Contents lists available at www.innovativejournal.in

Asian Journal of Computer Science and Information Technology

Journal homepage:http://www.innovativejournal.in/index.php/ajcsit

ISSN 2249 - 5126

A SECURITY OF WIRELESS SENSOR NETWORKS – ANALYSIS ON EFFICIENT BROADCAST AUTHENTICATION

N. Vikram Narayanadas*[1], N.Sainath[2]

[1]*JayaPrakash* Narayana Co*llege of Engg. Mahabubnagar, A.P., India*.

[2]*Sree Vishweshawarya college of Engg, Bareilly, Uttar Pradesh, India*

Contents lists available at www.innovativejournal.in

Asian Journal of Computer Science and Information Technology

Journal homepage:http://www.innovativejournal.in/index.php/ajcsit

ISSN 2249 - 5126

A SECURITY OF WIRELESS SENSOR NETWORKS – ANALYSIS ON EFFICIENT BROADCAST AUTHENTICATION

N. Vikram Narayanadas*[1], N.Sainath[2]

[1]*JayaPrakash* Narayana Co*llege of Engg. Mahabubnagar, A.P., India.*

[2]*Sree Vishweshawarya college of Engg, Bareilly, Uttar Pradesh, India*

ARTICLE INFO

Corresponding Author:
N. Vikram Narayanadas
JayaPrakash Narayana College of Engg. Mahabubnagar, A.P., India.
vikram.shadan@gmail.com

KeyWords: Wireless sensor Networks, Authorization, Security, Trade-off, Broadcast.

ABSTRACT

A Broadcast Authentication is enabling with base station to send commands and requests to low-powered sensor nodes in an authentic manner, is one of the core challenges for securing wireless sensor networks. The multi-level variant of µTESLA based on delayed exposure of one-way chains are well known valuable broadcast authentication schemes, but concerns still remain for their practical application. To use these schemes on resource-limited sensor nodes, a 64-bit key chain is desirable for efficiency. Our work show, by both theoretical analysis and rigorous experiments on real sensor nodes, that if µTESLA is implemented in a raw form with 64-bit key chains, some of the future keys can be discovered through time memory data tradeoff techniques. This paper presents an extendable broadcast authentication scheme called X-TESLA, as a new member of the TESLA family, to remedy the fact that previous schemes do not consider problems arising from sleep modes, network failures, idle sessions, as well as the time memory data tradeoff risk, and to reduce their high cost of countering DoS attacks. In X-TESLA, two levels of chains that have distinct intervals and cross-authenticate each other are used. This allows the short key chains to continue indefinitely and makes new interesting strategies and management methods possible, significantly reducing unnecessary computation and buffer occupation, and leads to efficient solutions to the raised problems.

INTRODUCTION

Todays Technology advancement in large scale distributed networking and small sensor devices has led to the development of wireless sensor networks with numerous applications. Sensor nodes are usually constrained their computation, communication, storage, and energy resources for economical reasons, but need security functions since they are deployed in unattended or even hostile environments. The high risk of physical attacks and the limited capabilities of sensor nodes make it difficult to apply traditional security techniques to wireless sensor networks, posing new challenges. Authenticated broadcast, enabling a base station to send authentic messages to multiple sensor nodes, is one of the core challenges, while even the broadcast by nodes is an important topic in wireless sensor networks. For the purpose, digital signatures (public-key) are not very useful in a resource limited environment, while native use of HMAC (secret-key) does not work either, as node capture can lead to a key compromise. μTESLA and its multi-level variants , based on TESLA], use a one-way chain practically under a loose time synchronization assumption. The sender attaches a MAC (Message Authentication Code) to each packet, computed using a key from the chain in reverse order. The keys are exposed after a certain time delay. The receiver buffers the received packet until the corresponding key is disclosed and verifies the MAC, after authenticity of the key itself has been verified by following through the chain.

Few Motivation points & Problems Mentioned

μTESLA and its variants are designed to be practical, but significant concerns still remain.

1 64-bit key chain: A short 64-bit key chain is desirable for efficiency in resource-limited sensor nodes, but care must be taken, even with short time intervals. As we show, if the chain is generated in a straightforward manner, TMD(Time Memory Data) tradeoff techniques can be applicable, leading to discovery of future keys.

2 Sleep mode or network failure: If sensor nodes go into a sleep mode or key disclosure messages are lost frequently, μTESLA may force heavy key computation to be done at once on sensor nodes for chain verification, during which incoming packets get dropped. If CDMs (Commitment Distribution Messages) are missing, multi-level μTESLA makes nodes wait and buffer for the long interval of upper levels, during which incoming packets are dropped due to the buffer limit.

3. Idle sessions: Even for idle sessions with no broadcasts, μTESLA forces chain computation for sensor nodes. Key disclosure messages should be broadcast constantly or heavy computation needs to be done later. Multi-level μTESLA needs CDMs to be broadcast for higher levels, with the number of CDMs increasing with the number of levels.

4 Extended lifetime: With node malfunctions and premature power exhaustion, there are needs for node additions [1] or rechargeable sensor nodes [15]. Thus, the lifetime of a network may extend beyond that of each node. As noted in ,lifetime extension was not clearly considered in μTESLA. Multi-level μTESLA should also fix the lifetime.

5 DOS attacks: To resist DOS attacks, multi-level μTESLA requires many CDMs to be distributed for longer intervals. Its DOS tolerant version needs sufficiently large buffers on sensor nodes for random selection of received CDMs. The DOS resistant version requires CDMs to be received stably along with a larger packet and additional hash function.

The remainder of this article is organized as follows. In Section 2, we review related work on broadcast authentication for wireless sensor networks, and discuss their problems and shortcomings. In Section 3, we show how TMD-tradeoff can be applied to μTESLA with a detailed attack algorithm and also a concrete implementation result. In Section 4, we introduce an extendable broadcast authentication scheme called X-TESLA. Security and performance of X-TESLA are analyzed in Section 5. We conclude this paper in Section 6.

SECTION II

2 Survey over the security of Cryptosystems.

Implementation of public-key cryptosystems is becoming possible, but still expensive. Energy efficient sensor nodes are also great concerns [5]. More practically in this section, we briefly review μTESLA and its multi-level variant for moderate sensor nodes ,.All these schemes are constructed without using public-key cryptography.

1 μTESLA: We give a short description of μTESLA, referring readers to for more detail. μTESLA is a broadcast authentication mechanism for distributed sensor networks, which was adapted from TESLA . In short, a delayed exposure of one-way chain is used for authentication. For this, it is required that the base station and sensor nodes be loosely time synchronized with a known maximum synchronization discrepancy bound. Unlike TESLA, which authenticates the initial packet with a digital signature, μTESLA uses only symmetric key techniques. The sender first fixes a public one-way function F and chooses a random value Kn. The one-way chain $Ki = F(Ki+1)$ is iteratively calculated for all $n > i \geq 0$ and the last element $K0$ is pre-installed in each receiver, the sensor node, as an initial *commitment*. μTESLA also provides a method for bootstrapping a new receiver through *unicasting*. Time is divided into short intervals. During the *i*-th interval *Ii*, the messages broadcast are sent with a MAC keyed with *Ki*. After a suitable delay, the key *Ki* itself is broadcast. Given a key *Ki*, calculating *K* for $j > i$ is expected to be infeasible, but anybody can calculate *Kj* for $j < i$, so it is easy to check the validity of any newly received *Ki* with the commitment *K*0, or any other *Kj* satisfying $j < i$.

2 µTESLA Multi-Level: One drawback of the single chain used in μTESLA is that there is a practical limit to its length, leading to a usage time limit. Also, the bootstrapping of a new receiver in μTESLA utilizes unicasting and hence is not scalable. Multi-level μTESLA solves this problem by using multiple chains in multiple levels. Several levels of chains are used with each (except for the top) level consisting of multiple chains. The lowest level is a normal chain used for message broadcasts and usually lasts for a relatively short period. Upper levels exist to authenticate their very next lower level chains. When the lowest level chain draws to an end, the second level chain is used to authenticate the commitment for the next lowest level chain to be used. The second level broadcasts the CDM, with clear expression of *i* in the MAC only,

$CDMi = i\_Ki+2,0\_ \bot\_MACKi\ (i\_Ki+2,0\_ \bot)\_Ki-1,$

Note that it means a null value which is ignored in a basic version but will be replaced by a hash value in a DOS resistant version. The new commitment *Ki*+2,0 for the lowest level is authenticated with the key2 *Ki*. Note that *Ki*+2,0, the lowest level commitment corresponding to the ($i + 2$)-th second level interval, can only be verified by the sensor nodes after receiving CDM*j* with $j > i$. To authenticate a new 2nd level chain commitment, the 3rd level is used, and so on. The top level is a single chain that has to last as long as the sensor network lifetime. So even though the use of multiple levels allows shortening of each chain, the total lifetime still has to be predefined. Since CDMs are distributed within a longer time interval, DoS attacks must be considered. Multi-level μTESLA provides two variants for this. One is the DoS *tolerant* version with a random selection method, requiring large node buffers to store multiple CDMs for each level. The other is the DoS *resistant* version, and uses a hashing technique adapted from TESLA's immediate authentication method.

3. Tradeoff Memory Data of Attack on μTESLA: We will show that if μTESLA is used with parameters that are currently widely considered to be appropriate, say 64-bit keys, TMD-tradeoff techniques can disclose future keys, only relying on realistic resources. Since a simple fix is possible we do not insist on the insecurity of μTESLA, but this shows that a simpleminded implementation can be broken not only theoretically but also in a real-world sense.

a. Target System: To keep our discussion simple, we shall fix various parameters, but no small tweaking of these parameters will make the system immune to our attack.We assume a multi-level μTESLA with 64-bit key chains created by a one-way function *F*. Adjusting our attack tosingle-level μTESLA will be straightforward. Our target system will disclose a key every 200ms with a delay of two time intervals at the lowest level and start a new chain every 1 hour.

b. Attack Objective: Readers with experience in the tradeoff technique will see that applying it to the one-way function *F* is useless, as the current key is not retrieved..So we take the non-trivial approach of having our attacker recover the key to be released in the 16th future 200ms interval (3.2 seconds later), from the key most recently disclosed. If such an undisclosed key is discovered within 200ms of obtaining the current disclosed key, from it, an attacker can generate the keys which will allow sending of authenticated messages for the duration of approximately 14 intervals (2.8 seconds).Loss of control for even a short amount of time can be devastating to the sensor network security, as the attacker may force nodes to replace all their current level keys with commitments of his choice. Once this has been done, commands from the real base station will no longer be authenticated and the attacker gains full control over the sensor network.

SECTION III

1 Attack Overview: The attacker will work over a 40-day period. Throughout this period, on each of the 200ms intervals, he will repeatedly try to see if he can recover the key that is to be disclosed 16 steps later from the current disclosed key. The choice of 16 was taken to give the attacker enough time for commitment replacement and may be adjusted to meet the attacker's needs. Let us write *H* = *F*16 to denote 16-iterated applications of the one-way function *F* used in constructing the (bottom-level) chain. Notice that if *y* is the current 64-bit disclosed key and *x* is the key to be disclosed 16 steps later, then *y* = *H*(*x*). So, the attacker wishes to find *x*,given *y* = *H*(*x*). As *H* is not injective, not all such *x* will be the correct future key, but we shall ignore this for now.

Consider the set of all keys disclosed during the 40-day period. These would consist of multiple shorter chains, each lasting one hour. After removing 15 starting3 keys from each of these shorter chains, we name the resulting set that contains

$$D := \left(\frac{1}{0.2}60\cdot 60-15\right)\cdot(24\cdot 40)\approx 2^{24.04}$$

We shall give an algorithm which processes each of the keys from ˆ*D* , over a 40-day period, and finds a pre-image under *H* for some of these. The 15 keys were removed as there are no 16-step future keys for these one-way chain beginnings. The algorithm is expected to find the *correct* future key with 64.3% probability, and can process each key within 200ms of receiving it, when run on a PC.Our choice of 40 days, which is equivalent to the choice of *D* ~ 224, and the choice of parameters *m* = 227 and *t* = 213, to appear below, may seem arbitrary. At this stage, we can only state that any choice with their product *mtD* approximately equal to the key space size 264, will work. Our specific choices were made so that the storage size, pre-computation effort *mt*, and target count *D* are all within available resources. While their true meaning cannot be understood after the algorithm and its analysis are understood, one may keep in mind that the success of the attack basically relies on the birthday paradox to produce a collision between the pre-processed *mt* keys and the *D* online keys.

2 Attack Implementation.

We run both simulation and real-world test of attacks.

1 Function Choice- Following the most commonly cited example in the related literature, our one-way function *F* was created from RC5. We need to be more explicit, as RC5 is a parameterized family of block ciphers, among which the most commonly used version utilizes 32-bit words, 12 rounds, and 128-bit keys. The 32-bit word implies 64-bit blocks and is suitable for us, but as the chain needs the key size to be of 64 bits, we took the 32-bit word, 12 round, 64-bit key version. The one-way function maps a 64-bit key to the 64-bit cipher text which is an encryption of the all-zero plaintext under the given key. The swapping of two 32-bit words constituting a 64-bit value was used as permutation *P*.

2 Hellman Table Creation- Instead of starting each Hellman chain with a random 64-bit value, we used the numbers 0 through 227 – 1 as these initial points. As these can be written down in a 32-bit space, the total Hellman table size became 1.5GB instead of the 2GB, referred to in our discussion. This allows for the Hellman table to be loaded onto a PC's 2GB memory with ample room left for the OS. In fact, we use a Cygwin Unix emulation environment on a PC in which only 1.5GB memory is allowed, and the 1.5GB table must fit into that memory without a large loss .

3 Online Phase Simulation- Random chains corresponding to 40 days were generated, with each hour starting a new chain from a new random starting point, for a total of 960 = 24 · 40 independent chains. We simulated the online phase on our Opteron system, with the target chains distributed over the 8 cores. It took 40 hours to complete, meaning 13.3 days on a single core. As our requirement of 200ms per key processing allows this to be done over a 40-day period, this is three times faster than what we would need .Using the same Hellman table, we did ten simulations with ten independently generated target data sets. Many *correct* 16-step future keys were obtained and we observed 80% probability of success. Details are given in

4 Sensor Network Application- To check the online phase in real time and to demonstrate the effectiveness of this attack, we took one of the many 1-hour chains that resulted in a correct 16-step future key and performed a test using real sensor nodes. We first construct our *μ*TESLA base station by placing the chosen 1-hour chain on a PC with dual AMD Opteron 244 (1.8GHz) processors and 4GB of memory. A

Tmote Sky (Telos rev.B) is connected to the PC through a USB port, and is forwarded *μ*TESLA messages through a Serial Forwarder (SF) using UART, which is then sent over a IEEE 802.15.4 radio channel. *μ*TESLA is implemented in C with 200ms time intervals. A Berkeley Mica-Z sensor node in which the chain commitment is installed, can verify the *μ*TESLA messages containing data and keys. We have the sensor node blink its yellow LED for verified data messages and its green LED for key disclosure messages. The Hellman table is placed on another PC to act as the attacker. Two Tmote Skys are connected to the attack PC through USB ports, so as to listen to the base station and send out forged messages. An attack program implemented in C communicates with the *Listener* and the *Sender* through UARTs, and sends out forged commands making the sensor node blink its red LED. Through our attack experiment, we were able to visually check the red LED flashing. This is the result of the attacker's forged messages, created using 14 valid keys, each corresponding to one interval.

4 Tradeoff Attack Analysis

This section will give a brief idea on analysis of attack algorithm is somewhat technical and may be skipped by anyone that can believe that our attack succeeds with a reasonable probability and that it can be applied to most modifications of our explicit attack target.

1 Attack Success Probability- Let us see what probability of success we can expect from our attack. We start with a small lemma, whose proof is elementary. Consider a set *N* of size *N*. Randomly choose and fix *D* distinct elements from *N* and name the set *D*. Next, randomly choose *H* elements from *N*, one at a time, with replacements, and call the collection *H*. The family *H* may contain overlapping elements. *Lemma 1:* Assuming *D*<<< *N* and *DH* ¬ *N*, the probability of *D* and *H* containing at least one element in common can be approximated by

$$P \sim 1-\exp\left\{-\frac{DH}{N}\right\}$$

Let us apply Lemma 1 to our attack setting. The base space *N* will be the set of all possible 64-bit keys, so that *N* = 264.

Next, consider the set of all online keys disclosed during the 40-day period. These would consist of multiple shorter chains, each lasting one hour. We remove 15 *ending* keys6 from each of these shorter chains and take the resulting set as ˇD .7 The number of elements D in ˇD is given by equation (1), as before. These D elements may be assumed to be distinct, for, if otherwise, the key chain would repeat itself and the authentication system would fall under a more trivial attack. Take ˇH to be the family of keys appearing as input points to mapping ˜H applied during creation of T . This excludes the ending points of each Hellman chain, and refers to H = $t{\cdot}m$ = 240, possibly overlapping, elements.8 Now, a careful review of Algorithm 2 will reveal that should there be any element common to ˇD and ˇH , it will be returned by Algorithm 2. This common element x €˜ˇD maps to the disclosed key $H(x)$ €ˆD and implies success of attack. The success probability of our attack can be calculated as p ^ 1 − exp(−1.029) ^ 0.643, by substituting various numbers into Lemma 1.

2 Parameter Tweaks-

In this subsection, consider the application of our tradeoff attack to other sensor network configurations.

a Shorter Disclosure Interval: Suppose the sensor network uses key disclosure interval shorter than the 200ms we have considered. This would result in a larger online target set being available to the attacker for the same (40- day) period of attack. This allows the success probability of attack to be maintained with a shorter Hellman chain. Hence the attacker can cope with the shorter time interval allotted to processing of each key. There may still seem to be one problem, as the attacker recovers the 16-

Interval future key and this is closer in real time than before. But a faster disclosure interval would usually mean a faster radio network, and hence the attacker would be satisfied with the shorter time available for trying out of the recovered key. Another approach the attacker may take is to attempt to recover keys further steps into the future. This would require longer precomputation time for the same length Hellman chains and a more powerful system during the online phase. By a more powerful system, we mean that one could either use a faster processor, or let multiple processors take turns processing the target data, each for a time span longer than the disclosure interval.

b Longer Disclosure Interval: If the opposite approach of using longer disclosure interval is used, the attacker has less online target data available than before. But this gives him more time to process each target data, so longer Hellman chains can be used. This will result in the pre-processing time increasing, but an increase by a small factor is well within current computational power. The attacker can also take the approach of trying to recover keys smaller steps into the future. Then the longer Hellman chains will not take longer to create.

c. summary: The tradeoff attack technique has been known for a long time, and this raises the question as to why delayed exposure of 64-bit one-way key chains had widely been accepted as a plausible authentication method. Note that for a straightforward application of the original Hellman method or the more widely known

rainbow table method , a pre-computation phase consisting of about 264 calculations of the one-way function is required. While no one can say for sure that this is currently impossible, it does seem to be out of reach for most organizations. Coupled with the resource constrained environment, these 64-bit one-way chain methods seem acceptable at first sight. But the Hellman method and rainbow table method deal with only a *single* target data. Our approach of trying multiple times over an extended period and being content with succeeding just once seems to have been overlooked.

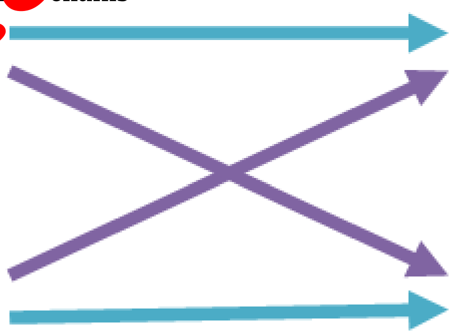
The multiple target version of tradeoff attack technique we have used in this paper, applicable to any one-way function, is not new and has been developed in But until it was made explicit by the recent work , many took this to be applicable to only stream ciphers in a particular way. The main contribution of this paper concerning the weakness of current μTESLA is of pointing out that *multiple* target version of pre-computation attack is naturally applicable to the one-way chains. In doing this, the idea

of looking into a 16-step composition of what would usually have been taken as the one-way function of interest was crucial. As long as succeeding even once within an extended time period is a realistic threat, there seems to be no way of using 64-bit one-way chains without *salting* them, that is, even on low-security applications.

SECTION IV

4.1. Broadcast Protocol for Secure X-TESLA: The basic idea starts from the extendable management of short key chains. In essence, we make two levels of chains having distinct time intervals cross-authenticate each other to provide permanently extendable chains. Our protocol X-TESLA, read either as TESLA or cross TESLA, stands for extendable TESLA. As with other TESLA variants, X-TESLA provides broadcast authentication, under the assumption that the base station and sensor nodes are loosely time synchronized with a known maximum synchronization discrepancy.

Key chains



Key chains

Fig: The crossing of illustrates the followings.

(a) The lower level chain naturally authenticates the next upper level chain, as they are connected in a single chain by construction.

(b) Multiple *distinct* keys in the upper level chain authenticate the initial commitment of the next lower level chain repeatedly. The repeated authentication will help in resolving problems from DoS attacks, sleeping nodes, and idle sessions.

4.2 Basic Framework of X-TESLA

4.2.1 X-TESLA chains: Two functions $F0(\cdot,\cdot)$ and $F1(\cdot,\cdot)$, mapping $K{\times}S$ to K will be used. Here, K denotes the key space, and S is the *salt* space. For each fixed $s \in S$ we expect the operator $Fi(\cdot, s)$ on K to be one-way, even when s is known. In practice, we design the two functions with 64-bit blockciphers taking 64-bit keys and salt as plaintext in Section 4.4. The two functions may even be instantiated with the same blockcipher. Let us divide time into intervals with indices u, v_ and u, v,w_ used for the upper and lower levels, respectively. Let u index both level chains having the same durations for $u > 0$, $v$do intervals of each upper level chain for $0 < v \le n$, and w divide those intervals minutely for a corresponding lower level chain for $0 < w \le m$. Intervals themselves will be denoted as Iu,v and $Iw\ u,v$. We let Ju,v and $Kw\ u,v$ denote the corresponding upper and lower level keys. When $v = 0$ or $w = 0$, an indexed key is a commitment.

One of distinctive features of X-TESLA is the use of salt values denoted by Su,v and Swu,v, whose choice we defer to Section 4.4. These will remove TMD-tradeoff concerns by making pre-computation infeasible. After fixing each salt value, we define the upper level chain for each positive integer $u > 0$, by starting from a random seed key Ju,n € K and recursively setting

4.2.2 Communication Packets: For the framework of Tiny OS, we design communication packets to fit within its 29-byte default payload size. It is trivial to allow larger packets if necessary. As depicted, we define four types of packets that use the first byte of data payload for type distinction and the following four bytes for an index. Type 1 is an authenticated data packet of which 16 bytes are used for data transmission and the remaining 8 bytes are used for MAC generated by a lower level key. Type 2 is another form of authenticated data packet of which only 8 bytes are used for data transmission with an 8-byte MAC ,while the remaining 8 bytes are used for key disclosure of a previous lower level interval. Type 3 is designed to handle sleeping nodes and idle sessions. It is the same with Type 2 except that the 8-byte data is a future lower level key masked with a future upper level key Data payload in packets of X-TESLA.The masked key is authenticated soon but unmasked much later. Of course, Type 2 and Type 3 can *trivially* be merged up to a single type of slightly larger packet. Type 4 packets hold a future lower level commitment at the data portion with a MAC calculated from an *upper* level key. Notice that the same lower level commitment is sent throughout a whole upper level chain. The AUX header field and the structure of CCM encryption mode of ZigBee packets may be of some use for making more efficient variants of the packet types.

4.3 X-TESLA Details

4.3.1 Initialization: We assume a base station broadcasts authenticated messages to sensor nodes. A method to choose salt values is fixed at system design phase. The base station generates the first upper level chain by choosing seed key $J1,n$ € K at random and also generates the first lower level chain together with the second upper level chain by choosing another seed key $J2,n$ € K randomly. The values $J1,0 = F1(J1,1,\ S1,1)$ and $K0\ 1,1$ are stored in each sensor node as initial upper and lower level commitments, respectively. Depending on the way salt is chosen, some extra information may also need to be stored. It would be advisable to keep these values secret until just before deployment. Generation of the second lower level chain together with the third upper level key chain should soon follow, so as to be ready for commitment distribution. When the initialized nodes are deployed, they are to be loosely time synchronized with the base station, as assumed in μTESLA.

4.3.2 Broadcast Authentication: During an $Iw\ u,v$, the base station uses $Kw\ u,v$ as the MAC key for Types 1, 2, and 3 packets being sent out, and reveals Kwu,v after a wait of time δ from the end of Iwu,v,in Type 2 or 3 packets. We shall abuse interval indices, setting $Iu,n+1 = u+1,1$, $Im+1\ u,v = I1\ u,v+1$, $Iu,0 = Iu-1,n$,and $I0\ u,v = Im\ u,v-1$. The following is a Type 2 packet for use with "δ = *one* time interval." Here, / denotes concatenation and ∗ signifies the index and data portion.

T2Pw u,v =(*u, v,w)!!*data!!MAC!!*Kw u,v* (∗)!!*Kw*−2

This corresponds to what is usually stated as key disclosure delay of *two* time intervals.If the key disclosure message is lost, the sensor node buffers all messages it receives until a key disclosure message is successfully received, and computes.

4.3.3 Commitment Hopping: With TESLA variants, there are at least two situations in which verification of a newly disclosed key places heavy computational load on a sensor node, resulting in many message drops, for the duration of this computation. First, if a sensor node falls into sleep mode or turns off its radio power to save energy, it may not be able to listen to the key disclosure messages during that period. Second, if there are long idle periods with no broadcast, it would be wasteful to disclose keys on schedule and a base station might minimize the key disclosures for those periods. As a result, there could be a large gap between the current commitment and the key to be verified. Type 3 packets can resolve this problem, by providing *commitment hopping*. Let $Iu\_,v\_$ be an interval appearing after Iu,v. The distance9 between the two intervals depends on the application needs. We set *T*3

Pwu,v = *u, v,w*!!*Kw u,v* ⊕*Ju_,v*!!MAC*Kwu,v*(∗)!!

Kw−2*u,v* .*w v* .

4.3.4 Cross Authentication: With X-TESLA, keys of the upper level chain can be authenticated by the previous lower level chain since they are connected in a single chain by construction and since the latest commitment key of the previous lower level is available to sensor nodes. Type 3 packets further help in making this available. After any verification, the

Commitment for the upper level can be updated. For authentication of a new lower level chain, the upper level chain is used. The following is a Type 4 packet. It distributes the commitment of the next lower level chain while disclosing a previous upper level key.*T*4

Pu,v = (*u, v)!!K*0*u*+1,1!!MAC*Ju,v* (∗)!!*Ju,v*−1The next lower level commitment *K*0*u* +1,1 is distributed *at random instances* within *Iu,v*, authenticated with *Ju,v*. In fact, many (different) Type 4 packets are constructed and broadcast to deliver the same next lower level commitment *K*0*u*+1,1 during *Iu*. Therefore, a sensor node would have numerous chances to receive a correct *K*0*u*+1,1 during *Iu*, and resist DoS attacks without the use buffers of multi-level μTESLA. A node buffers a single or slightly .The offset should be reasonably set. Multiple offsets may be used by assigning different message types to handle distinct offsets.

4.3.5 Flexible Constructions: We now place more flexibility, in addition to the choice of chain lengths, into the X-TESLA construction. This will resolve even the most extreme situation that could occur with Type 4 packets. Starting from the basic flow we extend the upper level chain over a number of lower level chains for better survivability against high communication faults and long idle sessions depicted in. Even a short extension of the upper level chain with only small bits allows many lower level chains to be attached, and these may be generated on the fly. The extension increases stability of chain verification in both levels. The change also provides longer periods in which to distribute the next chain commitments for both levels through Type 3 and Type 4 packet

variants.The reverse flow allows reduction of Type 4 packets for environments in which authenticated messages are broadcast very frequently. Since an upper level chain serves as commitments for the next lower level chain, Type 4 packets distribute $Ju+1,0$:=10. Within each Iu,v, a random selection process (which might come naturally from the environment) can also be employed. But the dependance on Type 4 packets is smaller, because the upper level keys can be recovered stably from Type 3 packets if the authenticated broadcasts of the lower level
are very frequent. The hybrid flow offers *extreme durability*. For every $u \equiv 1 \pmod 3$, a lower chain of $Iu+3$ is generated from a random seed with upper chains of $Iu+2$, $Iu+1$, and Iu, together with lower chains of $Iu+1$ and $Iu-1$ descending from this.

4.3.6 Sleep Mode Management: Energy efficiency is mandatory for sensor networks since tiny nodes are operated on batteries. Various types of sleep modes11 that stop CPUs or radio functions are commonly used but care must be taken, as nodes that have been inactive for a long time may need to do much computation for key verification or lose commitment. Let Tu denote the starting time of interval Iu, and set $\Phi = Tu+1-Tu$ to the length of one upper level chain. sensor node shall not be allowed to go into a long term sleep or, at the least, not be allowed to stop radio functions for a long term period unless it has obtained the next lower level commitment, while short term sleeps are always allowed. More specifically, we fix some threshold value θ that takes the clock discrepancy of nodes into account, and for a node that has verified a Type 4 packet at time T, we allow it to set the maximum sleep length timer() to the duration of up to Φ only if $T < Tu + \theta$ (as in node A of Fig. 4), and to the duration of up to $Tu+1 + \theta - T$ if otherwise . These values are meant to be the maximum sleeping length, and a sensor node may repeat going to sleep and waking up freely (or stopping and awaking radio components) within the given duration. The value θ should be fixed so that the security parameter $\varepsilon = P[\Phi] - P[\Phi - \theta]$ is kept appropriately small, where $P[\Phi]$ and $P[\Phi - \theta]$ denote the probabilities for a node to receive and verify a Type 4 packet within respective time lengths. Note that Φ is quite long, amounting to 3.6 hours if, for example, 216-key chains of 200ms intervals are used for the lower level. When a sensor node finally awakes, it should be allowed to go back to sleep for a long period only if it receives and verifies the next lower level commitment. If a node fails to verify any Type 4 packet in some Iu, it should be made to try harder in the next interval $Iu+1$, for example, by sleeping less, but often Type 3 packets could already have provided the lower level commitment of $Iu+1$. The sleep mode management system explained here should make the extendable property of X-TESLA work stably .Still, the upper level length in X-TESLA needs to be Chosen carefully, so that unexpected length of communication failure does not completely disrupt the system. Though this makes parameter selection challenging, the lengthening of the upper level is relatively cheap, and the use of flexible construction of Fig. 3 is also possible.

4.4 Implementation of X-TESLA

4.4.1Practical Construction: We use 28-key chains for the upper level and 216-key chains for the lower level with 200ms intervals but various other combinations are also possible. The broadcast module is implemented by connecting Tmote Sky to a PC, and the receiving module is ported into Mica-Z, with 17KB of program memory of which 9KB are occupied by system. We set 28-byte12 payloads. we utilize the 64-bit key version of RC5 for generating chains. The salt values, used as plaintext, should be known to the verifying node as well, and can be defined in various ways. Taking a practical approach, we use Su,v = $u, v\_$ and $Sw\ u,v = u, v,w\_$ in the implementation, where the indices are zero-extended to fill the 64-bit block size, with the exception of the most significant bit, which is used to differentiate $F0$ from $F1$.We caution that this is *not a complete solution* against TMD-tradeoff attacks. If the indices are short, so that index repetition is common, an attacker may decide to focus on (multiple) target points corresponding to one fixed index. Even if index is long enough not to repeat itself within the lifetime of the network, a tradeoff attack on a single target would still be possible. This is not an immediate threat with average-powered attackers, but probably not so for long with 64-bit chains. Rather we propose a more robust solution to combine old (disclosed) key with the index to produce salt.. Thus, a sort of randomness, unpredictable until near the time of use, could be employed, so as to prevent pre-computation.

4.4.2 Running X-TESLA: To start with, we need a 28-key chain for the upper level and a (216 + 28)-key chain for the lower level with its source, which is the next upper level. For commitments soon to be distributed, one additional future chain must be prepared. As a result, the base station maintains three upper level and two lower level chains at run time. It takes only 797ms to compute these chains on a PC with dual AMD Opteron 244 (1.8GHz) CPU and only 1MB to store them. In our test implementation, for simplicity, we preset the starting time and had the base station send out a synchronization command. This is acceptable, as the initial deployment phase is usually assumed to be secure in the literature. More sophisticated synchronization methods can be found .The number of unauthenticated packets buffered by a sensor node depends on the period and reliability of key disclosure messages. Concerning the key disclosure interval, note that a 36-byte TinyOS packet, consisting of 5-byte header, 29-byte data payload, and 2-byte CRC tailer, takes 28.8ms to send on a 10kbps radio network, with round-trip taking less than 60ms. Similarly, a 39-byte ZigBee packet in which 29 bytes are data payload,takes 5.1ms on average to send on a 60kbps13 radio network, with round-trip taking less than 15ms. So any key disclosure interval larger than 50ms is possible. In case the shorter 50ms intervals are used, it might be preferable to use slightly longer chains, to preserve the duration covered by a single lower level chain.

SECTION V

5.1. Security Issues: As was stated in Section 4.2.1, X-TESLA protects against TMD-tradeoff attacks through the explicit use of salt, so that even the 64-bit key chains can be practically secure .The extendable management of short chains leads to security advantages as well as efficiency advantages. In (multi-level) μTESLA, the lifetime of the sensor network is pre-determined and a chain (or at least one chain) that spans throughout this very long period is used. This means that the seed key (and far future keys) should be protected very securely, for were it to be compromised without the base station being aware, it could. Be troublesome for a very long period. So, depending on the adversary model, X-TESLA, which uses shortlived chains, will have security advantages. In any case, using long chains is less than ideal, as function iteration continually reduces the entropy of key space.

5.2. DoS Attack Resistance: Communication faults and DoS attacks may result in ***packet loss* or *forged*** packets. To overcome these problems, a base station could repeat a packet for a reasonable number of times. For example, if a packet loss rate is 30%, the probability of receiving can be increased to 99.2% by repeating the packet just four times. Since the time interval of lower level chains is tiny and the verification key is disclosed shortly, forged messages can be deterred by an affordable buffering in the lower level. Compared to the other types, Type 4 packets could be less resistant to DoS attacks because they have to be buffered until verification for the duration of the longer upper interval. By jamming a whole interval15 *Iu,v*, a DoS attacker can drop all Type 4 packets from that interval, but fortunately the impact diminishes rapidly as the attacker loses domination, especially when considered over all of *Iu*. Let *pl* be the packet loss rate of sensor nodes due to communication faults and sleep modes and set ¯*pl* = 1 − *pl*. Suppose that the base station randomly chooses *r* of the *m* intervals within each *Iu,v* to broadcast Type 4 packets and that the attacker dominates *k* intervals within each *Iu,v*. Since the time interval of upper chains is relatively long, the attacker could try to overflow sensor node buffers with forged Type 4 packets after listening to the correct key *Ju,v*−1 disclosed in that interval. However, it is sufficient with X-TESLA that each node buffers only a single (or slightly more) Type 4 packet received in each interval *Iu,v* for verifying *K*0*u* +1,1 within *Iu*. Among the four constructions of X-TESLA , the basic method delivers the next lower level chain commitment through Type 4 packets, but repeats it within *Iu*, so that a node can receive a valid one with very high probability. The other three constructions allow even better probability. Consequently, X-TESLA resists DoS attacks of forged packets intrinsically, whereas multilevel μTESLA necessitates a large buffer and much precomputation with storage for its CDM packets. We could observe at least such a big difference between them.

5.3 Efficiency Comparison: While multi-level μTESLA and X-TESLA provide comparable resistance against DoS attacks, in this section, we show that the required resources are different.

5.3.1 Computation and Storage for Base Stations: With X-TESLA, only a small number of short chains need to be stored in the base station, with the rest computed on the fly, and the chains are extendable indefinitely. In μTESLA and DoS resistant multi-level μTESLA, full chains covering all of the expected lifetime (and their CDMs in multi-level μTESLA) have to be pre-computed and stored. By storing the pre-computed chain only in part, storage can be reduced, but at the cost of online recomputation.

5.3.2 Storage for Sensor Nodes: To verify a message, a sensor node has to buffer the index, data, and MAC fields until the delayed exposure of the corresponding key. This is a shared property of all μTESLA variants. X-TESLA shares another property with multi-level μTESLA in that new commitments for future chains need to be buffered and verified, but XTESLA requires less storage in the nodes than multilevel μTESLA for three reasons. First, only two levels are used in X-TESLA, while more levels (or longer chains) are necessary in multi-level μTESLA. Second, X-TESLA verifies an upper level commitment, which is masked for later use, almost immediately after reception, following the shorter lower level schedule, but with multi-level μTESLA, verification of an level-_ commitment must wait through level-(_+1)'s longer interval, and this situation worsens as we go up the levels. Third, verification of lower level commitment in X-TESLA follows the upper level interval schedule, but without large buffering. With X-TE SLA, a sensor node stores one most recently authenticated key as the current commitment for each level, along with the next lower level commitment, and possibly the masked key from a recent Type 3 packet, adding up to a total of four keys (taking 32 bytes) at runtime. In comparison, an *M*-level μTESLA node stores 3*M*−2 keys (taking *more* bytes), along with the buffered CDMs for each level except the highest level. Let us now look at the node storage required to handle Type 4 or CDM packets reliably, by comparing an X-TESLA of 28/216-key upper/lower chains with a 2-level μTESLA of 212/216-key chains and a 4-level of 24/28/28/28-key chains within the 228-key lifetime. Let *r* and *f* denote the number of real and forged Type 4/CDM packets appearing in a single *Iu,v* or lowest level 28-key interval.

5.3.3 Computation and Communication for Sensor Nodes: In sensor networks, power consumption of sensor nodes is one of the most significant issues since sensor nodes are usually operated on batteries. With μTESLA variants, sensor nodes may consume energy while computing chains (for verification), computing MAC, and receiving broadcast packets. We analyze computation and communication costs of sensor nodes from this perspective. Let a random process *X*(*t*) have an exponential distribution and let *E*[*X*] be the expected value that is the average distance between two packet arrivals, with regard to a message rate. We take Δ*t* to be the time interval for which a single lowest level key is valid.

CONCLUSION

Through the application of TMD-tradeoff techniques we observed that care should be taken with the shortkey chain based broadcast authentication schemes. We have proposed X-TESLA, an efficient scheme which may continue indefinitely and securely, that addresses this and many other issues of the previous schemes. With the advent of more powerful sensor node commodities such as iMote2 [14], the future of public-key technique application to broadcast authentication looks bright, but X-TESLA can efficiently be combined with public-key techniques also. For example, we could modify X-TESLA to use digital signatures on Type 4 packets, keeping everything else the same.

REFERENCES

[1]I.F.Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," IEEE Comm. Magazine, Vol. 40, No.8, pp. 102–114, Aug. 2002.

[2] G. Avoine, P. Junod, and P. Oechslin, "Time-Memory Trade-offs:False Alarm Detection Using Checkpoints," Indocrypt 2005, LNCS
3797, pp. 183–196, Springer-Verlag, 2005.

[3] A. Durresi, V. Paruchuri, S. Iyengar, and R. Kannan, "Optimized Broadcast Protocol for Sensor Networks," IEEE Trans. Computers,vol. 54, no. 8, pp. 1013–1024, Aug. 2005.

[4] S. Ganeriwal, S. Capkun, C. Han, and M. Srivastava, "Secure Time Synchronization Service for Sensor Networks," Proc. ACM Workshop on Wireless Security (WiSe), pp. 97–106, 2005.

[5] Y. Hu, M. Jakobson, and A. Perrig, "Efficient Constructions for One-way Hash Chains," Proc. ACNS 05, LNCS 3531, pp. 423–441,Springer-Verlag,2005

[6] Intel IMote2 Overview, http://www.intel.com/research/ downloads/imote_overview.pdf, 2005. Commercialized by Crossbow, INC. http://www.xbow.com/.

[7] M. Karaata and M. Gouda, "A Stabilizing Deactivation/Reactivation Protocol," IEEE Trans. Computers, vol.56, no. 7, pp. 881–888, Jul. 2007.

[8] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," IEEE Trans. Computers, vol. 55, no. 2, pp. 214–226, Feb. 2006.

[9] D. Liu, P. Ning, "Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks," Proc. ISOC Network and Distributed System Security Symposium (NDSS), pp. 263–276, Feb. 2003.

[10] D. Liu and P. Ning, "Multi-Level μTESLA: Broadcast Authentication for Distributed Sensor Networks," ACM Trans. Embedded Computing Systems, Vol. 3, No. 4, pp. 800–836, Nov. 2004.

[11] M. Luk, A. Perrig, and B. Willock, "Seven Cardinal Properties of Sensor Network Broadcast Authentication," Proc. ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), October 2006.

N.Sainath B.Tech from JayaPrakash Narayana College of Engineering M.Tech SE from Srinidhi Institute of Technology. Currently he is working as Associate Professor at Sree Vishweshawarya college of Engg. His areas of interest include Data mining, Network Security.

N. Vikram Narayanadas M.Tech from Shadan Engineering College B.Tech from Sree Datta College of Engg. Currently he is working as Associate Professor at JayaPrakash Narayana College of Engg. His areas of interest include Information Security, Databases.

Secure and Efficient Broadcast Authentication in Wireless Sensor Networks

Taekyoung Kwon, *Member*, *IEEE*, and Jin Hong

Abstract—Authenticated broadcast, enabling a base station to send commands and requests to low-powered sensor nodes in an authentic manner, is one of the core challenges for securing wireless sensor networks. μTESLA and its multilevel variants based on delayed exposure of one-way chains are well known valuable broadcast authentication schemes, but concerns still remain for their practical application. To use these schemes on resource-limited sensor nodes, a 64-bit key chain is desirable for efficiency, but care must be taken. We will first show, by both theoretical analysis and rigorous experiments on real sensor nodes, that if μTESLA is implemented in a raw form with 64-bit key chains, some of the future keys can be discovered through time-memory-data-tradeoff techniques. We will then present an extendable broadcast authentication scheme called X-TESLA, as a new member of the TESLA family, to remedy the fact that previous schemes do not consider problems arising from sleep modes, network failures, idle sessions, as well as the time-memory-data tradeoff risk, and to reduce their high cost of countering DoS attacks. In X-TESLA, two levels of chains that have distinct intervals and cross-authenticate each other are used. This allows the short key chains to continue indefinitely and makes new interesting strategies and management methods possible, significantly reducing unnecessary computation and buffer occupation, and leads to efficient solutions to the raised problems.

Index Terms—Security, broadcast authentication, time-memory-data tradeoff, wireless sensor networks.

1 Introduction

Technological advancement in large-scale distributed networking and small sensor devices has led to the development of wireless sensor networks with numerous applications [1]. Sensor nodes are usually constrained in their computation, communication, storage, and energy resources for economical reasons, but need security functions since they are deployed in unattended or even hostile environments. The high risk of physical attacks and the limited capabilities of sensor nodes make it difficult to apply traditional security techniques to wireless sensor networks, posing new challenges [29].

Authenticated broadcast, enabling a base station to send authentic messages to multiple sensor nodes, is one of the core challenges [20], while even the broadcast by nodes is an important topic in wireless sensor networks [7], [21], [25]. For the purpose, digital signatures (public key) are not very useful in a resource-limited environment, while naïve use of HMAC (secret key) does not work either, as node capture can lead to a key compromise. μTESLA and its multilevel variants [18], [19] based on TESLA [26], [27], use a one-way chain practically [13] under a loose time synchronization assumption. The sender attaches a Message Authentication Code (MAC) to each packet, computed using a key from the chain in reverse order. The keys are exposed after a certain time delay. The receiver buffers the received packet until the corresponding key is disclosed and verifies the MAC, after authenticity of the key itself has been verified by following through the chain.

- *T. Kwon is with the Department of Computer Engineering, Sejong University, Seoul 143-747, Korea. E-mail: tkwon@sejong.ac.kr.*
- *J. Hong is with the Department of Mathematical Sciences and ISaC, Seoul National University, Seoul 151-747, Korea. E-mail: jinhong@snu.ac.kr.*

Manuscript received 21 Feb. 2009; revised 22 Aug. 2009; accepted 6 Oct. 2009; published online 29 Oct. 2009.
Recommended for acceptance by S. Nikoletseas.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2009-02-0080.
Digital Object Identifier no. 10.1109/TC.2009.171.

1.1 Motivation (and Problems)

μTESLA and its variants are designed to be practical, but significant concerns still remain.

1.1.1 64-Bit Key Chain

A short 64-bit key chain is desirable for efficiency in resource-limited sensor nodes, but care must be taken, even with short time intervals. As we show, if the chain is generated in a straightforward manner, Time-Memory-Data (TMD) tradeoff techniques can be applicable, leading to discovery of future keys.

1.1.2 Sleep Mode or Network Failure

If sensor nodes go into a sleep mode or key disclosure messages are lost frequently, μTESLA may force heavy key computation to be done at once on sensor nodes for chain verification, during which incoming packets get dropped. If Commitment Distribution Messages (CDMs) are missing, multilevel μTESLA makes nodes wait and buffer for the long interval of upper levels, during which incoming packets are dropped due to the buffer limit.

1.1.3 Idle Sessions

Even for idle sessions with no broadcasts, μTESLA forces chain computation for sensor nodes. Key disclosure messages should be broadcast constantly or heavy computation needs to be done later. Multilevel μTESLA needs CDMs to be broadcast for higher levels, with the number of CDMs increasing with the number of levels.

1.1.4 Extended Lifetime

With node malfunctions and premature power exhaustion, there are needs for node additions [1] or rechargeable sensor nodes [15]. Thus, the lifetime of a network may