Key Update Strategies for Wireless Sensor Networks

Ender Yüksel, Hanne Riis Nielson, and Flemming Nielson

***Abstract*—Wireless sensor networks offer the advantages of simple and low-resource communication. Challenged by this simplicity and low-resources, security is of particular importance in many cases such as transmission of sensitive data or strict requirements of tamper-resistance. Updating the security keys is one of the essential points in security, which restrict the amount of data that may be exposed when a key is compromised. In this paper, we investigate key update methods that may be used in wireless sensor networks, and benefiting from stochastic model checking we derive characteristics of these methods in security perspective.**

***Index Terms*—Key management, security, sensor networks, verification.**

I. Introduction

Protection of data in transit is often provided by encryption algorithms that rely on cryptographic keys. Algorithms are assumed to be known by the attackers, whereas only secrecy of the key provides security as stated in Kerckhoffs' principle [1]. However, keys may get compromised over the time, therefore networks secured by encryption usually have the notion of key update where the current key is revoked and a new key is established.

Today it is a very well-known precaution that the cryptographic keys should be updated over time, while it is still not clear how and especially *when* the keys should be updated.

Several challenges are present in wireless sensor networks, in terms of cryptography. Most of the challenges arise from the constrained resources on the sensor devices. Memory and processing power limitations, as well as the demand for a long battery life generally eliminate the use of public key cryptography, therefore symmetric cryptography is widely used [2]. Besides, key types can also be affected by the limitations since pairwise keys (e.g. link keys) are more resource-consuming than group keys or network keys. In this study, we will focus on the symmetric network key updates.

Key update, or rekeying, is an area that very limited work has been done. In essence, it is related to two standard notions of secrecy. First notion is *forward secrecy, which* implies that compromise of the current key should not compromise any future key. Then comes *backward secrecy*, which implies that compromise of the current key should not compromise any earlier key. Wallner *et al.* suggested that a shared key has to be updated on *every* membership change and redistributed to all authorized members securely to provide backward and forward secrecy [3]. This all-or-nothing approach lacks flexibility to be suitable for low-resource networks where excessive number of updates can drain the batteries faster than expected and cause loss of functionality.

In this paper, we investigate characteristics of certain key update strategies that can be used in resource-constrained networks, using formal verification. We employ *stochastic model checking* techniques for reasoning about security, which allow deeper insights than qualitative techniques or simulation alone [4].

The main contributions of this paper can be summarized as:

1) we present a collection of key update methods that can be used with wireless sensor networks,

2) we establish a quantitative security analysis of key update methods using formal methods,

3) we provide a modelling example on ZigBee wireless sensor networks that can be generalized to other types of networks

4) we derive insights on the key update strategies based on analysis results, and interpretations of pros and cons in the security perspective.

Besides, we present an application of stochastic model checking in security domain.

The rest of this paper is organized as follows. In Section II, we define the key update strategies that we work on. In Section III, we present the details on constructing an analysis and the sample network that we work on. In Section IV, we interpret the analysis results and provide valuable information for not only researchers but also network experts that deploy security precautions; followed by the conclusion.

II. Key Update Strategies

In this section we explain the key update strategies that we consider in this work.

A. Time-Based Key Update (TB)

The notion of key update is often assumed to be *periodical,* such that key is updated after a certain time period. From now on we will refer to this periodical updating method as Time-based key update (TB). In contrast with other methods below, TB does not consider any key compromising event for triggering key update, except time.

B. Message-Based key Update (MB)

In this method, key is updated after a predefined number of messages are communicated [5]. The original method is actually proposed for pairwise keys, however we will consider MB for network keys so that it can be comparable with other methods in the paper. This approach counts the number of communicated messages and issues a key update

Manuscript received January 10, 2012; revised February 25, 2012. This work was supported in part by the IDEA4CPS project granted by the Danish Research Foundation for Basic Research, and in part by MT-LAB, a VKR Centre of Excellence for the Modelling of Information Technology.

The authors are with the Department of Informatics and Mathematical Modelling, Technical University of Denmark, Kongens Lyngby, Denmark (e-mails: {ey,riis,nielson}@imm.dtu.dk).

after a certain number of messages communicated. MB does not consider leaving (and joining) devices, therefore is not aware of a device leaving the network while still holding a valid key.

C. Join-Based Key Update (JB)

In this method, key is updated after a predefined number of new devices join the network [6]. Such an event presents a security risk since the device may become a legitimate attacker. For instance, it may have joined using an illegally obtained network key - which might happen by means of hardware (e.g. local key extraction from the chipset such as connecting a debugger, erasing the chip, then freely reading the contents of RAM), or software (e.g. a bug in the implementation that discloses the key after the session expires or terminates with the natural assumption that a new session key will be used for a future session) defects - and legitimate device can try to decrypt old communication that it has captured before joining the network.

D. Leave-Based Key Update (LB)

In this method, key is updated after a predefined number of devices leave the network [6]. In practice, when a device leaves the network it may still own a valid key, hence a device leave presents a security risk. A counter in the trust center keeps track of the number of the devices left (or removed from) the network. When this number reaches the predefined threshold value, all the keys in the network are updated and the counter is reset to zero. The idea here is to have a key update strategy that is inspired by the nature of the wireless sensor networks where the number of exchanged messages can be relatively low and devices can be tampered with by outside parties.

E. Join-Leave-Based Key Update (JLB)

In this method, key is updated after a predefined number of devices join *or* leave the network [6]. We consider each join and leave event as suspicious events and do not distinguish between them. JLB is powered by both JB and LB key update strategies therefore exhibiting the strength of these two. Since both join and leave events are considered, the threshold value is more sensitive than JB or LB.

F. Hybrid Key Update (Hy)

In this method, we employ multiple key update strategies, and issue an update whenever any of the update counters reaches its threshold [6]. All the counters (i.e. leave, join, message, and timer in this case) will be reset after the key update. In this way, we will be able to benefit from all useful key update strategies. Naturally, each key update strategy has a different strength *e.g.* performing well when too many leaves happen, or too many messages communicated, or the environment has less malicious activity, etc. In the hybrid key update strategy we will have all these strengths, with the cost of more computational power. Therefore we suggest the hybrid key update to be used in networks where the *coordinator* device has sufficient resources. For example, if the coordinator device is a mobile computer or a powerful handheld then we can implement the hybrid strategy. Note that we did not include JLB in Hy, since it is limiting a real hybrid phenomenon and we want to observe leave and join events separately.

III. Quantitative Analysis

A. Verification Technique

We employ *Stochastic model checking*, a powerful technique for verification and performance analysis of stochastic systems [4]. As a rough comparison with simulation which is widely used in networking perspective, one application of this method is equivalent to a sufficient number of simulation runs as it covers the full behaviour of the model and delivers provably correct results.

In the quantitative verification of all the key update methods considered, we formalized each method by doing necessary abstraction as needed for formal verification. In the network perspective, we focus on events such as joining and leaving devices, and messaging over the network. In the security perspective, we consider that leaving devices and messaging may cause key compromises. We designed our formal models to be stochastic, as we have events with stochastic delays such as device leave, device join, and messaging. As a result we developed a continuous-time Markov chain (CTMC) model for each of the key update methods.

CTMC models are well-suited for modelling of performance and reliability (e.g. of computer networks). In this formalism, transitions can occur at any (real-valued) time instant and they are modeled using exponential distribution, which is the only memoryless continuous distribution.

We specified the properties that we want to validate, in a stochastic temporal logic called Continuous Stochastic Logic (CSL) [7]. CSL is a temporal logic for describing properties of CTMCs. CSL introduces the probabilistic operator P, and the steady state operator S as an extension to the non-probabilistic temporal logic CTL. In addition to the standard steady-state and transient measures, CSL allows for the specification of probabilistic measures over paths through CTMCs.

To evaluate whether a CTMC conforms to a CSL property specification, we use a set of algorithmic techniques known as stochastic model checking. The basic algorithm for model checking a CSL formula is constructing a parse tree of the formula, and then recursively computing the set of states that satisfy each subformula upwards towards the root of the tree. In the end, for each state it is determined whether it satisfies the formula or not.

In this way, we are able check if the properties are satisfied and get quantitative answers. The model checking work is fully automated using the well-established stochastic model checker PRISM [8].

All the models and properties that are relevant to this paper and sufficient to replicate the results are available in [9], allowing customization for user defined networks.

B. Network

We work on *ZigBee* sensor network standard for the analyses throughout the paper. The latest specification of ZigBee (ZigBee-2007) [10] specifies a suite of security services that includes methods for key establishment, key transport, etc. Although each revision and supporting

specifications (such as stack and application profiles) roll out improvements, *key update* strategies and proper determination of related security parameters still remain gaps in the ZigBee standard [11].

We present the key points that are necessary for a clear understanding of our results, and omit all the details which are irrelevant to this study. ZigBee uses symmetric encryption, AES standard with 128 bits keys. A *Network Key* (NK) is the mere mandatory key in a ZigBee network, which is shared amongst all the devices and used to secure broadcast communications. A *Trust Center* (TC), creates and distributes the NKs. TC is an application running on a ZigBee device, that is unique in every ZigBee network. As a key component of ZigBee security, TC is assumed to run on a more powerful device (*e.g.* a coordinator) rather than a regular ZigBee end device. Two more types of security keys may exist in a ZigBee network depending on the security configuration: *Master Key* and *Link Key*. Unlike NK, those keys are pairwise shared. In this study we focus on NK as the key type and refer to it as the *key*, and we assume that the devices in the network already acquired the *key*.

At this point we want to clarify an ambiguity. ZigBee has a security protocol named as *NK Update*, however this protocol does not specify *when* to update NK, which is what we are focusing on in this paper.

C. Constructing the Analysis

Let us first introduce the main assumptions. We assume that when TC updates the key, all the devices in the network successfully update their keys. Compromise of a NK affects all the devices in a network. TC is fully responsible of creating and distributing the NK, therefore there is no role of the devices in NK establishment. We assume that each device talk directly to the network coordinator, namely in star topology. This assumption makes sense especially in MB method, where coordinator is able to count all the messages.

We configured all models with the same parameter values as we have listed in Table 1. For simplicity, we use a single parameter for key compromise per device which is the same for compromise by messaging and compromise by device leave. In addition to those parameters, we also have a unique threshold parameter for each key update method. The threshold values vary in order to get more different results, and are available on the result graphics in Fig. 1.

Further details of the verification technique, network, and analysis construction are available in [12].

IV. Characteristics of Strategies

In this section, we interpret the quantitative analysis results for each key update strategy by identifying strengths and weaknesses. In addition, we provide insights on verification perspective. We provide the graphical results of the analysis in Fig. 1 where transient probability of the network key being compromised is considered for each strategy. We excluded the numerical results since they may change when different parameter values are used. Instead we focus on the patterns in transient key compromise probability which define the characteristics of the considered strategy.

TABLE I. Model Parameters

Parameter	Description	Value
Rjoin	Rate of join, per device	1/180
Rleave	Rate of leave, per device	1/180
Rmessage	Rate of messaging, per device	1/15
Pcomp	Probability of compromise, per device	1/1000
Size	Initial and maximum network size	10

TABLE II. Update Thresholds

Threshold	Strategy	Description
M	TB	Number of months
MSG	MB	Number of messages
J	JB	Number of joining devices
N	LB	Number of leaving devices
JL	JLB	Number of joining or leaving devices

We have used the input parameter values that we presented in Table 1, and we have issued a model checking for each time instant (and for each threshold value) that we specified in Fig. 1. The time unit (in the *x*-axis) in Fig. 1 is a *month* (30 days). We also specified the threshold values that we used for analysis in Fig. 1, attached to each subfigure. The mapping of threshold names to update strategies is presented in Table II, including with the descriptions. One exception is the Hy method, where the threshold parameter is a collection of the corresponding threshold values of the methods that are included. For example, Hy=1 stands for M=1, MSG=20, J=1, N=6 in Fig. 1.

A. Time-Based Key Update (TB)

Transient behaviour: TB exhibits a repeating sawtooth pattern where the probability of key compromise increases until a key update happens, and a sharp drop is occurring afterwards. Sawtooth pattern is not visible for sufficiently small threshold values, in other words update periods.

Strength: This classical key update strategy is strong because it is robust and it does not allow any unexpected updates which can increase power consumption. Namely, it is easy to foresee how many key updates will take place, and it is easy to guess the impact of a change on the threshold value.

Weakness: TB does not consider any source of key compromise, its mere concern is the time passed. Therefore it cannot be fitted easily to a specific network that has distinct properties such as frequency of communicated messages, joining/leaving devices etc. Besides there is always a significant difference between local maxima and minima of the risk due to the sawtooth pattern. Therefore, computing average risk is not sufficient in TB, maximum risk should also be computed before employing it as key update method.

Verification perspective: Even though join, leave and messaging events exhibit stochastic delays, periodical updates have *deterministic* delays. Thus we improved the model with phase-type distribution in order to increase

precision, which causes the model size to grow by factor of a shape parameter. This makes formal verification of TB difficult for large networks. However, produced state-space and model checking times are still better than MB.

B. Message-Based key Update (MB)

Transient behaviour: MB produces oscillations in the key compromise probability, which get stabilized after a certain time period that depends on the threshold value. For sufficiently small message thresholds the oscillations disappear.

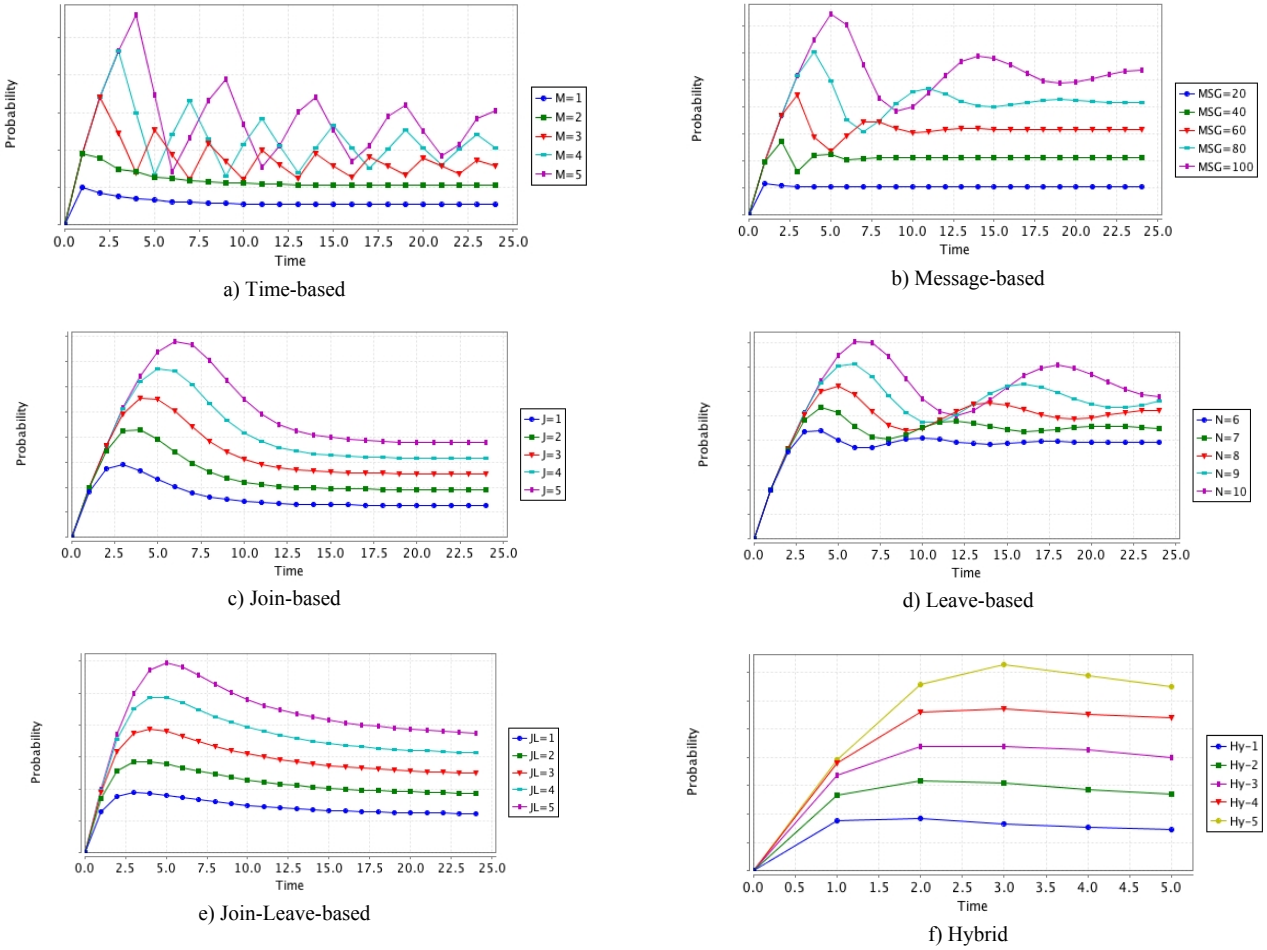
Strength: MB is very strong when majority of the key compromise events are caused by the messages over the network. Therefore, it should be considered when devices send vast amount of messages.

Weakness: MB does not consider neither leaving nor joining devices, therefore ignores the risk of a device leaving with a valid key. Thus, in dynamic environments in terms of network membership MB itself is not strong.

Verification perspective: MB is only better than Hy, in terms of state-space and model checking time. Number of messages that are communicated in a network for a certain time amount is directly proportional to the network size and messaging activity per device, therefore can quickly reach to very large numbers. This causes a big burden on the stochastic model checking since large numerical threshold values quickly exceed time and memory limits. We easily observed situations like state-spaces of 10^8, required memory of 3GB and fairly long computation times.

C. Join-Based key Update (JB)

Transient behaviour: In the beginning, key compromise probability increases with a decreasing slope until it reaches the maximum probability value. Then comes the smooth drop in the probability that lasts until reaching the steady-state.



a) Time-based

b) Message-based

c) Join-based

d) Leave-based

e) Join-Leave-based

f) Hybrid

Fig. 1. Probability of network key being compromised at a certain time instant for different threshold values.

Strength: In a network where number of join events is reasonably high, JB will perform better and have more control on the key compromise probability adjustment. Minimum threshold values can offer very high security, provided that compromises mostly happen by joining and leaving devices.

Weakness: Naturally, a join event can only happen whenever there is room for a new device. If a network is fully utilized, then a join event is strictly bound to a leave event. Therefore, in such cases not only the rate of join but also rate of leave should be considered when assigning thresholds.

Verification perspective: JB is very compact in model size, therefore one of the strongest strategies against state-space explosion.

D. Leave-Based Key Update (LB)

Transient behaviour: LB also produces oscillations in the key compromise probability, which reaches steady-state after a certain time period that depends on the threshold value.

Strength: LB was inspired by the leaving devices who still possessed the valid network key. Therefore is very useful when a network is not stationary but dynamic.

Weakness: Whenever a network is stationary, LB will fire

less key updates which might be insufficient for high security concerns.

Verification perspective**:** LB is a very compact model, like JB; therefore can successfully be used in verifying very large networks.

E. Join-Leave-Based Key Update (JLB)

Transient behaviour: Even though JLB is a synthesis of JB and LB, the resulting transient behaviour actually resembles to JB. Yet, it exhibits a smoother stabilization pattern.

Strength: JLB is powered by both JB and LB key update strategies therefore employs the strong parts of these two. Since both join and leave events are considered, a threshold value is twice more sensitive than JB or LB. In other words, we have more precision on the key update thresholds.

Weakness: If the majority of the key compromise events occur from the communication over the network, then JLB might not be the optimum strategy or the threshold should be minimized to deal with this problem.

Verification perspective: JLB is larger than JB and LB as a model but still produces less state space than TB, MB, and Hy key update strategies. JLB is not as compact as the models that inspired it, namely JB and LB.

F. Hybrid Key Update (Hy)

Transient behaviour: Hy exhibits a very stabile transient behaviour, where we do not observe significant fluctuations. Thus, its maximum risk is very close to its average risk.

Strength: This strategy incorporates the strengths of all the other strategies excluding JLB, therefore we consider it to be the strongest one. An update is issued by considering the time passed, the number of devices joined, the number of devices left, and the number of messages sent.

Weakness: Implementing this strategy on real devices require much more resource than implementing only one of the strategies above. Therefore, if the coordinator device (that is running the trust center application) that is responsible of key updates is not strong enough, then Hy won't be feasible to choose.

Verification perspective: Hy is very modular, it is easy to add a new key update strategy, or remove an unwanted key update strategy. Technically, the maximum state-space is produced in Hy models, and in parallel with that the time needed to complete one model checking is larger than in any other strategy. However, we can exclude some of the heavy strategies such as TB or MB to customize Hy to have more performance.

V. Conclusion and Future Work

In this paper, we discussed the problem of updating the cryptographic keys in wireless sensor networks. We have studied different key update methods and produced insights supported by formal verification and quantitative analysis. As a result we identified strengths and weaknesses of the methods we considered, which will assist network designers and security experts to employ a key update method. Our future work will be on how these key update methods adapt to changes in network conditions.

References

[1] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, vol. IX, pp. 5-83, January 1883.

[2] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: A survey," in *Security in Distributed, Grid, and Pervasive Computing*, Ed. Yang Xiao, CRC Press, 2006.

[3] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key management for multicast: Issues and architectures," *IETF*, RFC 2627, Jun. 1999.

[4] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal Methods for the Design of Computer, Communication, and Software Systems*, ser. LNCS, M. Bernardo and J. Hillston, Eds., vol. 4486. Springer, 2007, pp. 220–270.

[5] J. V. Misic, F. Amini, and M. Khan, "Performance implications of periodic key exchanges and packet integrity overhead in an 802.15.4 beacon enabled cluster," *Int'l Journal of Sensor Networks*, vol. 3, no. 1, pp. 33–42, 2008.

[6] E. Yüksel, H. R. Nielson, F. Nielson, M. Fruth, and M. Kwiatkowska, "Optimizing zigbee security using stochastic model checking," Technical University of Denmark, Tech. Rep. *IMM-Technical Report*-2010-08, 2010.

[7] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton, "Verifying continuous time markov chains," in *Proc. of the 8th International* Conference *on Computer Aided Verification*, London, UK, Springer-Verlag, 1996, pp. 269–276.1996.

[8] The PRISM model checker website. [Online]. Available: http://www.prismmodelchecker.org

[9] The formal models for key update methods. [Online]. Available: http://www2.imm.dtu.dk/people/ey/models/

[10] ZigBee *Specification*, ZigBee Alliance Std. 053 474r17, 2008.

[11] E. Yüksel, H. R. Nielson, and F. Nielson, "Zigbee-2007 security essentials," in *Proc. of the 13th Nordic Workshop on Secure IT Systems*, Copenhagen, Denmark, 2008, pp. 65-82.

[12] E. Yüksel, "Qualitative and quantitative security analyses for zigbee wireless sensor networks," Ph.D. dissertation, Department of Informatics and Mathematical Modeling, Technical University of Denmark, Kongens Lyngby, Denmark, 2011.

Ender Yüksel received the B.Sc. and M.Sc. degrees in computer engineering from Istanbul Technical University, Turkey, in 2003 and 2007, respectively, and the Ph.D. degree in computer science from the Technical University of Denmark, in 2011. He is currently a postdoctoral researcher with the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. His research interests include security and performance in wireless sensor networks, qualitative and quantitative security analysis, and formal methods for verification.

Hanne Riis Nielson received the M.Sc. degree in mathematics and computer science from Aarhus University, Denmark, in 1980, and the Ph.D. degree in computer science from Edinburgh University, UK, in 1984. She is currently a professor with the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. Her research interests include development of theories, techniques and tools for the modelling, analysis, validation and realisation of software systems. This includes process calculi, static analysis, model checking, and qualitative and quantitative analysis of, among others, safety and security properties.

Flemming Nielson received the M.Sc. degree in mathematics and computer science from Aarhus University, Denmark, in 1981, the Ph.D. degree in computer science from Edinburgh University, UK, in 1984, and the D.Sc. degree in computer science from Aarhus University, Denmark, in 1990. He is currently a professor with the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. His research interests include process modelling, static analysis, abstract interpretation, type and effect systems, operational semantics, model checking, qualitative and quantitative analysis, and software security.