of rekeying messages M1–M6 becomes $(k, \sigma_{sm}(k))$, where $\sigma_{sm}(k)$ is the digital signature of k made by the Security Manager. While RSA signing is beyond the computation capabilities of simple devices, RSA verification has an acceptable computing overhead, e.g., about $14.5s$ on resource-constrained sensor nodes when it is implemented with small exponent ($e = 3$) [13]. However RSA causes quite a large message expansion as it requires to append a digital signature that is nowadays 1024-bit. For a 128-bits key, the expansion overhead is four times the key size. An alternative is to use digital signatures based on Elliptic Curves Cryptography (ECC) [13], [14]. In the case of ECC-160, the signature size is 40 bytes and thus the expansion overhead is two times and half the key size.

Another alternative is to use key chains as suggested in S^2RP [6]. Key chains provide an efficient key authenticity mechanism that requires only the computation of a hash function for verifying the authenticity of a key. Such a computation is $1 \div 2$ orders of magnitude faster than digital signatures. In addition, a proof of key authenticity does not require any additional information that causes message expansion as digital signatures.

In short, the key authentication mechanism levers on *key-chains*, a technique based on the Lamport's one-time passwords [15]. A key-chain is a set of symmetric keys such that each key is the hash pre-image of the previous one. A key chain of ℓ elements is built by initially assigning a random

value to $k^{(\ell)}$, $k^{(\ell)} \leftarrow random()$, and then applying ℓ times the hash function h such that $\forall i \in [1, \ell], k^{(\ell-i)} \leftarrow h(k^{(\ell-i+1)})$. We call $k^{(\ell)}$ the *chain tail* and $k^{(0)}$ the *chain head*.

Given a key $k^{(i)}$ in the key-chain, anybody can compute all the previous keys $k^{(j)}, j \leq i$, but nobody, except for the key-chain creator, can compute any of the next keys $k^{(j)}, j > i$. Keys are revealed in the reversed order with respect to the creation order. Given an authenticated key in the key-chain, anybody can authenticate the next revealed keys by simply applying a hash function. For example, if $k^{(i)}$ is an authenticated key, then anyone can verify the authenticity of $k^{(i+1)}$ by verifying that $k^{(i)} = h(k^{(i+1)})$. A node which receives the chain head in an authenticated way can later authenticate every key in the chain. When even the chain tail is disclosed, then the key chain is over, a new one has to be built and its chain head has to be distributed in an authenticated way.

Key chains can be integrated in the key tree by associating a different key chain to each tree node. The current key k_ν of node ν is the last revealed key in the chain whereas the next key k_ν^+ of that same node if the next key to be revealed in the chain. It follows that $k_\nu = h(k_\nu^+)$. At system set-up, every device is initialized with the heads of the chains related to its key ring. Later, when a new device joins the system, it is associated with a leaf and receives the current keys of the key chains related to its key ring.

Upon receiving a rekeying message, after it has been properly decrypted, the authenticity of the next key therein contained is verified by computing its hash and comparing the result to the corresponding current key. For instance, upon receiving rekeying message M5, d_4 decrypts the message by means of k_2 and verifies the authenticity of by ascertaining that $k_1 = h(k_1^+)$. If the test succeeds, the Agent installs key k_1^+ as the current (group) key, otherwise it drops the key.

B. Confidentiality Service

The rekeying service described above makes it possible to rekey the group and logically remove a compromised device, or supposed so, from the group. This is a reactive service that contributes to protect integrity because a suspect node is evicted from the group as soon as detected and thus cannot continue injecting modified or fake packets anymore. However, sometimes, industrial applications pose also some confidentiality requirements which have an impact on the rekeying strategy.

First of all, notice that in this case key authentication based on key chains may not be adequate. Let us assume that the adversary manages to compromise a given device d and steal the current (group) key $\widehat{k_1}$ that corresponds to the t-th key in the chain associated to that node. Given the way the key chain is built, the adversary can re-compute all the previous keys $k_1^{(i)}, i \leq t$, as $k_1^{(i)} = h^{(t-i)}(\widehat{k_1})$, where $h^{(r)}$ denotes the application of h for r times and $h^{(0)}$ is the identity function. It follows that the adversary can disclose all the past traffic encrypted by means of those keys. This attack is not possible in the case of digital signatures because the next key has no relationship with the current one. In the key chain, such a relationship, that is embodied by a hash preimage, is necessary to make key authentication more efficient. Such an increase in performance is paid in terms of reduced confidentiality.

Notice that merely using digital signatures is not sufficient. In the scheme described above, rekeying only occurs on leaving events. This means that if a node is captured, an adversary obtains a network key that dates back to the last leaving event. In order to reduce to the minimum the segment of traffic that can be disclosed after capturing a device it is necessary to rekey the system even at every joining event too. So doing a joining node is not able to access the traffic prior its joining. This requirement is also called *backward security* [3]. With reference to key tree in Figure 2, when device d_2 joins the group, for example, keys k_1, k_2, and k_4 have to be redistributed as follows

M1	$SM \rightarrow d_1 :$	$E_{uk_1}(k_4^+, \sigma_{sm}(k_4^+))$
M2	$SM \rightarrow d_1 :$	$E_{k_4^+}(k_2^+, \sigma_{sm}(k_2^+))$
M3	$SM \nrightarrow \{d_3, d_4\} :$	$E_{k_5}(k_2^+, \sigma_{sm}(k_2^+))$
M4	$SM \nrightarrow \{d_1, d_3, d_4\} :$	$E_{k_2^+}(k_1^+, \sigma_{sm}(k_1^+))$
M5	$SM \nrightarrow \{d_5, d_6, d_7, d_8\} :$	$E_{k_3}(k_1^+, \sigma_{sm}(k_1^+))$.

IV. Related Work

In this paper we discuss two mechanisms for key authentication, the one based on digital signatures the other one on key chains. Choosing the one or the other largely depends on the current deployment at hand. Notoriously, digital signatures are considered too computationally demanding for low-end embedded devices. However, things are changing on this front. ECC-based digital signatures have been proven viable even for this class of devices [13], [14], and ECC-based key establishment is part of the ZigBee security specification [16], [17]. On the other hand, key chains have been proposed for efficient authenticated key management in [6], [18]. Given their computing efficiency, key chains have been used as a building block for authenticated broadcast in WSNs such as μTesla and multi-level μTesla [19], [20]. We believe that for the sole purpose of authenticated key distribution, authenticated broadcast such as μTesla and its derivatives is not adequate because they require time synchronization and twice as many messages. However, certain solutions for mitigating DoS attacks conceived for authenticated broadcast could be integrated in key chains [21].

V. Conclusions and Future Work

We have shown that the WirelessHART standard has limitations regarding rekeying. To overcome these limitations we have presented a key renewal protocol. This protocol significantly reduces the communication overhead. It guarantees the integrity and confidentiality of data and preserves backward and forward security.

We are currently working on the design and implementation of a more complete WirelessHART network to more thoroughly evaluate the rekeying approach presented in this paper.

ACKNOWLEDGMENT

This work has been supported by the European Commission with contract FP7-2007-2-224053 (CONET), CHAT (Grant Agreement no. FP7-224428), and by VINNOVA. This work has been partly performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus and Peerialism.

REFERENCES

[1] A. N. Kim, F. Hekland, S. Petersen, and P. Doyle, "When hart goes wireless: Understanding and implementing the wirelesshart standard," *IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 899–907, September 2008.

[2] S. Raza, A. Slabbert, T. Voigt, and K. Landernäs, "Security considerations for the wirelesshart protocol," in *Proc. ETFA 2009*, Sep. 2009.

[3] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309–329, September 2003.

[4] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 444–458, May 2003.

[5] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, February 2000.

[6] G. Dini and I. Savino, "S^2RP: a secure and scalable rekeying protocol for wireless sensor network," in *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 06)*, Vancouver, Canada, October 2006, pp. 457–466.

[7] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer, December 2009.

[8] T. Smith, "Hacker jailed for revenge sewage attacks," October 2001, "http://www.theregister.co.uk/2001/10/31/hacker_jailed_for_revenge_sewage/".

[9] "Iec approves wirelesshart," *Control Engineering*, vol. 55, no. 10, pp. 34–34, October 2008.

[10] D. Chen, M. Nixon, and A. Mok, *WirelessHART: Real-Time Mesh Network for Industrial Automation*. Springer Verlag, 2010.

[11] D. Whiting, R. Housley, and N. Ferguson, *Counter with CBC-MAC (CCM), RFC 3610*. Fremont, California 94538 USA: IETF, Network Working Group, September 2003.

[12] S. Raza, T. Voigt, A. Slabbert, and K. Landernäs, "Design and implementation of security manager for the wirelesshart network," in *Proc. WSNS 2009*, Oct. 2009.

[13] R. J. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," in *SASN*, S. Setia and V. Swarup, Eds. ACM, 2004, pp. 59–64. [Online]. Available: http://doi.acm.org/10.1145/1029102.1029113

[14] D. J. Malan, M. Welsh, and M. D. Smith, "Implementing public-key infrastructure for sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 4, pp. 22:1–22:23, Aug. 2008.

[15] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, November 1981.

[16] G. Dini and M. Tiloca, "Considerations on Security in ZigBee Networks," in *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*. IEEE, 2010, pp. 58–65.

[17] "ZigBee Smart Energy Profile Specification, ZigBee Alliance," December 2008, http://www.zigbee.org/.

[18] G. Dini and I. M. Savino, "LARK: a Lightweight Authenticated ReKeying scheme for Clustered Wireless Sensor Networks," 2011.

[19] D. Liu and P. Ning, "Multilevel μTESLA: Broadcast authentication for distributed sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 4, pp. 800–836, 2004.

[20] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "SPINS: Security protocols for sensor networks," *Wireless networks*, vol. 8, no. 5, pp. 521–534, 2002.

[21] P. Ning, A. Liu, and W. Du, "Mitigating DoS attacks against broadcast authentication in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 1, pp. 1–35, 2008.