THE EFFECT OF TIME DIMENSION AND NETWORK DYNAMICS ON KEY DISTRIBUTION IN WIRELESS SENSOR NETWORKS

by

ÖMER ZEKVAN YILMAZ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
March 2009

THE EFFECT OF TIME DIMENSION AND NETWORK DYNAMICS ON KEY DISTRIBUTION IN WIRELESS SENSOR NETWORKS

APPROVED BY

Assoc. Prof. Albert Levi .....
(Thesis Supervisor)

Assist. Prof. Hüsnü Yenigün .....

Assoc. Prof. Özgür Gürbüz .....

Assist. Prof. Thomas Brochmann Pedersen .....

Assoc. Prof. Tuna Tuğcu .....

DATE OF APPROVAL: .....

to my family

&

sensor networks community

Acknowledgments

I have spent 7,5 years in Sabancı University for bachelor and master degrees. If this study is coming to end then it is due to Sabancı University family with professors, personnel, students and others who are involved. Besides this, my beloved family and dearest friends, whose support was inevitable in this period, deserve to be mentioned.

In this long run, my thesis advisor Assoc. Prof. Albert Levi had much of the role both in curriculum courses and thesis. I thank him for his valuable effort and motivation. Assoc. Prof. Erkay Savaş has also a special place that made us love cryptography. I thank all computer science and in general all faculty of engineering professors who are the reasons why I have been in Sabancı University this long. I am thankful to my thesis defense committee members: Assist. Prof. Hüsnü Yenigün, Assoc. Prof. Özgür Erçetin, Assoc. Prof. Özgür Gürbüz, Assist. Prof. Thomas Brochmann Pedersen and Assoc. Prof. Tuna Tuğcu for their support and presence.

During my master study, I was supported by scholarships of Sabancı University and TÜBİTAK. I m grateful to these foundations for enabling the education of many young people like me. At the same time, TÜBİTAK is the place where I have spent the last 6 months of this study as an employee. During this time, the support of this institution to academic development proved well that Turkish academic community and Turkish R&D owes much to TÜBİTAK. I appreciate the support of my peers both in Sabancı University and TÜBİTAK and wish the best of success in their studies.

THE EFFECT OF TIME DIMENSION AND NETWORK DYNAMICS ON KEY DISTRIBUTION IN WIRELESS SENSOR NETWORKS

Ömer Zekvan Yilmaz

Abstract

The majority of studies on security in resource limited wireless sensor networks (WSN) focus on finding an efficient balance among energy consumption, computational speed and memory usage. Besides these resources, *time*, *network dynamics* (e.g. routing), and *implementation and integration issues* of the security solutions are relatively immature aspects that can be considered in system design and performance evaluations. In the first part of this thesis, we develop and analyze different implementation options of a Random Key Predistribution scheme in a real network simulation environment. Implementation options include *Proactive Key Establishment* and *Reactive Key Establishment*. In *Proactive Key Establishment*, pairwise keys are established at the beginning, prior to start of application. In *Reactive Key Establishment*, keys are established only whenever needed by the application during its execution. In literature the latter is known to preserve energy since it reduces useless key establishments; however, it also introduces delay in application traffic. We implement the reactive key establishment in such a way that key establishment traffic and energy consumption are reduced. As a result our reactive key establishment implementation has similar throughput performance with proactive scenarios despite the longer lifetime of reactive scenario. We also simulate an attack scenario and measure different metrics including a novel one. This new metric, the packet compromise ratio, reflects the harm caused by the adversary in a more realistic way. In our simulations, we show that packet compromise ratios are very high as compared to link compromise ratios for a long period. However, when the majority of nodes die, link compromise ratios exceed packet compromise ratios. This is an indication to the fact that link compromise ratios seem high even though there is no high amount of traffic in network to be compromised by adversary.

Due to the results showing that classical key distribution schemes in WSNs have

actually low resiliency, in the second part of this thesis, we propose new deployment models that improve resiliency. In a recent study by Castelluccia and Spognardi, the time dimension is used to lower the ratio of compromised links, thus, improving resiliency in key distribution in WSNs. This is achieved by making the old and possibly compromised keys useful only for a limited amount of time. In this way, the effect of compromised keys diminishes in time, so the WSN *selfheals*. We further manipulate the time dimension and propose a deployment model that speeds up the resiliency improvement process with a tradeoff between connectivity and resiliency. In our method, *self healing* speeds up by introducing nodes that belong to future generations in the time scale. In this way, the duration that the adversary can make use of compromised keys becomes smaller.

ZAMAN BOYUTU VE AĞ DİNAMİKLERİNİN KABLOSUZ DUYARGA AĞLARINDA ANAHTAR DAĞITIMINA ETKİSİ

Ömer Zekvan Yilmaz

Özet

Sınırlı kaynaklara sahip olan Kablosuz Duyarga Ağları(KDA) konusunda çalışmalar çoğunlukla enerji tüketimi, işlem hızı ve hafıza kullanımı arasında verimli olacak bir denge üzerinde yoğunlaşmaktadır. Bu kaynakların yanında, *zaman*, *ağ dinamikleri* (rn. yönlendirme) ve güvenlik çözümlerinin *gerçekleme ve uyarlama detayları* sistem tasarımı ve performans değerlendirmelerinde gözetilmesi gereken yeterince işlenmemiş konulardır. Bu çalışmanın ilk bölümünde, bir Rastgele Anahtar Önyükleme şemasının farklı gerçekleme seçeneklerini gerçek ağ benzetim ortamında geliştirip analiz ediyoruz. Gerçekleme seçenekleri *Proaktif Anahtar Kurulumu*(PAK) ve *Reaktif Anahtar Kurulumu* (RAK) modellerini kapsamaktadır. *PAK*'ta, ikili anahtar kurulumları uygulamanın başlamasından önce yapılmaktadır. *RAK*'ta ise, anahtarlar uygulamanın çalışması sırasında gerektikçe oluşturulmaktadır. Literatürde, RAK fayda sağlamayacak anahtar kurulumlarını azaltarak ağ bazında daha az enerji harcaması ile bilinmektedir. Ancak, uygulama trafiğinin gecikmesine de neden olmaktadır. Çalışmamızda RAK modelini, anahtar kurulum trafiğini ve dolayısıyla düğüm başına enerji harcamasını azaltacak şekilde gerçekliyoruz. Sonuç olarak, tasarladığımız RAK gerçeklemesi daha uzun ağ ömrüne sahip olmasına rağmen PAK ile benzer uygulama verimine ulaşmaktadır. Aynı zamanda, bir saldırı senaryosu benzetimi uygulamakta ve çeşitli performans kriterleri kullanmaktayız. Bunların içinde yeni bir kriter olan paket ele geçirilme oranı, saldırgan tarafından verilen zararı daha gerçekçi bir şekilde yansıtmaktadır. Yaptığımız benzetimlerde, paket ele geçirilme oranının bağlantı ele geçirilme oranından uzun bir süre için çok daha yüksek olduğunu gösteriyoruz. Ancak, düğümlerin çoğunluğu öldüğünde, bağlantı ele geçirilme oranı paket ele geçirilme oranını geçmektedir. Bu durum, bağlantı ele geçirilme oranının yüksek gözükmesine rağmen saldırganın ele geçireceği yüksek miktarda trafik olmadığını işaret etmektedir.

Klasik anahtar dağıtım yöntemlerinin gerçekte düşük dayanıklılığa sahip olduğunu gösteren sonuçlar doğrultusunda, tezin ikinci bölümünde, dayanıklılık artışı sağlayan yeni dağıtım modelleri önermekteyiz. Castelluccia ve Spognardi'nin yaptığı yeni bir çalışmada zaman boyutu, ele geçirilen bağlantı oranının düşürülmesinde kullanılmıştır. Böylece anahtar dağıtımının dayanıklılığı artırılmıştır. Bu sonuca, eski ve muhtemelen ele geçirilmiş olan anahtarlara yalnızca kısa süreliğine kullanım izni verilerek ulaşılmıştır. Bu şekilde, ele geçirilen anahtarların etkisi zaman içinde azalmakta ve KDA *öz iyileşme* sağlamaktadır. zaman boyutunun etkisini artırarak yerel bağlantı ile ödünleşim karşılığında dayanıklılık iyileştirme sürecini hızlandıran yeni bir dağıtım modeli önermekteyiz. Yöntemimizde, zaman çizgisi üzerinde daha ileriye ait düğümlerin kullanılmasıyla *öz iyileşme* hızlandırılmıştır. Böylece, saldırganın ele geçirdiği anahtarlardan faydalanma süresi kısalmıştır.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the nature of Wireless Sensor Networks(WSNs), such as being in hostile environment, unattended and the geographic constraints which prevent reusability of wireless nodes, these nodes are preferred to be manufactured with low cost. Besides this, the application fields of WSNs, like battlefield surveillance and habitat monitoring need security precautions in order to work as intended [2].

For secure communication among sensor nodes, the symmetric encryption is preferred due to low energy consumption and faster processing. For this purpose, the distribution of symmetric keys is obligatory and its difficulty is the main problem of secure communication in WSNs.

In 2002, Eschenauer and Gligor [3] proposed the Random Predistribution Scheme in which all nodes are given a random amount of keys from a large key pool. After deployment, some nodes have the same keys as their neighbors and some do not have. However this scheme results in reasonable levels of connectivity and resiliency against node capture attacks.

Random Predistribution Scheme was further developed by [4], [5] and many others. Chan and Perrig propose q-composite keys scheme, which significantly increases resiliency against small-scale attacks [4]. Du et al. [5] also increases performance of WSNs by using deployment knowledge which prevents unnecessary key assignments.

In these studies, performance metrics are mostly measured for an instant in network lifetime. The change in these metrics in time dimension is mostly ignored, however, in real life examples connectivity, resiliency and availability of resources are changing over time in WSNs. Castelluccia and Spognardi proposed RoK [6], a

robust key predistribution protocol that takes time dimension into account. In RoK [6], the network lifetime is divided into phases. At each phase a number of sensor nodes are deployed to replace nodes that have depleted. These newly deployed nodes are configured with keys from a key pool that is an updated version of the previous one. Therefore, the adversary will not be able to compromise links between newly deployed nodes unless it captures a fraction of these nodes. In order to have connectivity between old and new nodes a gradual update mechanism is applied as described in Chap. 5. As a result, the adversary is not able to compromise new links using the keys that are captured a defined number of phases earlier.

In this study, we investigate the effect of time and make use of it in order to improve the performance of WSNs in terms of resiliency and suggest new performance metrics that describe the performance of WSNs in a realistic way.

This study is divided into two parts that are explained in Chap. 4 and Chap. 5. In Chap. 4, we test different implementation options for Random Key Predistributon scheme in a real network simulation environment. The time in which the key establishments are made constitutes the details in these implementations. The first option for key establishment is to make nodes broadcast their key identifiers to one hop neighbors as soon as they become active. Nodes that receive key identifiers compare their own key identifiers with the received ones and decide on a pairwise key(*link key*) if a match exists(*Proactive Key Establishment Scenario*). The second option also includes proactive key establishments. In addition to that, path key establishments(PKE) are made after all nodes finish exploring link keys(*Proactive Key Establishment w/PKE Scenario*). The third option is to start the application before any key establishment. Nodes that need to communicate in a secure way exchange key identifiers with each other(*Reactive Key Establishment Scenario*). In simulations, we consider energy consumption of nodes and compare these scenarios in terms of resiliency and throughput values. In order to evaluate resiliency we have used new performance metrics besides classical ones. In particular, packet compromise ratio and the ratio of compromised alive links over all existing links were used for the first time in evaluating key distribution performances in WSNs to the best of our knowledge. In our simulations, these metrics show that resiliency of a WSN

might be even worse than what is measured in literature. Another indication of our simulations is that reactive key establishments provide energy preserving that makes the WSN live longer in comparison with proactive key establishments.

In Chap. 5 we improve the resiliency of RoK scheme [6] by further exploiting the time dimension. Our contribution is to use keys that are assigned to future uses, earlier than their times. As a result, we end up with improved resiliency. As explained in Sect. 5.3, we propose two models called Constant Offset Future Random Generations (COFRG) and Growing Offset Future Random Generations (GOFRG). At each deployment phase, both of them choose a time interval in future. Some of the keys from this time interval are chosen randomly and used in the current time. In COFRG, the time interval is a fixed offset from current time. However, in GOFRG this interval has growing offsets with respect to present. In this way, at each deployment of GOFRG, a high fraction of deployed keys become new to adversary. This is valid for COFRG too, but the fraction of new keys at each deployment of COFRG becomes lower after the initial stages of the network. Therefore, the contribution of GOFRG to resiliency is better as compared to COFRG. On the other hand, connectivity decreases in both models due to higher number of nodes that belong to future generations in comparison to RoK. However this backdrop in connectivity is tolerated with path key establishments(PKE).

The rest of this thesis is organized as follows. In Chap. 2 background information on cryptography and networking concepts together with related work are given. Chapter 3 gives the motivation and contributions of this thesis. After that the thesis is divided into two parts. The first part (Chap. 4) gives details about the network simulations which are done to experiment different implementation options for Random Predistributions scheme. In Chap. 5, the second part, MultiPhase Deployment Models are explained.

Chapter 2

Background

2.1 Wireless Sensor Networks (WSNs)

Wireless sensor nodes are manufactured with low cost and limited capacity hardware. The reason is that these nodes are used in large and unattended areas where reusage is almost impossible. Therefore, most of the research aims to find efficient solutions for tiny sensor nodes. One of the popular approaches to this problem is to distribute the computing burden among nodes, then to combine and aggregate the results received from nodes in a hierarchical way so that a global decision is achieved for the whole WSN. In this way, the computing power and energy resources of nodes can be used equally and the lifetime of network can be increased.

This kind of organization among nodes requires security precautions as well. Military applications need to secured against enemy. Commercial applications require to be safe from rival companies. Even healthcare applications need to be secured to provide the privacy of its users. In these applications, data packets that are transfered among nodes for aggregation and transmission towards sink needs to be hidden from adversary and protected against manipulation. Furthermore, the packets in the network should be authenticated. The following section describes these security requirements and explains their solutions.

2.2 Security Requirements

2.2.1 Confidentiality

Confidentiality is defined as allowing only authorised parties to access information. Unauthorised parties can gather no bits of information from material the confidentiality of which has been provided. In order to achieve this, the agreed solution is to encrypt the material and give the authorised parties the ability to decrypt. The limitations of WSNs primarily affect how and to what extent this is achieved.

2.2.2 Integrity

Integrity is the ability of involving parties to understand whether the message has been altered after it was sent. This is generally provided by a message extension that is produced according to message with some secret. Later, the message and its extension are compared in a way that any modification in the message can be noticed. This works given that the extension was not altered by adversary who knows the secret.

In WSNs the messages that go through air can be easily manipulated. Preventing this is not always possible, however, the receiving side can check whether a message has been altered or not. This is what integrity aims.

2.2.3 Authentication

Authentication ensures that the sender of a message has the identity that it claims. In other words, authentication is the knowledge of sender of some secret.

If a group of wireless nodes need to authenticate each other, they can use pairwise keys or a group key that is assigned for all nodes involved. In general, the sender node should provide a cryptographic code of the message using the key. In this way, the receiver side can verify the cryptographic code and make sure of the identity of sender.

2.3 Cryptographic Overview

Security requirements of WSNs that are mentioned above can be fully or partially provided with cryptographic protocols. Actually, these requirements are provided in a more secure way in traditional networks. The reason is that in traditional networks computers are supported with sufficient amount of energy and computational resources. Resource limitations in WSNs obligate alternative and less efficient cryptographic methods for data security.

Basically, there are two ways of data encryption: Symmetric and asymmetric encryptions. These two methods have different features such that according to the needs and specifications of an application area one of them is preferred.

In asymmetric encryption, two separate keys are used such that one is public and the other one is kept private. The owner of private key should keep this key and no other party should be able to access it. The public key should be available to any involving party. This is the main rule of key distribution in asymmetric cryptography.

Asymmetric encryption, which is widely used in computer networks, is not feasible in resource limited sensor nodes. Many researchers, such as [7],[8],[9],[10] and [11] investigate feasible ways to implement asymmetric keying in WSNs, however, recent proposals are still energy and computation hungry approaches.

On the other hand, symmetric encryption uses a single key for both encryption and decryption. This key should be supplied to all authorised parties without revealing information to others regarding these keys. Otherwise both encryption and decryption can be done by unauthorised parties without being noticed. This is the main challenge in symmetric key distribution.

Although asymmetric encryption has two obvious advantages over symmetric encryption, the latter is preferred for its advantages in energy consumption and processing time. One advantage of asymmetric encryption is that dealing with asymmetric key distribution is easier as compared to symmetric key distribution. The other advantage is that private key ownership implies identification of an entity while symmetric encryption requires the existence of the same key in different entities.

In case symmetric encryption is preferred to asymmetric encryption, node-to-node communication is needed in order to perform data aggregation WSN. Encryption in WSNs can be done end-to-end or node-to-node according to the needs of application. In end-to-end encryption, intermediate nodes have no right to access the information that they carry so that these nodes are not able to process data. As a result, in order to enable intermediate nodes to aggregate data along its way to sink, node-to-node encryption is necessary.

In order to perform node-to-node encryption in WSNs, any two nodes that need to transfer data between each other should have a pairwise key.

2.3.1 Key Distribution

In WSNs, two communicating nodes are generally close to each other geographically. On the other hand, two nodes that are away from each other are not expected to communicate. For this reason key distribution should be done in a way that neighboring nodes have common keys. Meanwhile, neighboring nodes are not known prior to their deployment to application area. The reason is that the nodes are generally deployed from airplanes and show a two-dimensional gaussian distribution in terms of their ultimate geographic coordinates.

As mentioned above, secure communication between node pairs is preferred to be with symmetric encryption. One of the extreme approaches is that every node pair should have a common symmetric key. In this case, a WSN that has n number of nodes should assign *n-1* number of symmetric keys to each node in order to guarantee a pairwise key for every node pair. This approach provides 100% connectivity. However, tiny sensor nodes do not afford the memory to store all those pairwise keys. In fact, most of these keys will not be used in the WSN application, since a node is usually a neighbor only to a fraction of the nodes in the network.

Another extreme approach which also provides 100% connectivity and is memory friendly at the same time is using a single key in the whole network. A master key that is assigned to all nodes in the WSN can be used by node pairs to generate random session keys. If adversary somehow learns the master key all communication that is encrypted using the master key is revelaed. Therefore, adversary will be able

to compromise session keys of all node pairs in the network.

As a tradeoff between these two extreme approaches Random Key Predistribution schemes are proposed in literature as detailed in the next section.

2.4 Related Work

In their seminal paper, Eschenauer and Gligor [3] proposed a random key predistribution model, for pairwise key sharing of sensor nodes. This study inspired many researchers and motivated them to propose other random key predistribution schemes. For examples, the authors of [5], [12], [13], [14] and [15] contributed to the area of random key predistributions.

Du et al. [5] use randomness to apply Blom Scheme [16] in large scale WSNs with the advantage of deployment knowledge. Therefore, nodes that are not supposed to communicate due to geographic distance, do not waste their memory to have the same keys.

The study of Yu and Guan [14] divides the field into n-gon shapes and focuses on the parameters of geometric design according to the outcomes of connectivity and resiliency. In these designs, multiple key spaces are used and lambda-degree security is provided.

In a relative study, Mehta et. al. [13] use hash chains and hide key space identifiers to improve resiliency against key space targeted attacks.

In another study, Yang et. al. [17] use the location identifiers as node IDs. Two sensor nodes are able to establish a pairwise key if they know the approximate location of each other, using polynomial calculations. There is no need to use extra memory for unused keys. However, in order to increase the security of polynomial spaces, node-to-node communication is done over temporary chosen group heads. Therefore, there is a need for secure in-group key establishment via another key distribution algorithm.

Camtepe and Yener [12] consider a number of block designs to generate key pools and construct both deterministic and randomized algorithms. This model is referred as Combinatorial Approach which results in improved connectivity with less

memory consumption in comparison to [3].

Anjum [15] adds security to the random key rings of sensor nodes using random beacons from anchor nodes. These beacons are attached to the pairwise keys in order to increase resiliency. In this scenario, obviously neighboring nodes will possibly receive the same beacons and are able to increase the security of their existing keys.

A deterministic method by Dong and Liu [18], uses a number of assisting nodes, that are used only for key establishments. These assisting node correspond to a fraction of 0.8% over all the nodes. In [18] assisting nodes are deployed with hashes of master keys of all the regular nodes. When a key establishment is needed between a couple of nodes, one of them calls for the assitence, then each of the neigbouring assisting nodes provide randoms encrypted with each of the master keys of the nodes in question. The encrypted randoms are sent to the couple. These two nodes xor all the randoms they recieve to get the pairwise key.

The study of Lu et. al. [19] considers the routing mechanism. This work devises a heterogeneous network structure where some of the nodes have extra capabilities in terms of storage, transmission power etc. Applying previous schemes on top of this scenario turns out to have better connectivity measures due to considering routing scenarios.

Chan and Perrig [20] use intermediate nodes for a scalable key establishment algorithm, where communication and memory overheads grow sublinearly with the growth of network size. This study provides higher resiliency using multiple intermediate nodes for a single key establishment.

Castelluccia and Spognardi [6] propose the RoK scheme which limits the activity of sensor nodes to a given amount of time. This requires new nodes to be added to the network sequentially. In return, some of the captured nodes turn out to be useless at the hand of adversary in terms of compromising new links.

The significance of RoK [6] is that it considers another dimension, the time, which is clearly, a point which most of the previous key distribution schemes underestimated while dealing with the problem of compromised keys. In Chap. 4, we improve resiliency of RoK scheme by further manipulating time dimension.

Similar to our study, one of the studies that investigate the timing of key es-

tablishments [21], compare proactive and reactive key establishments. According to [21] proactive key establishment has the advantage of no delays during routing process but storing previously established keys in memory is the disadvantage. On the contrary, reactive key establishment does not establish keys that will not be used so that prevents waste of energy. Traynot et al. [21] focus on comparing Proactive and Reactive methods in terms of average delays, packet losses, effects of mobility etc. without applying attack scenarios. In our study, we consider energy consumption of nodes and apply attack scenarios. One of our contributions is that we have used Directed Diffusion, which is a data centric routing algorithm and is more appropriate for WSNs. Second, our reactive key establishment model is rather energy efficient and has smaller delays.

Chang and Tassiulas [22], propose routing algorithms that distribute routing weight among several paths and maximize the lifetime of the network instead of minimising the absolute energy consumption by using the most efficient path.

In [23], Deng et. al. compare the speed and memory performances of RC4, RC5 and AES on sensor nodes which are used in the proposed INSENS, an intrusion-tolerant routing protocol for WSNs. Using the results achieved in [23], we assume one block of 128-bit AES encryption takes 102 milliseconds in our simulations.

Another study that compares the performances of RC4 and AES [24], implement these algorithms in wireless LANs. This study shows that RC4 should be preferred for large packets and AES for small packets. Despite this study is related to wireless LANs, it is also enlightening for WSNs.

Heinzelman et. al. [25], develop SPIN, Sensor Protocols for Information via Negotiation, which efficiently conveys information with a given amount of energy, 60% better than available schemes and with a close performance to the thoratical optimum.

In studies that take the energy of batteries into account, many different approaches are observed. Gehrke et al. [26] use WINS sensor nodes that have batteries of 35000 Joule capacity. Park and Srivastava [27] use these nodes with 36 Joules of initial energy. Gupta and Younis [28], prefer sensor nodes with 0.5 J. In [29] batteries of 2 J are used.

Silva et. al. provide a summary of the data driven routing, *directed diffusion*, in [1]. They also contribute with introducing two new algorithms that implement directed diffusion.

Finally, NS-2 Manual [30] has been our main guide during the long implementation period with detailed and extensive content and examples it covers.

Chapter 3

Threat Model, Motivation and Contribution of the Thesis

3.1 Threat Model

Adversary in both parts of this thesis can be described as passive, with infinite resources and there is no intrusion detection system at the hand of user to detect adversary. Passive adversary does not alter or harm communication among sensor nodes, but it only captures packets in order to understand the content of messages. Therefore, detection of adversary is not possible through the analysis of packets that arrive to destination. Furthermore, we do not consider any other intrusion detection system in our simulations. Adversary is capable of listening to all packets in the area and is able capture any sensor node except for the sink node. In this way, adversary learns the key rings of nodes. With the help of the key rings adversary is able to decrypt what is sent or received by the captured node. Furthermore, adversary is able to compromise links in other regions of the network if they are established using the keys that are captured. The packets that pass through these compromised links are revealed to adversary.

3.2 Motivation

The majority of the studies in literature deal with static and short-lived WSNs. Concepts like mobility of nodes; changes in network dynamics such as resiliency and connectivity; time related differentiations, such as battery lifetime and the effect of adversary in timely basis are not evaluated thoroughly. First of all, connectivity may

vary due to disconnection of some nodes.This will require deployment of additional nodes to maintain the reliability of the network. In addition to that, adversary will prefer to increase the number of compromised links in time. In case the key pools and key rings remain the same and there is no extra defense, adversary faces no difficulty in approaching 100% compromise ratio of established links given that new nodes are captured continously. All these, indicate that due to changes in network dynamics over time extra precautions are needed in order to maintain resiliency and reliability of WSNs.

Security of a WSN is also changing with time. Adversary may continuously capture nodes and gradually increase its control over network. Furthermore, the established links that are out of reach of adversary might become disconnected from network due to relocation of nodes or depletion of energy source. On the other hand, new secure links might be established with new node deployments or key update mechanisms. Therefore security related parameters are changing throughout network lifetime. As a result resiliency measurements that were done in literature are in majority the initial values and are prone to change in latter stages due to dynamics of the network.

3.3 Contribution of the Thesis

In this study we considered the time dimension in key distribution in WSNs.

In Chap. 4, we implement Random Key Predistributions scheme in three different ways in NS-2, which simulates the network environment in a realistic way. In order to reduce communication overhead and achieve lower energy consumption key establishments require only one broadcast message of each node, except for the path key establishments. We apply attack scenarios and analyze their results in detail. In these analysis, we use packet compromise ratio and the ratio of compromised links over all existing links that were not used before in literature. These resiliency metrics show to what extent adversary is successful in terms of packets and links respectively. Furthermore, we propose packet compromise ratios to be used in measuring resiliency since it is the ultimate goal of adversary.

In Chap. 5, the proposed MultiPhase Deployment Models are discussed. These deployment models are based on the idea that noncaptured keys that are planned to be used in future can be used in present and contribute to the resiliency of the network. In this way GOFRG scheme benefits from the time dimension better than other schemes and increase resiliency while decreasing connectivity due to a trade off mechanism between each other. This feature is notable considering that connectivity can be tolerated with path key establishments but low resiliency cannot be cured.

Chapter 4

Part I: Implementation of a Random Key Predistribution Scheme within Directed Diffusion

4.1 Introduction

Plenty of key distribution schemes for Wireless Sensor Networks (WSNs) exist in literature. However, practical implementations and real life experiments need to be done in order to test the efficiency of these schemes. In this study we focus on the implementation details of the Random Key Predistribution scheme that is proposed by Eschenauer and Gligor [3] and many contributions were made to this scheme by various researchers.

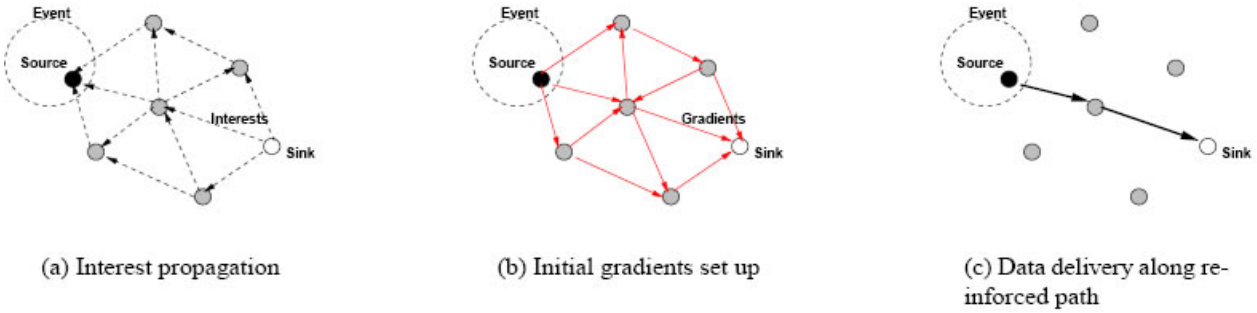
In this study, we simulate and compare three different Random Key Predistribution scenarios, namely Proactive key Establishment without Path Key Establishments(Proactive w/o PKE), Proactive Key Establishment with Path Key Establishments(Proactive w/PKE) and Reactive Key Establishment. We apply attack scenarios to these scenarios and use several performance metrics two of which are novel metrics.

4.2 Background in Directed Diffusion

Traditional routing protocols are interested in end-to-end communications like server-client or peer-to-peer architectures. Therefore, routing is based on addressing of specific nodes in the network. However, in WSNs the interest is mainly on data produced by sensor nodes and not on the nodes themselves. The data that is accu-

mulated from all sensors are treated as a whole. The effort to reach a specific node in the network is not needed in WSNs. Therefore, a new routing paradigm Directed Diffusion is intended to establish a data centric communication in WSNs.

Directed Diffusion aims to transport data from source nodes towards nodes that need this data, which are called *sinks*. At the beginning, nodes that are able to sense and produce data define the type and value range of data they are sensing. On the other hand, nodes that need particular data define the type and value range they are interested in. The rules on how these nodes communicate and transfer data are defined by a number of algorithms: *Two Phase Pull*, *One Phase Push*, *One Phase Pull* etc. [1]. In Fig. 4.1 a schematic that explains the working principle of One Phase Pull algorithm is shown.



(a) Interest propagation (b) Initial gradients set up (c) Data delivery along reinforced path

Figure 4.1: Illustration of One Phase Pull algorithm[1].

In *One Phase Pull* algorithm, which we use in our simulations, a sink node broadcasts *interest messages* to declare its interest on particular data defining its type, source location and range. When these interest messages reach nodes that are able to publish data, these nodes send *data messages* towards the sink using the fastest path that brought the interest message.

With the help of Directed Diffusion, sensor nodes forward data towards sink nodes with less energy and computational resources. In this study, we use directed diffusion to analyze a key distribution scheme and its energy consumption values.

4.3 Motivation

Each node in a Directed Diffusion protocol, is considered an end point that processes and forwards messages if necessary. This requires neighboring nodes to understand each other's messages. This communication is preferably secured with symmetric encryption and in node-to-node basis. For that reason, key distribution and key establishment mechanisms should take place before nodes become in need to encrypt and decrypt messages. Many key distribution schemes are evaluated and discussed in literature. However, the majority of these schemes are applied immediately after wireless sensor nodes are deployed into area. In this case, all neighboring node pairs try to establish a pairwise key, without estimating whether it will be used in following stages. In other words, some keys are established even though they are not needed. This might increase communication overhead which might result in unnecessary energy consumption.

In this study we considered three scenarios. In the first, a key establishment procedure is triggered when a node intends to send an application message to one of its neighbors (*Reactive Key Establishment*). Second, a scenario employs classical key establishment which requires all neighboring node pairs to establish pairwise keys as soon as possible (*Proactive Key Establishment*). In the third scenario, we add path key establishments to the proactive scenario, where node pairs that could not find a common key in their memories ask their neighbors to help them produce a random secret (*Proactive with Path Key Establishment*).

4.4 Methodology

Key establishment algorithm can be triggered directly after node deployment and before application begins or can be scheduled as nodes become in need of pairwise keys to maintain application requirements. In the latter case, the key establishment protocol is employed only between those nodes that require to exchange messages between each other. In this study we analyze three different key establishment scenarios: (i) Proactive key establishment followed by a directed diffusion application, (ii) proactive key establishment (PKE) together with path key establishment phase

followed by a directed diffusion application and (iii) a reactive key establishment scenario where nodes establish pairwise keys as needed by the directed diffusion application. These three scenarios are compared in terms of their data throughput, node disconnection times, network lifetime, the compromise ratio of links and the compromise ratio of packets produced. The method proposed by Eschenauer and Gligor [3] is used as the key establishment scheme and the application is a directed diffusion application called "ping", which is developed by [1] using one phase pull routing algorithm.

4.4.1 Proactive Key Establishment without Path Key Establishments

A number of sensor nodes are deployed into application area with the sink being in the middle. Sensor nodes immediately begin to key establishment protocol by declaring their identities and key identifiers. Neighboring nodes that are able to catch the declaration of each other establish (*link keys*) by using common keys in their key rings. This way all possible secure links are ready prior to directed diffusion application. This phase is completed within seconds. Following this phase the directed diffusion application starts.

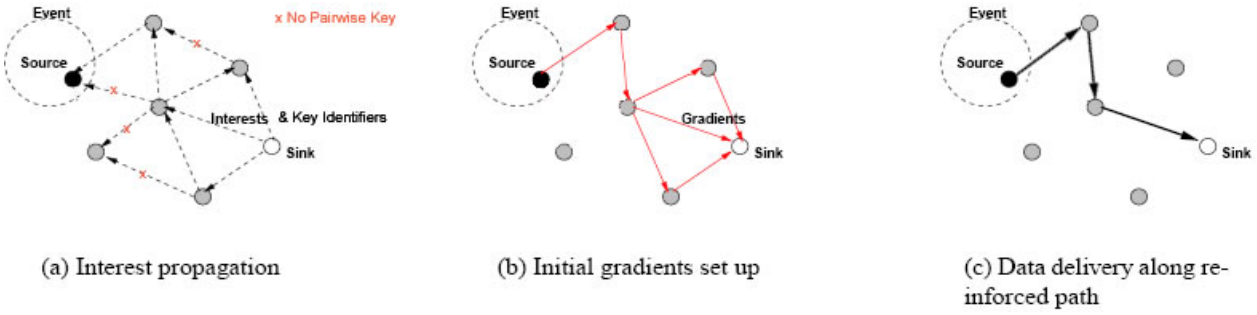
At first the sink broadcasts interest messages to declare its interest on specific information produced by sensor nodes. A fraction of nodes in the area take the role of data publishing (*publisher nodes*) while only one sink waits for this data in the middle of the area. The remaining nodes take the role of processing and forwarding interest and data packets. The interest messages from sink go hop by hop to nodes that produce data. However intermediate nodes that are supposed to receive and forward interest messages from sink to data publisher nodes, ensure that they are able to send data in the opposite direction in a secure way. If they do not share a key with the previous node in the path of the interest message, then they do not forward this interest.

Once an interest message arrives at a publisher node, this node checks if the requested data matches with the value that it produces. If they match, then it responds with a data message along the path that is constructed for the interest

message in the opposite direction. This way a data message can reach the sink.

The sink broadcasts an interest message periodically. Therefore, routes towards publisher nodes are constructed periodically and any failure or extra delays in the previous path is tolerated by establishing a shorter path. Then, data messages from publisher nodes take the new path to reach the sink.

As shown in Fig. 4.2, some of the links that were in data transmission towards sink are not used in secure scenarios. The reason is that these links cannot be secured using a pairwise key between the nodes involved. For this reason, data takes another path towards sink.



(a) Interest propagation

(b) Initial gradients set up

(c) Data delivery along reinforced path

Figure 4.2: Illustration of Secure One Phase Pull algorithm.

4.4.2 Proactive Key Establishment with Path Key Establishments

This scenario is similar to the first scenario except for the path key establishments that begin shortly after link key establishments. First of all, sensor nodes are deployed to application area with the sink being exactly in the middle. Then, all nodes broadcast their identities and key identifiers. In this way, all possible pairwise keys between neighbors are established except for the cases of packet loss due to packet loss probability in our simulations.

According to the rules of this scenario, node pairs that are not able to establish pairwise keys in the first phase (*help seeking nodes*), multicast path key establishment requests to their one hop neighbors. These requests include the IDs of all neighboring nodes (*help waiting nodes*) that the help seeking node could not be able

to establish a pairwise key with. The IDs of corresponding nodes are listed row by row.

A one hop neighbor(*the helper node*) that receives a *one hop path key establishment* request, responds with positive or negative signal for each row in the message. For each positive respond, the node produces a random number as a path key and attaches the random number to the response. These random numbers are encrypted with both the pairwise key of helper-help seeking pair and the pairwise key of helper-help waiting pair. If any of the help seeking node or the help waiting node gets this response, they can use the related random number as the symmetric key. However, in our simulations if only one of the sides get the response while the other misses it, then, a secure communication will not be possible.

Later, nodes that still can not establish pairwise keys, multicast *two hop path key establishment* requests to their two hop neighbors. The intermediary nodes forward these messages if they share a key with the help seeking nodes. When two hop path key establishment messages reach two hop neighbors of help seeker nodes, these helper nodes check the lists inside the messages to produce random numbers for node pairs as possible. The response that contains the random numbers are sent back to intermediate node and the help waiting nodes. Later, intermediate nodes decrypt these responses, encrypt the random numbers with keys shared with help seeking nodes and send the responses to them. This way all nodes can get the path keys they need.

After the link key and path key establishment protocols are completed, the directed diffusion application begins.

The implementation details of PKE includes a few decisions that we have made. In the beginning, a node that could not be able to establish a pairwise key one or more of its neighbors, lists the identifiers of all these nodes and broadcasts the list. In this way, all PKE requests are announced in a single message. At the same time, the helper node sends a response containing all path key candidates together.

For two hop PKE, the helper node does not have to respond back if it is not able to produce a path key. Therefore, a huge number of response messages which are actually useless are avoided.

4.4.3 Reactive Key Establishment

Reactive key establishment refers to exchanging key establishment messages as needed while the application is running. For this reason, as opposed to proactive key establishment scenarios, nodes do not immediately broadcast key establishment requests as soon as they are deployed into area. Instead of this, they begin with the directed diffusion application.

In the beginning of Directed Diffusion application, as interest messages are broadcasted in the network, a node that sends or forwards an interest message for the first time (*interest sender node*) sends a broadcast message declaring its identity and key identifiers. This way neighboring nodes are informed if they have a common key with the interest sender node. Since interest messages are not encrypted, there is no need for the interest sender node to establish keys at this stage. A node that receives an interest message checks if it has a common key with the interest sender node. If it has such a key it continues with the interest forwarding procedure. Otherwise it discards the interest.

When the interest message reaches a publisher node, the publisher node also checks if it has a common key with the interest sender node. If they have a common key, then the publisher node continues with processing the interest message. In case it decides to send a data message back, it uses the same route that brought the interest in the opposite direction. In this route, a node that did not send an interest message before and sends a data message for the first time *data sender node*, broadcasts its identity and key identifiers to make the next hop node in the route begin to key establishment procedure. Furthermore, this broadcast helps other nodes to see if they can establish a pairwise key with the data sender node. As the data message proceeds along intermediate nodes it eventually reaches the sink that is the only node that requires data messages.

As failures in the established routes appear then the same mechanism constructs new routes to maintain the application. This is until no secure link is available due to death of nodes and disconnection of sink from network.

4.4.4 Simulations

Simulation Setting

All simulations were run in ns-2.33 with Cygwin running on Windows Vista using Intel Core 2 Quad 2.4 Ghz CPU.

The sink node is located in the middle of 237 m$\times$ 237 m area. All other 399 nodes are deployed randomly into the area with uniform distribution. The communication range of all nodes is 20 meters. Sensor nodes use 802.11 MAC layer. 100 of them are able to gather the required data with their sensors and forward them as needed while other nodes are intermediary units between sink and data publisher nodes.

The simulations aims to provide performance results that demonstrate the behavior of (i) the scenario with no security, (ii) proactive key establishment w/o PKE, (iii) proactive key establishment w/PKE and (iv) reactive key establishment scenarios.

The main TCL code for simulations in ns-2 is given in Appendix A.

Performance Metrics

The performance metrics that we consider are as follows:

- First Node Death Time
- Last Message Receive Time
- Distribution of Energy Consumption
- Instantaneous Throughput
- Cumulative Throughput
- All Links Compromise Ratio
- Alive Links Compromise Ratio
- Compromised Alive Links over All Links Ratio
- Packet Compromise Ratio

First node death time, refers to the time of the simulation where a node fails due to depleted energy, earlier than all other nodes. This metric indicates the death of the node with the heaviest workload since all nodes start with equal energy capacity.

Sink last message receive time indicates to probable disconnection time of sink from network. If this metric is close to the end of simulation then the sink might have not disconnected from network yet. Instead it may receive more messages if network maintains its functionality. However our simulations are given enough time to distinguish between the sink disconnection time and network life time.

Distribution of energy consumption shows the energy consumption values of encryption and communication seperately. The representaion of these values are in timely basis, so that energy consumption can be analyzed in terms of the phases of the network.

Values related to throughput and resiliency are measured in 5 second intervals of the simulations. *Instantaneous throughput* is the ratio of received data packets by sink over the sent data packets by publisher nodes for each time duration of 20 seconds. *Cumulative throughput* is the ratio of received data packets by sink over sent data packets from publisher nodes from the beginning of simulation until the current time. Apparently, throughput calculations indicate how good the related scenario accomplish its functionality.

Link Compromise Ratio is the ratio of compromised links over all links that are established in network. This ratio does not include the links that belong to captured nodes, since they are already at the hand of adversary. *Live Links Compromise Ratio* is the ratio of compromised alive links over all alive links. *Compromised Alive Links over All Links Ratio* is the ratio of Compromised Alive Links over all existing links whether they are alive or not.

Packet Compromise Ratio is the ratio of number of unique data packets that have flown through one of the compromised links and have reached the sink over all data packets that are produced by publisher nodes and have reached the sink.

Packet Compromise Ratio and Compromised Alive Links over All Links Ratio metrics are our contributions that help for a better understanding of what adversary achieves. First of all, any adversary that compromises links actually intends to

compromise the packets that pass through those links. Therefore, in our simulations packet compromise ratio is our main criteria in defining resiliency.

Second, the effort of adversary in capturing new nodes mainly targets the packets that pass through this node and also the links that can be compromised using the keys in this captured node. However, if adversary succeeds in compromising any link, the activity of this link has a role in the efficiency of this link compromise. In other words, if this link is alive then adversary is really successful since the packets that will be transfered through this link in future will be compromised too. In the contrary, if this link is not alive, i.e. one of the related nodes has failed, then no continuous application traffic will flow through this link. Therefore, the compromise of this link becomes useless unless adversary has captured the messaging history of this link. In this sense, we propose the performance metric Compromised Alive Links over All Links Ratio to determine the efficiency of adversary's activities, thus having an idea of resiliency of network.

Assumptions and Simulation Details

In our simulations, sensor nodes have static locations, so that neighborhoods do not change. Furthermore, the locations and key rings of 400 nodes of all scenarios are kept the same for the sake of fairness in comparisons.

Packets may get lost due to current network conditions. This fact is incorporated in our ns-2 simulations using packet loss probabilities. As a result, misunderstandings between nodes are possible. For instance, *node A* might have sent its key ring but the sent packet might have been lost on the way to *node B*. If *node A* has received the key ring of *node B* and established a pairwise key, *node A* would think that they have a common key to make symmetric encryption but in fact they do not.

In all scenarios, each node broadcasts its ring key only once and this is enough for all link key establishment protocols to be accomplished. In proactive key establishment scenarios, the key rings are sent prior to any other messaging. In this phase, all neighboring nodes know with which neighbors they share keys. However, in reactive key establishment scenario, the broadcast of key rings is not performed

by all nodes. Moreover, a node broadcasts its ring key at most once. At the time a node needs to send its key ring to a group of its neighbors, it just broadcasts. In this way, all neighboring nodes receive the key ring. For this reason, there is no need to rebroadcast the key ring.

In reactive key establishment scenario, ring keys are sent whenever a node intends to send a directed diffusion packet for the first time. In this way, the key ring of this node is known by neighbors, so that they are able to see if they have common keys with this node or not. Using this information they decide to further process the received packet or not. However, in our simulations, key rings reach neighbors after the first directed diffusion packet due to the single thread structure in ns-2.33. Therefore, the first packet has no chance to be processed. In this case, we assume that the key rings do arrive at target nodes and key establishment process goes on accordingly, even if the related key ring packets have not arrived yet.

In all simulations, wireless nodes have an initial energy reserve which degrades according to their energy usage. Packet transmission, packet receive and encryption operations all reduce energy according to their durations and energy consumption coefficients.

All nodes begin with a battery of 200 J each. Transmission power is 0.281838 W [31]. Packet receive and encryption operations require one third of transmission power which is 0.093946 W [25]. Whenever a node performs one of the operations above the power needed for the operation is multiplied by the duration of the operation and the result is reduced from available energy of the node by ns-2 itself (communication energy) or through our implementation (encryption energy). If the node has zero energy, it is considered dead and cannot participate in network anymore.

Sensor nodes use 128-bit AES encryption to secure data packets in the directed diffusion application. Other packets like interest packets, positive reinforcements and negative reinforcements are not encrypted. Each 128-bit AES encryption or decryption are assumed to take 102 milliseconds [23]. Actually no real encryption or decryption is done in our simulations. Instead of this, nodes sleep for an appropriate time duration whenever they are supposed to perform any cryptographic operation

and the estimated consumed energy is reduced from the battery of the node. The equation below gives the amount of energy that is reduced.

$$Encr/DecrEnergy = PacketSize * ByteEncryptionTime * RX\_Power, \quad (4.1)$$

where *PacketSize* is the length of the message in bytes, *ByteEncryptionTime* is the time in seconds that takes to encrypt/decrypt one byte and *RX_Power* is the power in watts that is consumed for receive and processing operations. The resulting energy is in joules.

An attack scenario is applied to Proactive w/o PKE(without Path Key Establishment), Proactive w/PKE(with Path Key Establishment) and Reactive Key Establishment simulations. In all these simulations a random capture model is applied, i.e. adversary captures a random node with a defined *node capture rate* from the beginning of network until its end. The results below show what adversary benefits from this action.

Simulation Results

The result of simulations were extracted on a timely basis. In other words, the change in networking and security metrics over time is our main interest. In this way, different behaviors of scenarios that had been experimented were analyzed not only for an instant time section, but for the entire lifetime of the network. Performance metrics were measured for every 5 seconds of network. In the attack scenario, adversary captures a node in every 20 seconds.

First Node Death Time: A node that depletes its battery earlier than any other node in the network is the first node to die and its death time is a critical metric that shows the moment in time where network begins to lose its nodes. The first node to die is obviously has the biggest workload at the beginning of network, so that it could be said that the performance of network begins to degrade as the most hardworking nodes dies. The second column in Tab. 4.1 shows first node death times of the four scenarios that we tested. The scenario with no security has the best record due to being free of key establishment messages and encryption/decryption

operations. In Proactive w/ PKE scenario, the first node dies at second 162 which is nearly 100 seconds later than other secure scenarios. This shows that thanks to PKE high number of secure links helps to distribute the workload among nodes. Additionally the average lifetimes of nodes in simulations were calculated. Sensor nodes in Proactive w/o PKE has 2913 seconds of average lifetime. In Proactive w/PKE average lifetime of nodes is 1710 seconds and in Reactive scenario it is 2932 seconds.

Table 4.1: Performance results related to battery lifetimes of nodes.

	First Node Death (Sec.)	Sink Last Message Receive (Sec.)
No Security	10556	11455
Proactive w/o PKE	63	3317
Proactive w/ PKE	162	1936
Reactive	60	3450

Sink Last Message Receive Time: The actual lifetime of a network should be measured according to the moment when its functionality ends. In our simulations, the function of Directed Diffusion is to transport messages from publisher nodes to the sink node periodically. Therefore the disconnection time of sink from network can be considered the end of its lifetime, since no messages can be brought to sink anymore. In order to measure the disconnection time of sink, we have decided to look at the last message time received by sink. This does not directly show the disconnection time of sink, however, we have kept simulation times long enough and made sure there were no more messages arriving at the sink. Hence, we treat the sink last message receive time as an indirect estimation to sink disconnection time. As shown in Tab. 4.1, NoSec Scenario has the longest lifetime as expected, since it has no key establishment and security workload. Proactive w/PKE scenario, which performs energy costly PKE, reaches its end of lifetime much earlier than other scenarios. Despite our simulation setting and application algorithm are not much in favor of making use of silent nodes in network, Reactive scenario lives longer than Proactive w/o PKE since it preserves some energy in avoiding useless key establishments between active and silent nodes. More specifically, Reactive scenario

establishes a total of 1850 keys and Proactive w/o PKE scenario establishes 2122 keys in total.

Distribution of Energy Consumption: Figures 4.3, 4.4 and 4.5 show the distribution of energy consumption in Proative w/o PKE, Proactive w/PKE and Reactive scenarios respectively. The consumed energy values are represented in a cumulative way. In all of the figures communication energy is mostly spent in the beginning phases of network. However, the consumption of encryption energy shows that encryption is performed until the end of network, although the encrypted packets in the latter stages of network are mostly lost due to the death of intermediate nodes. In the beginning, all scenarios are busy with key establishments which are performed without encryption except for PKE. This is the reason why the communication energy has the biggest ratio over encryption energy at second 400 of the simulation.

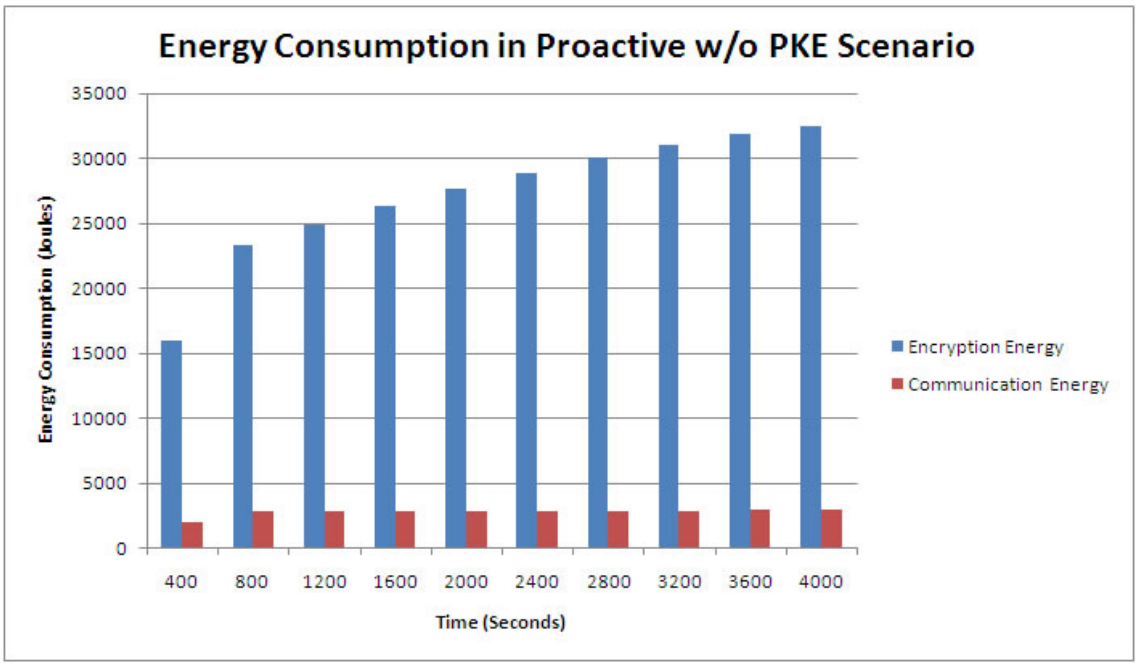


Figure 4.3: Distribution of energy consumption in proactive w/o PKE scenario.

In Proactive w/PKE scenario, path key establishments cause both encryption and communication energy consumption become higher than the values in the other two scenarios. Besides this, Proactive w/o PKE and Reactive scenarios seem very similar in terms of the distribution of energy consumption due to Reactive scenario

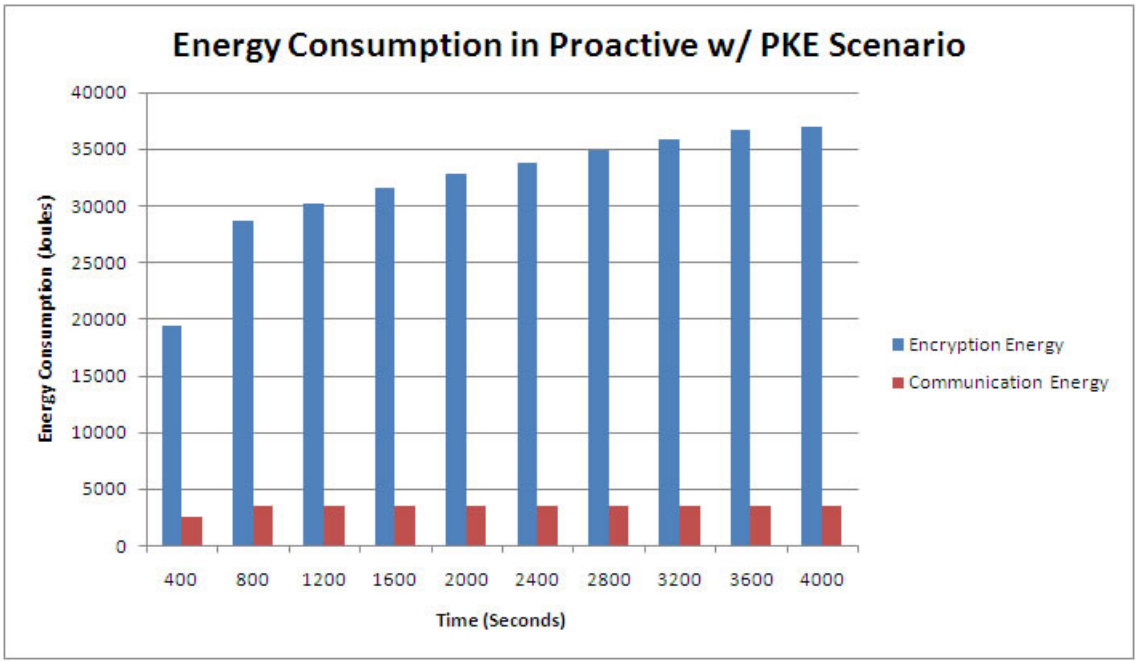


Figure 4.4: Distribution of energy consumption in proactive w/PKE scenario.

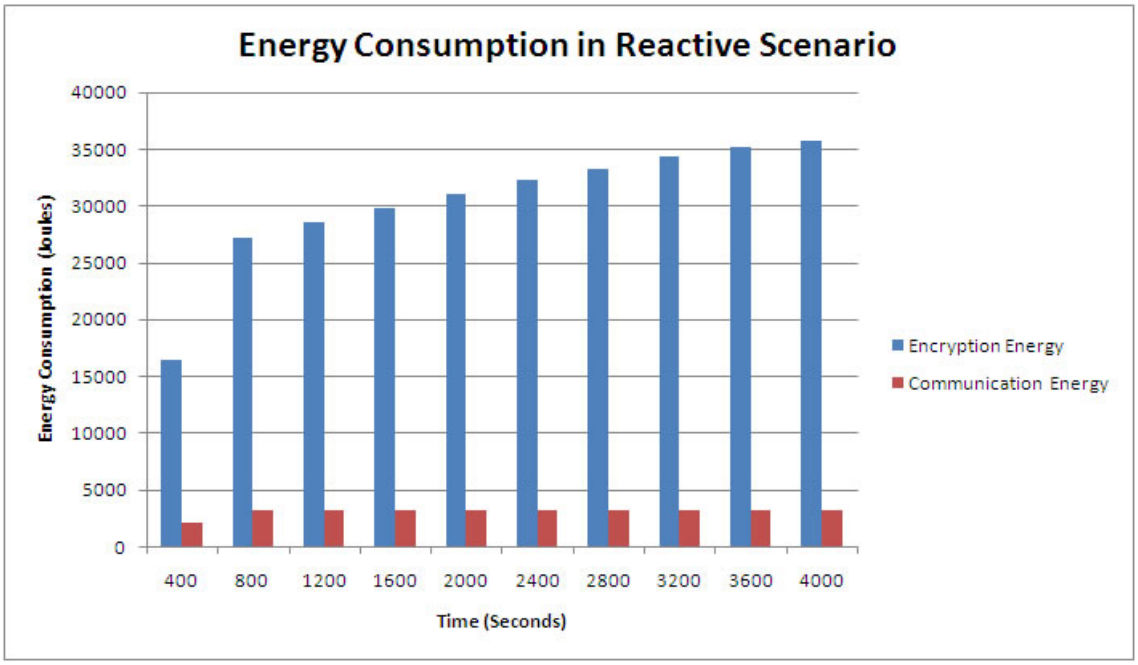


Figure 4.5: Distribution of energy consumption in reactive scenario.

which attempts for a high number of on-demand key establishments before second 400. The encryption energy in Proactive w/PKE scenario reaches 37000 Joules. In Reactive scenario the encryption energy (16454 Joules) is similar to Proactive w/o PKE scenario (15926 Joules) at second 400.

Results show that energy is mostly consumed for encryption. We observe that, encrypting a packet with size of 58 bytes causes to 0.0347 Joules of energy consumption and 0.000175 Joules are consumed for transmission and reception of the same packet. This might justify the big difference between encryption and communication in terms energy consumption.

Instantaneous Throughput: Instantaneous throughput is the ratio of received data packets by sink over the sent data packets by publisher nodes for each 5 seconds period individually. Therefore, immediate drops or increases in performance can be seen clearly. As seen in Fig. 4.6, NoSec Scenario has a very high performance as it goes with over 95% instantanous throughput ratio for most of the time. This is related to being without security overhead and having all available channels open to application traffic.

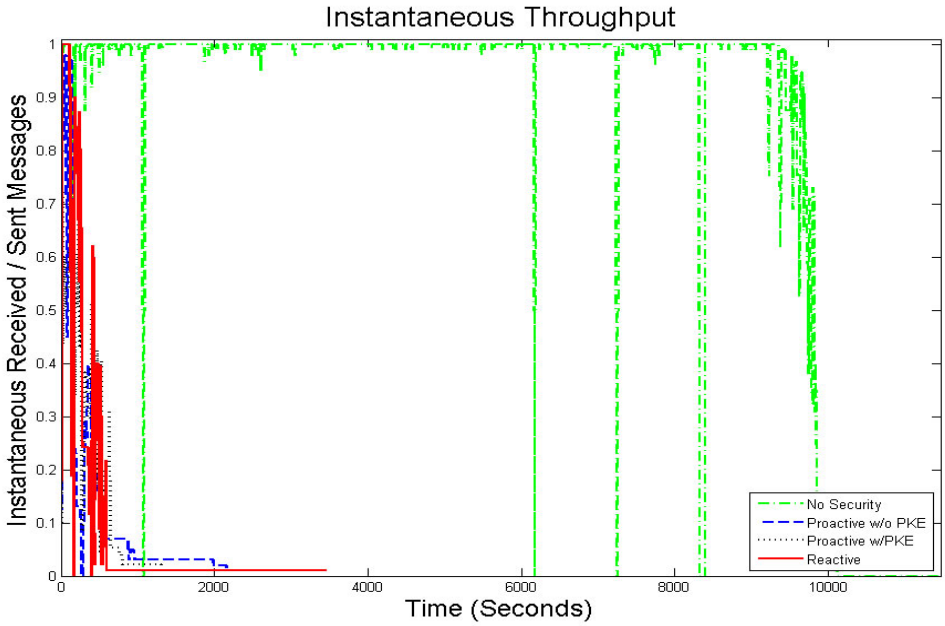


Figure 4.6: Received / Sent Data Packets for each minute individually.

In Fig. 4.7 we have a closer look to the differences between security scenarios.

Reactive scenario accomplishes most of the work in the beginning but its performance decreases towards the end. Proactive w/PKE scenario is much more stable in comparison to other security scenarios however its performance also drop when the majority of its nodes die.

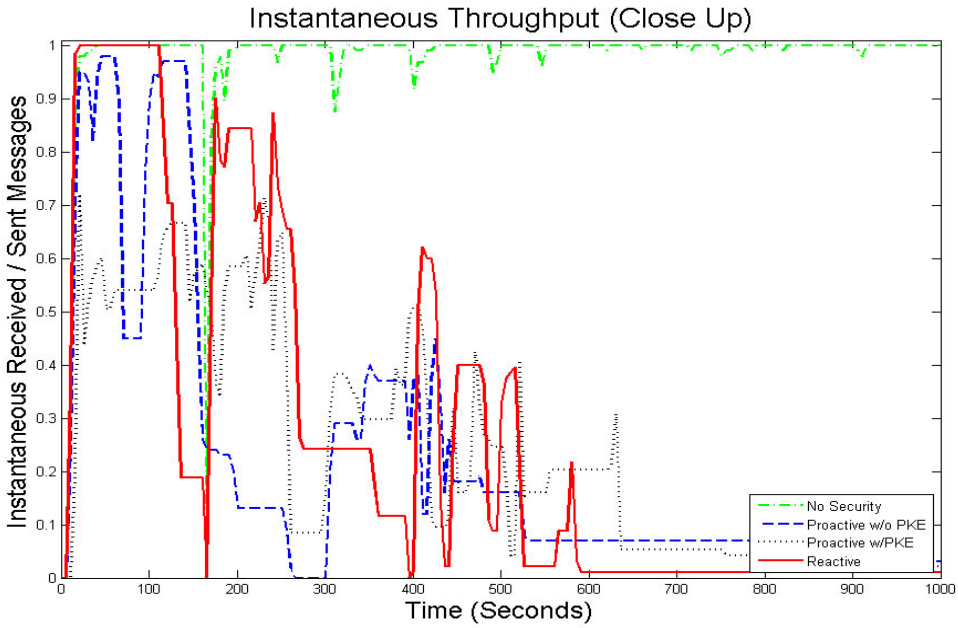


Figure 4.7: Received / Sent Data Packets for each minute individually.

Cumulative Throughput: The received data packets by sink over the sent data packets by publisher nodes until the current time shown in the horizontal axis of Fig. 4.8 is the cumulative throughput. As seen in Fig. 4.9 all scenarios record different throughput performances however the end result of security scenarios are around 0.1. The end result is a better criteria to measure throughput since this the final outcome that shows the number of successful packet arrives over all sent packets cumulatively. Therefore, all scenarios can be considered equal in terms of throughput.

Link Compromise Ratio: Link compromise ratio is the ratio of compromised links over all existing links excluding the links that are established by a captured node. In NoSec scenario, sensor nodes do not encrypt or decrypt messages, for this reason attack scenarios are not applied to NoSec. As seen in Fig. 4.10, all three scenarios, Proactive w/o PKE, Proactive w/ PKE and Reactive, have very

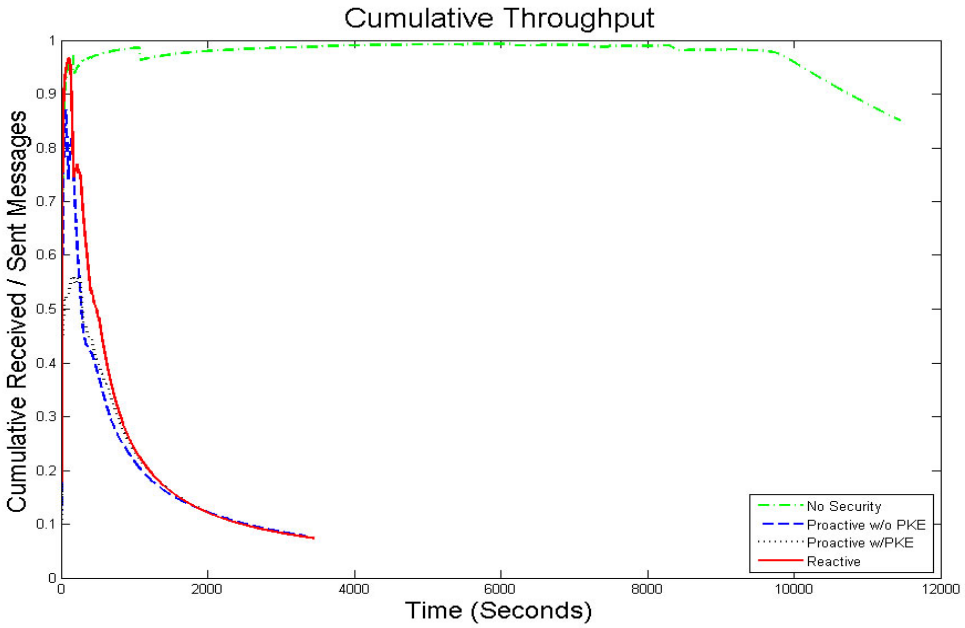


Figure 4.8: Cumulative Received / Cumulative Sent Data Packets.

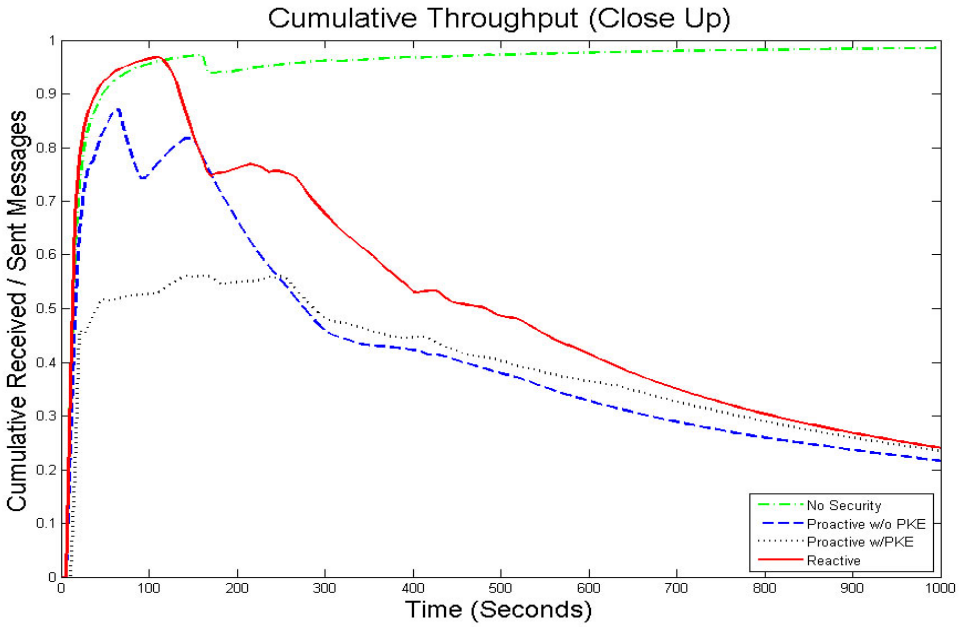


Figure 4.9: Cumulative Received / Cumulative Sent Data Packets.

similar compromised links ratios. This is due to having the same ring and pool sizes, so that the compromised links ratio become similar with the same number of captured nodes. Despite all scenarios begin to lose their node due to their battery depletion, adversary still captures nodes and increases the ratio of compromised links as it reaches around 0.64 for Proactive w/PKE and around 0.8 for Reactive and Proactive w/o PKE at the end of their lifetimes. As discussed below this action of adversary does not yield so much gain as it does in the earlier stages of network.

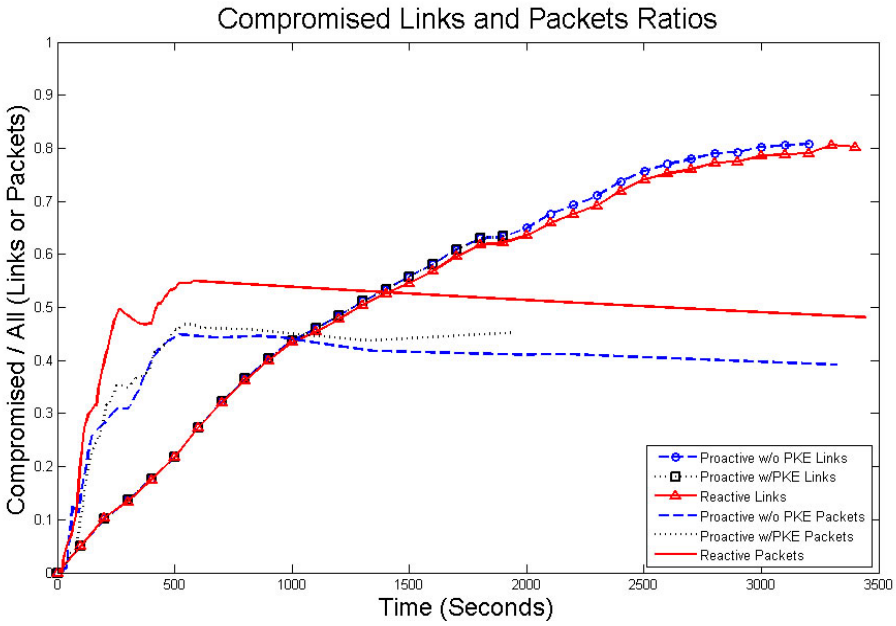


Figure 4.10: Compromised Packet Ratios and Compromised Link Ratios.

Alive Links Compromise Ratio: In this metric dead links, i.e. the links that have a dead node at one of its ends, are subtracted both from nominator and denominator of Link Compromise Ratio. However, the result has no big difference. At the end of Proactive w/o PKE scenario it is 0.02 higher than Link Compromise Ratio. In Proactive w/PKE scenario it is 0.02 less than Link Compromise Ratio. In Reactive scenario it is 0.06 higher than the same metric. Proactive w/o PKE and Reactive scenarios exceed 0.8, while proactive w/PKE scenario gets above 0.6 (Fig. 4.11).

Compromised Alive Links over All Links Ratio: In this metric that we propose, the ratio of alive links over all the existing links in the network is calculated.

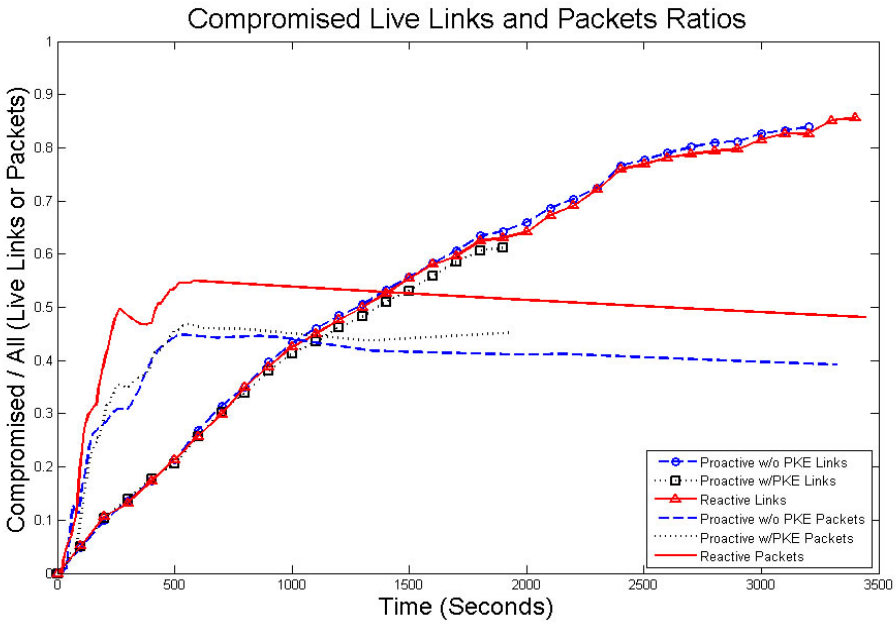


Figure 4.11: Compromised Packet Ratios and Compromised Alive Link Ratios.

As seen in Fig. 4.12 as opposed to other link compromise ratios, the worthiness of adversary's effort is reflected. According to results, this metric approaches to a constant behavior as very few alive links exist in the network. In Proactive w/o PKE it records around 0.5, in Reactive scenario 0.4 and in Proactive w/PKE just below 0.4.

Packet Compromise Ratio: Figures 4.10, 4.10 and 4.10 all show the same packet compromise ratios, which are the division of compromised unique data packets received by sink over all data packets that are received by sink. In all scenarios, packet compromise ratios exceed link compromise ratios which indicates that packets are more prone to compromise than links. In average, a packet travels through several links along its way to sink. Therefore, in case any of the links on its way becomes a compromised link then this packet is a compromised packet. This result is notable in the sense that most of analysis in literature consider link compromise ratios as resiliency metrics. However our results show that a scenario might show low link compromise when it is in fact reveals higher ratio of packets to adversary. At the end of their lifetimes, Proactive w/o PKE has 0.39, Proactive w/ PKE has 0.45 and Reactive has 0.48 of compromised packets ratios (4.10, 4.11, 4.12).

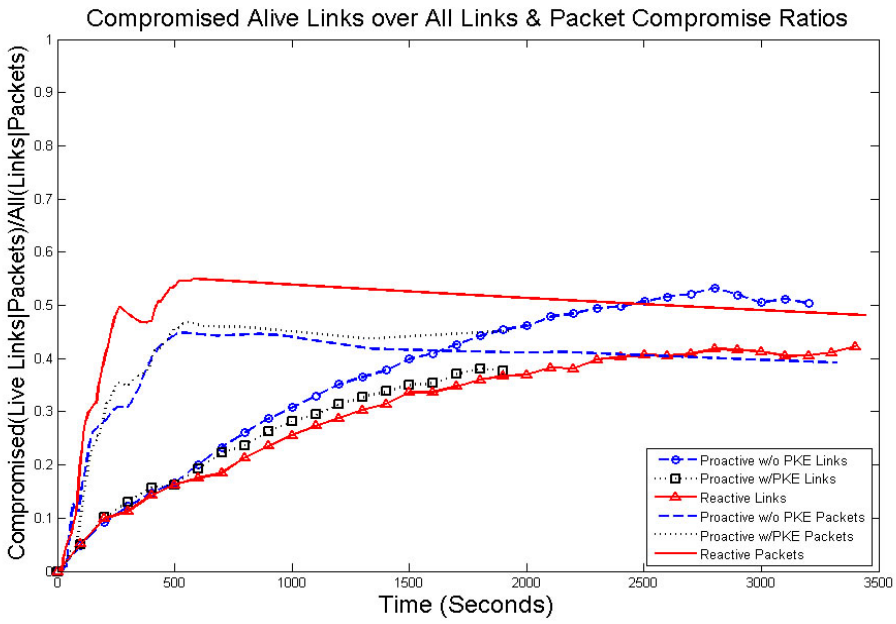


Figure 4.12: Compromised Packet Ratios and Compromised Alive Link over All Links Ratios.

In the attack scenario, adversary captures random nodes from area. However, these randoms in all security scenarios are the same. In other words, adversary captures the nodes that have the same geographic coordinates in the area. We have preferred this approach for the fairness of evaluation and analysis. Meanwhile, the compromise ratio packets are not similar as shown in figures. Therefore, we achieve that the packets in these scenarios take different paths on their way to sink. For this reason, a capture of one of the nodes that has high amount of packet traffic causes high compromise ratio. However the same node that has less amount of traffic in another scenario causes less compromise ratio. This relation is not valid for link compromises, since a captured node has the same key ring in all scenarios. As a result, adversary compromises almost the same umber of links.

4.5 Discussions and Conclusions

The study in this chapter analyze different implementation options of a key distribution scheme. An implementation option decides on which stage of network

a pairwise key should be established. In our simulations, we have tested Proactive scenario without Path Key Establishments(PKE), Proactive scenario with PKE and Reactive Key Establishment scenario. Additionally, a scenario without key establishment is run in order to see the benefits and drawbacks of security precautions. Simulation results are observed on timely basis, so that changes in network dynamics can be observed.

The WSNs that we simulate have fixed number of sensor nodes (400 nodes). A study on scalability of key distribution in WSNs [32], shows the behaviour of resiliency and connectivity when the size of area, number of nodes and sizes of key pool and key rings change. Meanwhile, increasing the number of sensor nodes in our ns-2 simulations might cause less number of established links due to MAC collisions. Therefore, routes to sink would change and routes that can transmit messages to sink faster might be not used at all. Additionally, packets would take longer paths towards sink. In this case, the packet compromise rate is expected to increase.

In order to evaluate the performance of scenarios we use various metrics two of which are novel ones. These two metrics are packet compromise ratio and compromised alive links over all established links ratio. Our simulations in ns-2 show that packet compromise ratio, which is the real target of adversary has higher values than link compromise ratio before majority of nodes die in the network. In our attack model, link compromise ratio and alive compromise ratio are around 0.2 during second 500 of simulations, while at the same time packet compromise ratio is approaching 50% in all scenarios. This shows the low resiiency of a WSN despite all security precautions. Similar results that have relatively high packet compromise ratio as compared to link compromise ratio, can also be observed in other applications which have packets that travel more than one link to reach their targets.

We also focus in energy consumption of nodes and throughput values. Considering these metrics, we use a single step key establishment procedure for implementing Random Key Predistribution scheme. In this way, Reactive Key Establishment which is known to have delays in application traffic due to key establishment during execution, has smaller amount of key establishment traffic and consumes less energy.

Chapter 5

Part II: MultiPhase Deployment Models

5.1 Introduction

This chapter describes two new deployment models Constant Offset Future Random Generations (COFRG) and Growing Offset Future Random Generations (GOFRG) [33]. These models are modifications to RoK scheme [6]. In RoK, network lifetime is divided into phases. At each phase new nodes are deployed into area to replace the old nodes that have failed. In addition to this, at each phase the key pools are updated in a way that adversary is not able to compromise links that are established between new deployed nodes unless it captures a fraction of the new nodes. However new nodes are also able to establish links with old nodes in area. This feature of network is due to a hash chain mechanism. Assuming that adversary captures a node in a given time, this mechanism prevents adversary from compromising a link that is established a defined number of phases after current time, using the keys in the memory of the captured node. Additionally, adversary is not able to compromise a link that was established a defined number of phases earlier than current time, thanks to the same hash mechanism.

5.2 PREVIOUS WORK IN MULTI-PHASE KEYING: THE ROK APPROACH

In $Multi - Phase\ Keying$ models, the network life is divided into phases of equal time intervals. All the keys in the key pool and in the key rings of nodes are updated with each phase, such that the adversary fails to derive the keys of future phases from previously captured keys. At the same time, deriving keys of previous phases from current phase keys is prevented. However for the sake of connectivity among nodes that are active in neighboring phases, this prevention mechanism is activated gradually, as explained below. All the schemes that are discussed in this paper, namely RoK, COFRG and GOFRG are different variations of the $multi - phase$ approach.

Each *phase* is called a *generation* which consists of 10 *rounds*, where one *round* is the smallest unit of time. The reason for this time segmentation is that the attack scenarios are based on *rounds*.

In RoK [6] all the keys are identified with the generation in which they are used. So that, by the end of each generation all the valid keys are updated. However, in order to compromise the maximum number of links, an attacker may prefer to update the key ring of each captured node forever. To prevent this, a security mechanism should be able to guarantee that the key ring of any node is bound to a given amount of time. After exceeding this time, a node should no more establish a secure communication between new deployed nodes with that key ring. The update function should be chosen such that every node should be allowed to update its keys only a given number of times. After exceeding this time, a node should no more establish a secure communication between new deployed nodes, due to being unable to update its keys. In RoK [6] this time duration is set to 10 *generations*, which is almost the maximum battery life of a node. This binding is provided by the backward and forward hash chains.

As a result of this binding, the keys obtained from captured nodes get old by time and new established links remain safe. This decreases the ratio of compromised

links with every *generation*, if adversary stops capturing new nodes. The decrease in this ratio, i.e. the improvement in resiliency, is also called the self-healing of the network.

The working principle of RoK scheme consists of two phases:

- Node Configuration Phase
- Key Establishment Phase

5.2.1 Node Configuration Phase

At the beginning of each generation, a set of sensor nodes are deployed with forward and backward key rings. These key rings are hashed at the end of each generation, so that the new key rings are identified with the new generation. This way nodes maintain their lives among generations. When a key ring is hashed LT times, which is the preassigned cryptographic lifetime of a node, the node can no more establish a new link. In this scheme, forward and backward hash chains, constitute the update mechanism mentioned above, satisfying its security requirements thanks to the irreversibility of hash functions.

Each element of the forward and backward hash chains will be referred as a *Forward Key* or *Backward Key*. The key rings are sets containing a number of chosen *Forward Keys* and *Backward Keys* from the pools, called *Forward Key Pool* and *Backward Key Pool*.

The *Forward Key Pool*, at *Generation* 0, i.e. the first deployment of the network, is defined as follows. Please refer to Table 5.1 for the definitions of symbols:

$$FKP^0 = fk_1^0, fk_2^0, ..., fk_{P/2}^0, \tag{5.1}$$

where each fk_i^0 is randomly generated.

At *Generation* $j+1$, the *forward keys* are refreshed as follows:

$$FKP^{j+1} = fk_1^{j+1}, fk_2^{j+1}, ..., fk_{P/2}^{j+1}, \tag{5.2}$$

where

$$fk_t^{j+1} = H'(fk_t^j) \tag{5.3}$$

Table 5.1: Symbols used in multi-phase keying.

P	Key Pool Size
m	Key Ring Size
FKP	Forward Key Pool
BKP	Backward Key Pool
fk	Forward Key
bk	Backward Key
g_X	The generation of node X
X_i^j	An item X with Generation j and index i
LT	Life Time of the key ring of a node.
H', H"	Two different hash functions.
$fk_i^j - bk_i^j$	A forward-backward key pair.

The Backward Key Pool, is first generated for *Generation* n, i.e. the last generation of the network. The *backward keys* at *Generation* n, are initialized with random values:

$$BKP^n = bk_1^n, bk_2^n, ..., bk_{P/2}^n. \tag{5.4}$$

At *Generation* j, the backward keys are refreshed as follows:

$$BKP^j = bk_1^j, bk_2^j, ..., bk_{P/2}^j, \tag{5.5}$$

where

$$bk_t^j = H'(bk_t^{j+1}) \tag{5.6}$$

Therefore, at *Generation* $j+1$ the *backward key pool* is defined as:

$$BKP^{j+1} = bk_1^{j+1}, bk_2^{j+1}, ..., bk_{P/2}^{j+1}, \tag{5.7}$$

which means that the bk_t^j key of *Generation* j is obtained from the key bk_t^{j+1} of *Generation* $j+1$, using the hash function H'.

Every node is configured with $forward$ and *backward* keys in the following way: For a node with ID A and deployment generation g_A, the i^{th} key of the $Forward\ Key\ Ring$ is the key from the $Forward\ Key\ Pool$ of index $H''(ID_A||i||g_A)$. This is done for all $m/2$ keys in the $Forward\ Key\ Ring$.

For the $Backward\ Key\ Ring$ the same operation is performed using the indices of the $Backward\ Key\ Pool$.

5.2.2 Key Establishment Phase

After deployment, a node A broadcasts ID_A and its generation, g_A. A receiver node B, at first, decides if their generations are close enough or not. This is done by testing if $|g_A - g_B| < LT$. In addition to this, if $g_A < g_B$ and the above holds, then, they can share keys starting from *Generation* g_B up to *Generation* "$g_A + LT - 1$".

Secondly, Node B calculates $H''(ID_A||i||g_A)$ and compares them with its indices $H''(ID_B||i||g_B)$ for all $i, j \in 0, 1, m/2$. If there are collisions such that

$$H''(ID_A||x||g_A) = H''(ID_B||y||g_B), \tag{5.8}$$

where

$$x, y \in 1, 2...m/2, \tag{5.9}$$

then, it is known that they both have the forward key $fk^{g_B}_{H''(ID_B||y||g_B)}$ and the backward key $bk^{g_A+LT-1}_{H''(ID_B||y||g_B)}$ in their memory. This way, all colluding local indices $a, bz \in 1, 2...m/2$ are found and the following becomes their pairwise symmetric key:

$$\begin{aligned} K = H'(&fk^{g_B}_{H''(ID_B||a||g_B)}|| \\ &bk^{g_A+LT-1}_{H''(ID_B||a||g_B)}||...|| \\ &fk^{g_B}_{H''(ID_B||z||g_B)}|| \\ &bk^{g_A+LT-1}_{H''(ID_B||z||g_B)}) \end{aligned} \tag{5.10}$$

Any attacker needs all these $forward$ and *backward keys* to compromise this pairwise key. These keys cannot be reached using a particular $forward - backward$

key pair. A $forward\ key$ is reachable only through a suitable past *forward key* and a *backward key* is reachable only through a suitable future *backward key*. Furthermore, these suitable keys need to have the same indices with the keys in 5.10. Therefore, an adversary would construct a table that is filled with the hash chains of the captured keys. This way future *forward keys* and past *backward keys* can be calculated using the hash function as in 5.3 and 5.6. In case a $forward - backward$ key pair is captured by adversary, the links that were established in the past are secured through $forward\ keys$ since the captured *backward key* will reveal all its hashes, while future links are secured through *backward keys* in the opposite way.

5.3 PROPOSED SCHEMES

The hashing mechanism in RoK [6] and its usage of time dimension through generations provide the $self\ healing$ ability of the network. In our study, we modify the node deployment model of RoK by using nodes of future generations. Therefore the network acts as if it has the state of a few generations later, which results in a faster $self\ healing$ process.

In this study, we propose to use nodes that belong to a random future generation, at each deployment. This method will be referred to as *Future Random Generations*. Two different models on how to choose from future generations are proposed as explained below. In the classical RoK approach, the attacker is able to compromise keys of established links provided that the captured nodes and the link that is to be captured have overlapping generations. In the proposed models, we enable a faster $self\ healing$ and improve resiliency by reducing the probability of overlapping generations via future random generations.

At the end of each generation, some of the nodes including the newly deployed ones have key rings that belong to a few generations ahead. In this way, each node in the network happen to live in a different generation than most of its neighbors. Therefore, this can be referred to as a generation mixture or traveling in time.

The early deployment of *Future Random Generations* would cause a decrease in connectivity. Actually our method creates a trade off between resiliency and

connectivity, which is analyzed in 5.3.3.

5.3.1 Deployment Models

In RoK [6] at each generation, the new nodes that are deployed over the field are chosen such that they belong to current generation. However in our schemes, the generations of the new nodes are chosen randomly. The range of generations from which the generation of each new node is randomly selected is defined as *deployment window*. The position of the deployment window on the time scale shifts towards future at each generation. The rules of shifting the deployment window constitute our deployment models. We propose two such models, namely COFRG (Constant Offset Future Random Generations) and GOFRG (Growing Offset Future Random Generations), that are detailed in the following subsections. In both models, the size of the *deployment window* is fixed to 10 generations.

In our models, each new node is assigned a uniformly random generation picked out of the current deployment window.

Constant Offset Future Random Generations (COFRG)

In COFRG, the deployment window has a constant offset to current generation. The deployment window shifts one by one at each generation. In this way, the offset between the deployment window and the current generation remains unchanged.

In COFRG, the network is initialized without considering the deployment window rules and all the nodes are deployed as *Generation* 0 nodes. However, all nodes to be deployed after *Generation* 0 have generations randomly selected out of *deployment window*.

The discrete uniform random variable that determines the generation of a specific node, G_{COFRG}, is defined as follows.

$$G_{COFRG} = \begin{cases} 0 & \text{if} \quad T = 0 \\ T + D + X & \text{if} \quad T > 0 \end{cases}$$

where X is a random integer uniformly distributed in $\{0, 1, ..., 9\}$, T is the index of current generation and D is the offset to current generation.

At *Generation* T, the deployment window covers the range $T + D$ to $T + D + 9$. The generation of each node to be deployed is a uniform random variable, G_{COFRG}, picked out of this deployment window. In the next generation, $T+1$, the deployment window is shifted one step forward having the range $T + 1 + D$ to $T + 1 + D + 9$. The generation of all nodes to be deployed at $T + 1$ is selected randomly from this deployment window. This goes on for all consecutive generations.

Fig. 5.1 exemplifies both deployment window and the existing generations on the field in COFRG with D = 5. Each cell with dotted background is a deployment window and the symbols in these cells represent the range of generations in that deployment window. The horizontal axis shows the current generation. For an example, the deployment range of *Generation* 4 is between the generations 9 and 18 (inclusive). A node that is to be deployed in the current generation is assigned a future random generation out of the *deployment window* that corresponds to this current generation.

The vertical axis is a reference to observe the existing generations on the field. In addition to dotted background that corresponds to deployment window of current generation, the generations with red grid texture show the ones that have been deployed prior to current generation. For example, the deployed generations at the time of *Generation* 3 are *Generation* 0 and the generations between 6 and 17.

The generations between 1 and 5 are never deployed in any generation. This is due to the constant offset feature of COFRG.

Growing Offset Future Random Generations (GOFRG)

In GOFRG, the deployment window shifts towards future with some jumps. Each node is assigned a generation which is determined by a discrete uniform random variable, G_{GOFRG}, as follows.

$$G_{GOFRG} = \begin{cases} 0 & \text{if} \quad T = 0 \\ (T-1) * JUMP + T + X & \text{if} \quad T > 0 \end{cases}$$

where X is a random integer uniformly distributed in $\{0, 1, ..., 9\}$, T is the index of current generation and $JUMP$ is the length of additional offset.

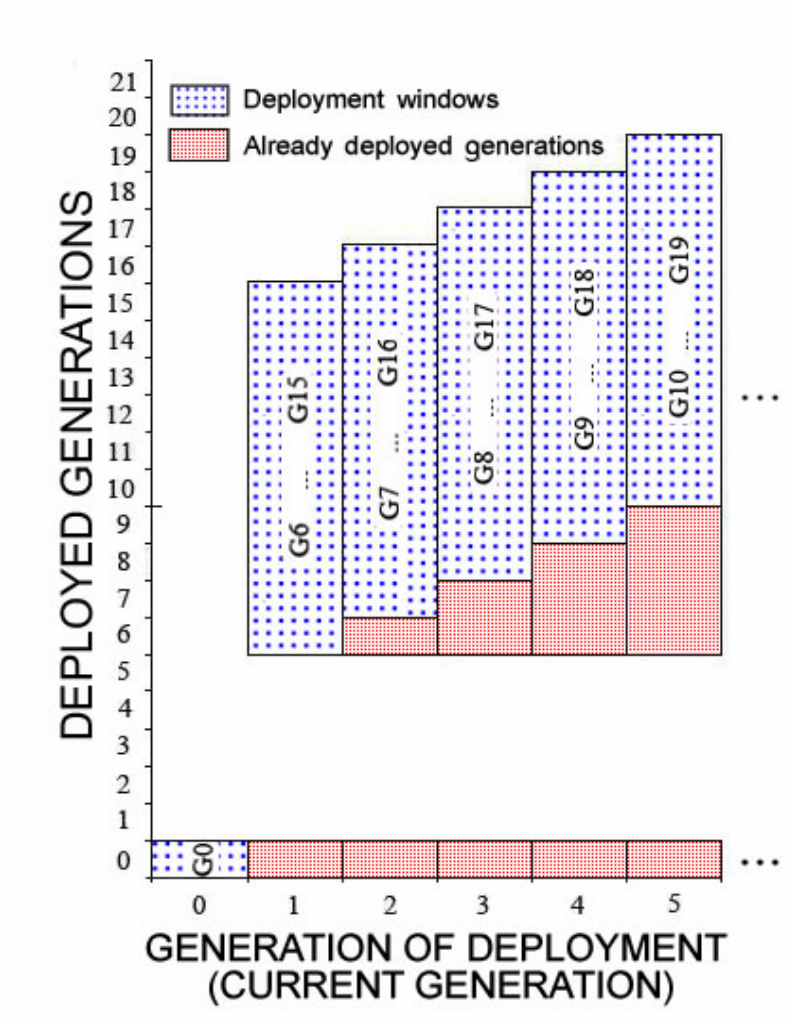


Figure 5.1: Deployed Generations vs. Generations of Deployment in COFRG.

Besides the natural increase in the time scale (one by one), the deployment window in GOFRG makes additional shifts with the length of $JUMP$ at each generation. Hence, a deployment window of GOFRG increases its offset to current generation with constant speed. The $JUMP$ parameter is constant for a given GOFRG model.

Fig. 5.2 illustrates the deployment windows and existing generations upto the fifth generation of the network in GOFRG with $JUMP$=2. The deployment range at *Generation* 3 is between 7 and 16. In this case the deployment has an offset of 4 to current time. The following generation (*Generation* 4) has the deployment window with range 10 to 19, which has an offset of 6 to the current generation. In this way, the difference between the deployment window and current generation increases as generations go by.

For each deployment in both COFRG and GOFRG, the links established using generations that are deployed for the first time cannot be compromised using the nodes captured in previous generations. The number of these safe generations is $(JUMP + 1)/10$ of the length of the deployment window. This ratio is 1/10 in COFRG. Therefore, as compared to COFRG, higher fraction of generations are out of reach of adversary in GOFRG. This difference leads to higher resiliency values for GOFRG as will be discusses in 5.3.3.

In COFRG scheme, the offset is kept constant in order not to be too far from the current generation. Consequently, connectivity is kept within reasonable levels. However, this balance between offset and connectivity is not taken into consideration in GOFRG for the sake of better resiliency.

5.3.2 Key Establishment Phase

The key establishment phases of both models COFRG and GOFRG are identical with RoK, however the results are different as explained in 5.3.3. The generation overlaps in COFRG and GOFRG are fewer compared to RoK. The reason is that, the deployment generations are mostly chosen from future time domains, so the generation overlap probabilities between the key rings of nodes are reduced. Therefore, less node pairs are able to establish shared keys, however, the resulting key becomes

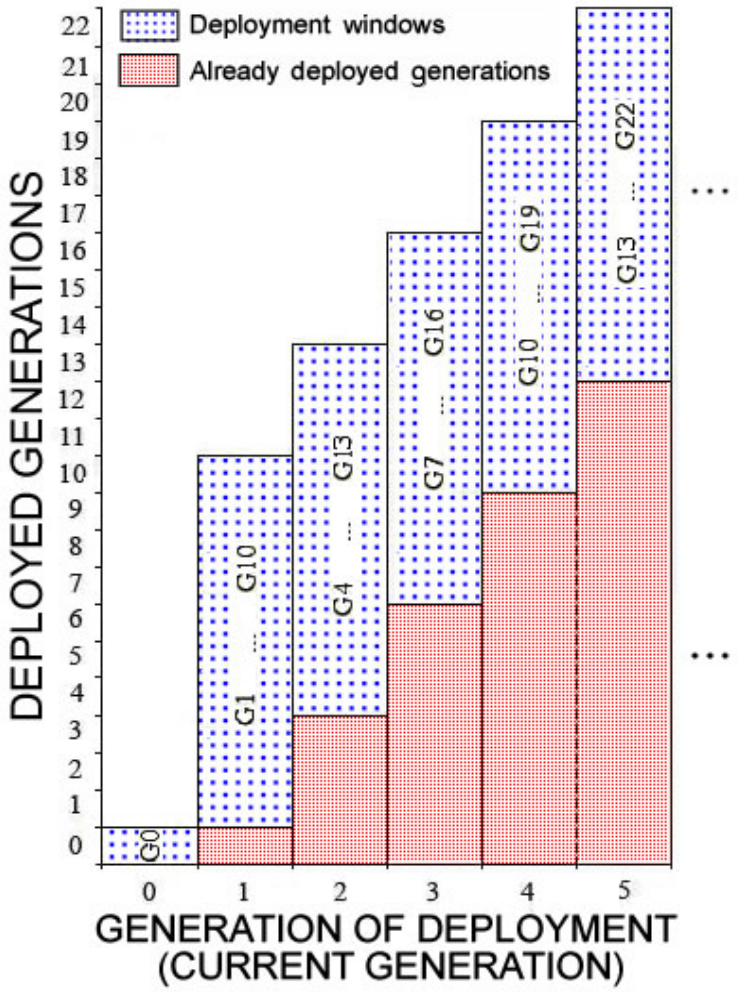


Figure 5.2: Deployed Generations vs. Generations of Deployment in GOFRG.

more resilient in GOFRG than RoK, as explained in 5.3.3.

As in most of the WSN applications, whenever two neighboring nodes are not able to establish a pairwise key using the key rings in their memories, they apply a *path key establishment* procedure in order to communicate in a secure way. The path key establishment phase has the following steps:

1. One of the nodes broadcasts a message that contains the IDs of the two nodes in question, looking for an anchor node that shares a key with both of the nodes.

2. This broadcast is flooded across the network until it reaches an anchor. This step will increase the communication overhead of the nodes involved. Therefore the broadcast is allowed to make at most, say, 3 hops.

3. The anchor node generates a random pairwise key for the two nodes and sends it to both parties using the secure channels established earlier.

The path key establishment is supposed to keep the connectivity in COFRG and GOFRG in desired levels, with a cost of energy consumption due to communication overhead. However, the positive effects of path key establishment on connectivity are not shown in the figures below in order to observe the connectivity prior to path key establishments.

5.3.3 Performance Evaluation

The RoK scheme [6] explains in detail how $Multi-Phase\ Keying$ mechanism improves resilience over time. This behavior is called the *self healing* ability of the network, which addresses the decrease of adversary ability to compromise new links with a given number of captured nodes. As a result the fraction of compromised active links used in the network decreases.

Since our goal is to speed up the $self\ healing$ process and observe the resulting resiliency and connectivity metrics by employing the proposed $Future\ Random\ Generations$ approach, the detailed comparison between previous schemes which was done in [6], is not repeated here.

Simulation Details and Performance Metrics

The simulations were implemented in C# .Net 2005 on Windows XP SP2. Each simulation run 20 times for the sake of accuracy.

COFRG and GOFRG schemes were tested together with RoK. For simplicity a 20*20 grid area is used to deploy 400 nodes. With vertical and horizontal neighboring, each node has exactly 4 neighbors.[1]

At the end of each generation, the nodes that run out of battery are replaced with new nodes, which are configured according to the rules of the related scheme. This replacement obviously is not feasible in real life but to cope and compare the results with RoK scheme, a similar deployment is adopted.

For all scenarios, the sizes of both $forward$ and *backward pools* are 100.000 and the sizes of $forward$ and *backward rings* of a node are both 100.

The lifetimes of nodes are decided according to Gaussian distribution with mean 5 and standard deviation 10/6. As it was calculated in 4, average lifetime of sensor nodes, 5 generations, might be considered 2913 seconds for a proactive scenario.

The simulations were run with two attacker models: In the first group, the attacker, called the constant attacker, captures 5 nodes per *round*. In the second, the attacker, called the temporary attacker, captures nodes only until the end of *Generation* 9, again with a rate of 5 nodes per *round*.

The figures below show two kinds of measurements, the *compromise ratio* and the *local connectivity* for all the models RoK, COFRG and GOFRG. For the calculation of the compromise ratio, all links that are compromised by adversary are counted except the links that belong to captured nodes. This count is divided by the total of all links that belong to non captured nodes. In addition, this ratio was calculated separately for active and total compromised links in order to differentiate between the compromise of active and dead links. Noting that the compromise ratio is the inverse of resiliency metric and the drop in compromise ratio implies the increase in resiliency and visa versa. Here a *dead link* refers to a link which has at least one of its end nodes has gone out of battery. An *active link* is visa versa, i.e. both of its ends have enough battery to communicate.

[1]These parameters are kept the same as [6].

For local connectivity, the key establishment requests between neighboring nodes are counted. In these key establishment attempts, the number of successful ones that end up with valid key establishments were divided into the total of all the attempts. The result show the amount of success of the related scheme in terms of local connectivity. Despite that low connectivity is supported by path key establishments, this support is not reflected the graphs below in order to observe the connectivity performances of all schemes.

Simulation Results

Fig. 5.3 shows the number of all compromised links over all the links established since the beginning of the network versus generations, with the constant attacker model. Here, at the early stages of the network the adversary is able to benefit from the captured nodes and increase the compromise ratio immediately, which is due to the majority of *Generation* 0 nodes in the area. After this early dramatic increase until around *Generation* 5, all the schemes change their behavior. The reason is that by *Generation* 5 the majority of the nodes scattered in *Generation* 0 are out of battery and replaced by new nodes. In this way, all the schemes start to follow a steady behavior. At the beginning of the network, all schemes record above 0.4 compromise ratios. After that, a temporary drop in compromise ratio, implying the improve in resiliency for COFRG and GOFRG schemes can be seen; where GOFRG with Jump 3 reaches around 0.2 compromise ratio. Finally, the resiliency of all schemes begin to drop slowly until the end of the network due to the compromise rate of 50 nodes per generation. Despite this, $multi-phase$ approach prevents the adversary to go beyond 91% of compromise ratio at worst case.

In RoK [6], there is no generation mixture, so at each new deployment only keys belonging to a single generation are introduced to the network. Therefore, they are certainly unknown to adversary at the time of deployment.

On the other hand, for each deployment of COFRG, after the network reaches a steady state after *Generation* 5, the generations of the nodes that are deployed contain already deployed generations with ratio 9/10. In other words, only 1/10 of the deployed nodes are from generations that do not exist in the area yet. This causes

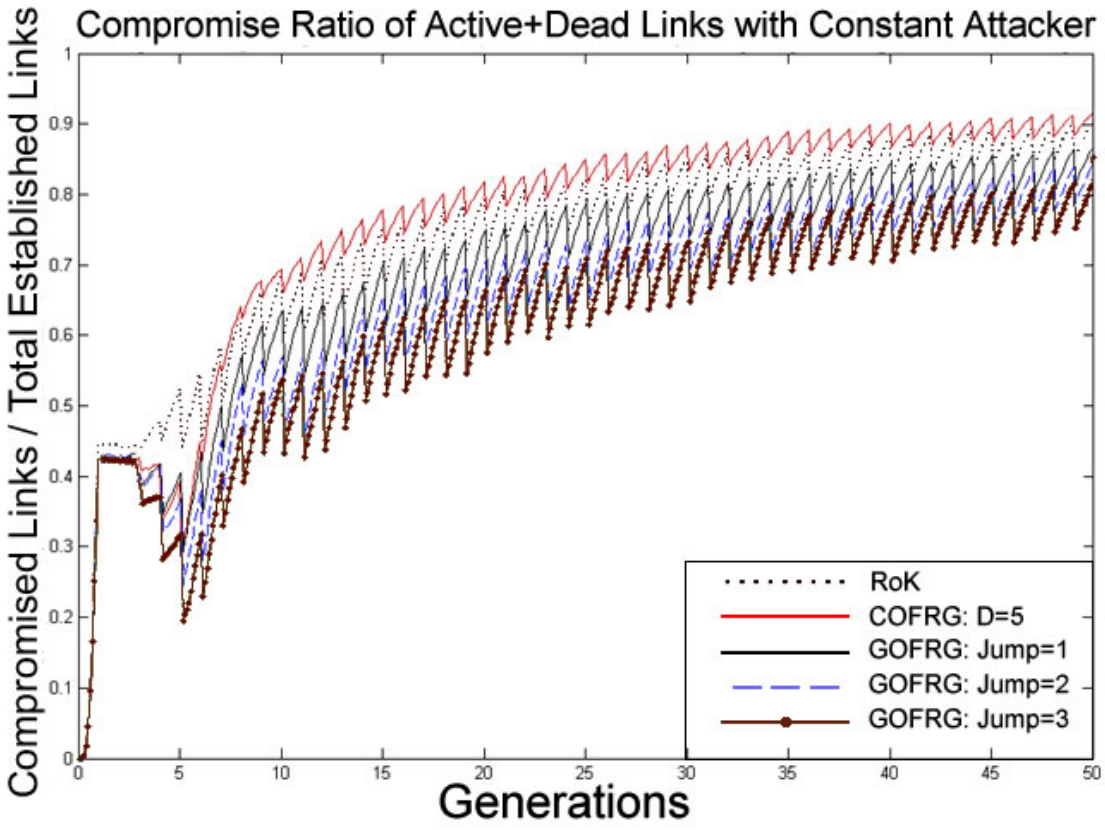


Figure 5.3: Compromise Ratio of Dead and Active Links together, for Constant Attacker Model.

high compromise ratios in the latter stages. However it has around 0.3 compromise ratio at *Generation* 5 while RoK records 0.45 at that time. This advantage for the resiliency in COFRG is due to having $Offset$ 5 from current time and the majority of nodes on the area being of *Generation* 0 (see Fig. 5.1).

The only difference of GOFRG compared to COFRG is the JUMP parameter which is 0 in COFRG and has larger values for GOFRG. The performance of both schemes is similar until *Generation* 5, where a significant drop in compromise ratio is achieved. In order to maintain this performance, at each deployment, extra jumps towards future is made by GOFRG. This results in a better resiliency performance than both of COFRG and RoK throughout the network life.

Meanwhile, using future generations in COFRG and GOFRG pays off with lower connectivity performance (Fig. 5.4). This is due to the nodes from future generations that have lower probabilities of having colliding generations with their neighbors compared to RoK. Fig. 5.1 and Fig. 5.2 show the generation diversity at a given time. As it is seen in Fig. 5.1, with COFRG at *Generation* 5 the generations between 0-19 exist in the network. Therefore it is more difficult for COFRG nodes to have colliding generations between their neighbors, according to RoK which has nearly 10 generations at a given time, considering the battery lifetimes of nodes. The same applies for GOFRG, where the diversity of generations causes loss in connectivity too. In Fig. 5.2, at *Generation* 5 there are 22 generations ranging from *Generation* 0 up to *Generation* 22. However, the low connectivity is tolerated with path key establishments which increase the communication overhead. Despite this communication overhead, the tradeoff between connectivity and resiliency is desirable since resiliency has no alternative.

Fig. 5.5 shows the compromise ratio of active links, which are certainly more valuable than the dead links for most of the applications of WSNs. The compromise ratio of all schemes oscillate with certain equilibrium and do not exceed certain limits, despite the capture rate of 5 nodes per round. In Fig. 5.5, the low compromise ratio of GOFRG throughout the whole network life, compared to RoK and COFRG show that, its high resiliency values is also valid for active links. During the steady state of the network, COFRG is around 0.7 of compromise ratio and RoK oscillates

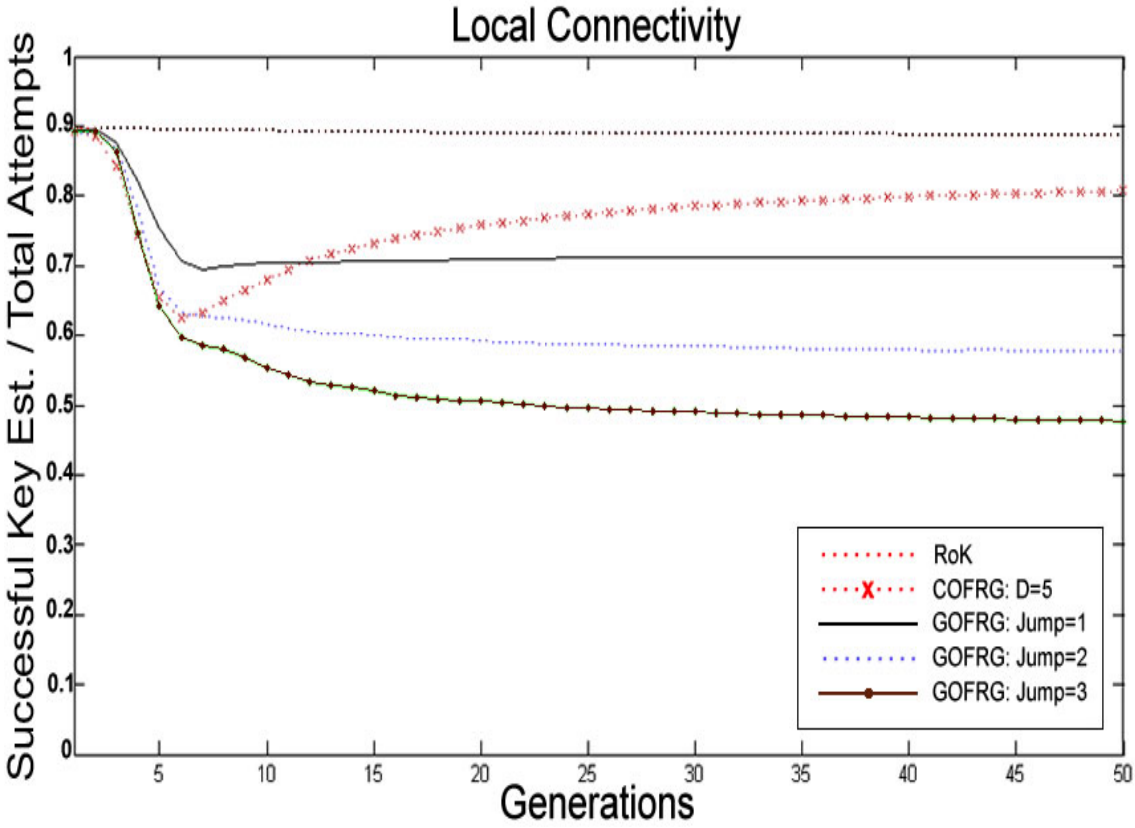


Figure 5.4: Ratio of Successful Key Establishments over all attempts.

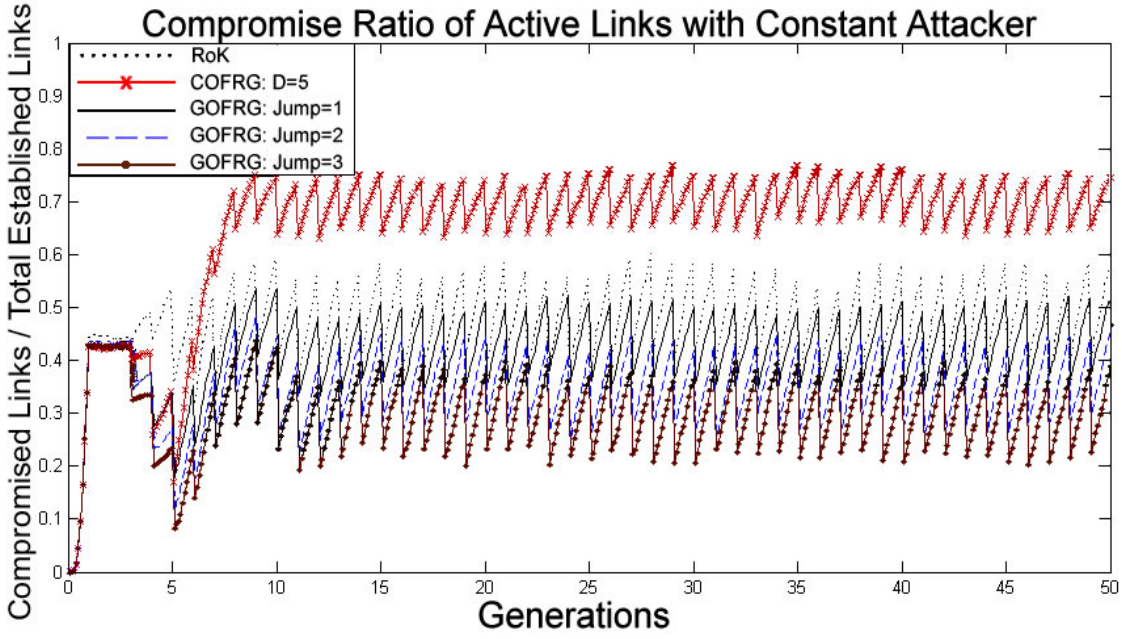


Figure 5.5: Compromise Ratio of Alive Links, for Constant Attacker Model.

between 0.55 and 0.4. However, GOFRG perform better with oscillations between 0.21 and 0.45.

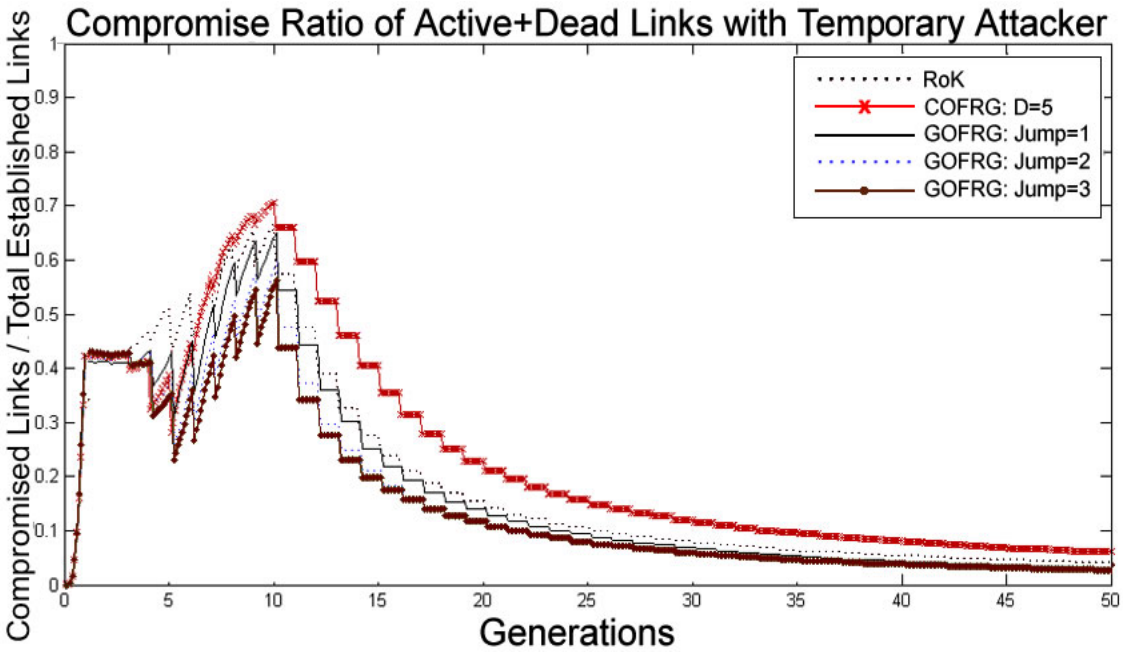


Figure 5.6: Compromise Ratio of Dead and Alive Links, for Temporary Attacker Model.

The temporary attacker in Fig. 5.6, does not compromise any nodes after *Generation* 9. In addition to that, the adversary can compromise new links only until *Generation* 19 in COFRG, which is the extremum case. In GOFRG, adversary can not compromise links after *Generation* 12. Later on, the compromise ratio that does not reach 0 is due to the dead links at the hand of adversary that are taken into account in Fig. 5.6.

Meanwhile, Fig. 5.7 is decisive in terms observing the *self healing* abilities of all the schemes. Fig. 5.7 shows for all schemes that the number of compromised active links becomes 0, which implies that the *self healing* of all schemes manage to heal the network completely. However, GOFRG Jump 3 and GOFRG Jump 2 reach 0 compromise ratio by *Generation* 14. While, GOFRG Jump 1 and RoK achieve it by *Generation* 15. Finally, COFRG achieve the same level by *Generation* 18. Noting that these statistics are not the only difference between the schemes, since until the complete *self healing* achievement of the schemes, high resiliency values of GOFRG against RoK and COFRG is notable.

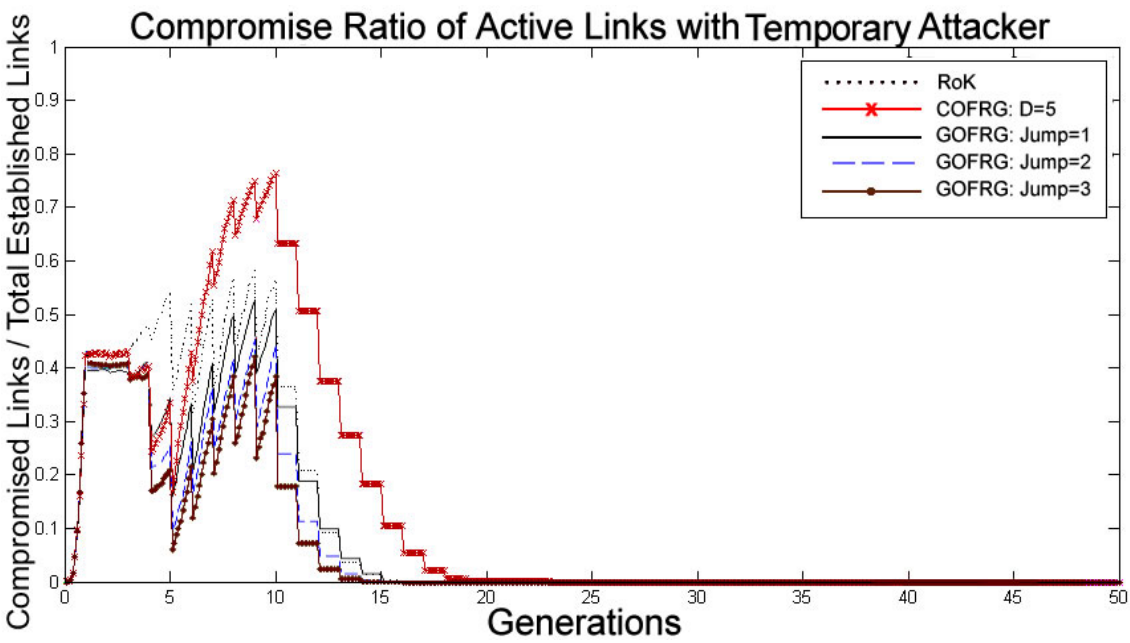


Figure 5.7: Compromise Ratio of Alive Links, for Temporary Attacker Model.

5.4 DISCUSSIONS

The faster *self healing* of GOFRG is observed, however this model gives us a wide range of new issues to consider, like the tradeoff between connectivity and resiliency, dynamic lifetime of key rings and uncaptured nodes that turn to be useless. In this section we discuss these issues.

5.4.1 Connectivity versus Resiliency Tradeoff

Deploying some of the nodes earlier than their own generations obviously will make it harder for the attacker to compromise new links, however it will also make it harder for new nodes to establish keys with older ones. In this case, path key establishments, certainly with some energy cost, bring connectivity to desired levels. Keeping in mind that resilience is rather harder to tolerate, this scenario would be fairly desirable for attack sensitive applications. In these applications, low local connectivity can be balanced by path key establishments in order to a achieve reasonable connectivity levels.

5.4.2 Dynamic Lifetimes of Key Rings & The Problem of Wasted Uncaptured Nodes

In RoK [6] all nodes have a fixed key ring lifetime, LT. Since a random mix of generations are deployed in COFRG and GOFRG, those key rings may expire earlier than expected. Since there may not be any colliding generations between the node in question and the neighboring nodes deployed a few generations later. This will cause a waste in sensor nodes that become useless even though they have enough battery to operate.

5.5 CONCLUSIONS

Despite the limited resources in wireless sensor networks, significant amount of energy and memory are spent for security needs. However, the time dimension is an immature aspect which was not considered to improve performance until recently.

In the recently proposed RoK Scheme [6], the key rings of nodes are updated such that older versions of the same key do not reveal the new version benefiting from the irreversibility of hash chain mechanisms. This scheme results in a *self healing* property of the network that improves resiliency in time.

In this study, we propose to speed up the self-healing process of RoK, which gives better results in terms of resiliency. Two new models, namely COFRG(Constant Offset Future Random Generations) and GOFRG(Growing Offset Future Random Generations) are described and their performances are analyzed and compared to RoK scheme. Both proposed models make use of generations that are assigned for future uses. COFRG keeps the offset to the current generation unchanged. Meanwhile GOFRG makes jumps towards future in order to increase the offset to current generation. $JUMP$ parameter defines the amount of increase in the offset at each generation in GOFRG. In our simulations, the compromise ratio of GOFRG with $JUMP$=3 approaches 0.2 where RoK scheme records more than 0.5 of compromise ratio. That means, GOFRG shows better resiliency as compared to RoK. The local connectivity value for GOFRG with $JUMP$=3 is around 0.5, whereas this metric for RoK is around 0.89. However, local connectivity increases in

GOFRG for smaller $JUMP$ values with a cost of reduced resiliency. These analysis indicate a tradeoff between connectivity and resiliency in our schemes. This tradeoff is the main difference between the proposed GOFRG scheme and RoK.

The COFRG model, which is actually a special case of GOFRG with zero $JUMP$, is a baseline for resiliency in terms of the $JUMP$ parameter. The advantage of GOFRG is that its deployment window shifts more than one generation each time, whereas the deployment window in COFRG shifts one by one. This small difference makes a big effect throughout the network life and resiliency significantly drops in GOFRG. In other words, GOFRG takes the advantage of time dimension in a better way than COFRG.

The advantage of GOFRG in terms of resiliency pays off with low connectivity values. This tradeoff between resiliency and connectivity can be justified considering that connectivity can be tolerated with path key establishments, where low resiliency cannot be cured.

Chapter 6

Conclusions

In this thesis, we first analyze the performance of key predistribution schemes in Wireless Sensor Networks (WSNs) in a realistic scenario that takes the time dimension and network dynamics into account. Using the results obtained via this analysis, we proposed some improvement techniques.

In Chap. 4 we propose two novel resiliency metrics which reflect the aim of adversary better than other performance metrics that are used in literature. One of these metrics, *Packet compromise ratio*, shows that WSNs actually has low resiliency if packets travel more than one link on their way to sink. In our tests and under the assumed attack scenario, an attacker can capture up to 50% of the packets although the classical link compromise metric shows 20% capture rate. These analyses show that there is a need to have more resilient key distribution schemes for WSNs. In 5, we elaborate on these resiliency improvement techniques.

In Chap.5 two schemes that are modifications of RoK scheme [6], are proposed. The working principle of RoK provides a significant improvement in resiliency by enabling self healing ability of the network. In one of our proposed schemes, *GOFRG*, resiliency is further improved with a trade off with connectivity. This improvement is due to a manipulation in the deployment times of nodes. Detailed conclusion of Chap. 4 is in Sect. 5.5.

Although RoK scheme [6] and the proposed *GOFRG* schemes clearly improves the resiliency of WSNs, it is necessary to simulate these schemes using a realistic simulation environment that we use in 4. This is left as a future work.

Bibliography

[1] F. Silva, J. Heidemann, R. Govindan, and D. Estrin, "An overview of directed diffusion," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security.* New York, NY, USA: ACM, 2002, pp. 41–47.

[4] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *In IEEE Symposium on Security and Privacy*, 2003, pp. 197–213.

[5] W. Du, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," 2004, pp. 586–597.

[6] C. Castelluccia and A. Spognardi, "Rok: A robust key pre-distribution protocol for multi-phase wireless sens or networks," in *SecureComm2007, Third International Conference on Security and Privacy in Communication Networks*, 2007.

[7] L. B. Oliveira, R. Dahab, J. Lopez, F. Daguano, and A. A. F. Loureiro, "Identity-based encryption for sensor networks," in *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 290–294.

[8] G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *PerCom Workshops*, 2005, pp. 146–150.

[9] F. Amin, A. H. Jahangir, and H. Rasifard, "Analysis of public-key cryptography for wireless sensor networks security," in *PWASET '08: Proceedings of World Academy of Science, Engineering and Technology*, JULY 31 2008.

[10] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Encryption overhead in embedded systems and sensor network nodes: modeling and analysis," in *CASES '03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems.* New York, NY, USA: ACM, 2003, pp. 188–197.

[11] B. Arazi, I. Elhanany, O. Arazi, and H. Qi, "Revisiting public-key cryptography for wireless sensor networks," *Computer*, vol. 38, no. 11, pp. 103–105, 2005.

[12] S. Camtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," 2004. [Online]. Available: citeseer.ist.psu.edu/camtepe04combinatorial.html

[13] M. Mehta, D. Huang, and L. Harn, "Rink-rkp: a scheme for key predistribution and shared-key discovery in sensor networks," in *IPCCC '05, Proc. 24th IEEE Int'l Performance, Computing, and Comm. Conf.*, 2005.

[14] Z. Yu and Y. Guan, "A key pre-distribution scheme using deployment knowledge for wireless sensor networks," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks.* Piscataway, NJ, USA: IEEE Press, 2005, p. 35.

[15] F. Anjum, "Location dependent key management using random key-predistribution in sensor networks," in *WiSe '06: Proceedings of the 5th ACM workshop on Wireless security.* New York, NY, USA: ACM, 2006, pp. 21–30.

[16] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and applica-*

tion of cryptographic techniques. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 335–338.

[17] C. Yang, J. Zhou, W. Zhang, and J. Wong, "Pairwise key establisment for large-scale sensor networks: from identifier-based to location-based," in *ACM International Conference Proceeding Series Proceedings of the 1st international conference on Scalable information systems*, 2006.

[18] Q. Dong and D. Liu, "Using auxiliary sensors for pairwise key establishment in wsn," in *NETWORKING '07. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*. Springer Berlin / Heidelberg, 2007.

[19] K. Lu, Y. Qian, and J. Hu, "A framework for distributed key management schemes in heterogeneous wireless sensor networks," in *IPCCC 2006, 25th IEEE International Performance, Computing, and Communications Conference*, 2006.

[20] H. Chan and A. Perrig, "Pike: peer intermediaries for key establishment in sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005.

[21] P. Traynor, G. Cao, and T. F. L. Porta, "The effects of probabilistic key management on secure routing in sensor networks," in *WCNC 2008 Proceedings*, 2006.

[22] Chang, "Pike: peer intermediaries for key establishment in sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005.

[23] J. Deng, R. Han, and S. Mishra, "Insens: Intrusion-tolerant routing for wireless sensor networks," *Computer Communications*, vol. 29, no. 2, pp. 216–230, January 2006. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2005.05.018

[24] P. Prasithsangaree and P. Krishnamurthy, "Analysis of energy consumption of rc4 and aes algorithms in wireless lans," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, 2003.

[25] Heinzelman, "Pike: peer intermediaries for key establishment in sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005.

[26] J. Gehrke and P. Seshadri, "Querying the physical world," *IEEE Personal Communications*, vol. 7, pp. 10–15, 2000.

[27] H. Park and M. B. Srivastava, "Energy-efficient task assignment framework for wireless sensor networks," Tech. Rep., September 7 2003. [Online]. Available: F1709182A29FDAE9E0306180528D6C28

[28] G. Gupta and M. Younis, "Performance evaluation of load-balanced clustering of wireless sensor networks," in *In Proceedings of IEEE International conference on communications (ICC*, 2003, pp. 1577–1581.

[29] E. Ould-Ahmed-Vall, G. F. Riley, B. S. Heck, and D. Reddy, "Simulation of large-scale sensor networks using gtsnets," *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, vol. 0, pp. 211–218, 2005.

[30] K. Fall and K. Varadhan, "The ns manual (formerly ns notes and documentation)," pp. 0–0, 2002.

[31] H. Hu and Z. Yang, "Cooperative opportunistic routing protocol for wireless sensor networks," in *International Conference on Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007.*, 2007.

[32] M. Ergun and A. Levi, "Scalability analysis of some key predistribution schemes in wireless sensor networks," TUBITAK 104E071 Project 4th Progress Report, Tech. Rep., March 2008.

[33] O. Z. Yilmaz, A. Levi, and E. Savas, "Multiphase deployment models for fast self healing in wireless sensor networks," in *International Conference on Security and Cryptography, SECRYPT 2008*, JULY 2008, pp. 136–144.

Appendix A

Main TCL Code in ns-2 Simulations

```
# ======================================================================
# Define options
# ======================================================================
set opt(chan) Channel/WirelessChannel
set opt(prop) Propagation/TwoRayGround
set opt(netif) Phy/WirelessPhy
set opt(mac) Mac/802_11
set opt(ifq) Queue/DropTail/PriQueue
set opt(ll) LL
set opt(ant)            Antenna/OmniAntenna
set opt(filters)        OnePhasePullFilter;
set opt(ifqlen) 50 ;# max packet in ifq
set opt(nn) [lindex $argv 0];# number of nodes
set opt(sndr)           100;#  [expr $opt(nn)/2]     ;# no of senders
set opt(rcvr)         1;#[expr $opt(nn)/2]      ;# no of recvrs
set opt(seed) 0.0
set opt(stop) [lindex $argv 1] ;# simulation time
set opt(tr) "routingfirst.tr" ;# trace file
set opt(trccc) "node.nam";
set opt(nam)            "routingfirst.nam"  ;# nam file
set opt(adhocRouting)   Directed_Diffusion
set opt(energymodel) EnergyModel ;
set opt(p_rx) 0.093946 ;#1/3 of p_tx
set opt(p_tx) 0.281838 ;#acc. to threshold.exe
set opt(initialenergy) 200.0;
set Pool [lindex $argv 2] ;#Pool Size
set RingSize [lindex $argv 3];#Ring Size
set Grid 237 ;# Grid Size
set WaitForKD 0;
set CaptureRate 20
set SECURE [lindex $argv 4]; # 0 => No SEUCRITY. 1=> Proactive w/o PKE. 2=> Proactiv

set opt(x) $Grid
set opt(y) $Grid
# ======================================================================
```

```
LL set mindelay_ 50us
LL set delay_ 25us
LL set bandwidth_ 0 ;# not used

Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface

#These values were calculated with Threshold.exe under ~ns/Indep-Utils/propagation
#for parameter distance = 40
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 4.80696e-7;#1.20174e-7;#3.652e-10
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 9.14e+8;#914e+6
Phy/WirelessPhy set L_ 1.0

# ======================================================================
# Main Program
# ======================================================================

#
# Initialize Global Variables
#

set ns_ [new Simulator/MySinkOtcl]
set topo [new Topography]

$ns_ set Secure [lindex $argv 4]; # 0 => No SEUCRITY. 1=> KE. , DD . 2=> KE. 3 , DD

set tracefd [open $opt(tr) w]
$ns_ trace-all $tracefd

#set nf [open $opt(nam) w]
#$ns_ namtrace-all-wireless $nf $opt(x) $opt(y)
```

```
#$ns_ use-newtrace

$topo load_flatgrid $opt(x) $opt(y)

set god_ [create-god $opt(nn)]

#global node setting
$ns_ node-config -adhocRouting $opt(adhocRouting) \
 -llType $opt(ll) \
 -macType $opt(mac) \
 -ifqType $opt(ifq) \
 -ifqLen $opt(ifqlen) \
 -antType $opt(ant) \
 -propType $opt(prop) \
 -phyType $opt(netif) \
 -channelType $opt(chan) \
 -topoInstance $topo \
     -diffusionFilter $opt(filters) \
 -energyModel $opt(energymodel) \
 -rxPower $opt(p_rx) \
 -txPower $opt(p_tx) \
 -initialEnergy $opt(initialenergy) \
 -agentTrace OFF \
     -routerTrace OFF \
     -macTrace OFF


#  Create the specified number of nodes [$opt(nn)] and "attach" them
#  to the channel.
for {set t 0} {$t < $opt(nn) } {incr t} {

set node_($t) [$ns_ node $t]
$node_($t) setRingSize $RingSize
$node_($t) random-motion 0 ;# disable random motion
  $god_ new_node $node_($t)

##Binding the new sink to the MySink pointers of all nodes
$node_($t) setSink $ns_
}

#set trc [open $opt(trccc) w]
#$node_(1) log-target $trc

puts "Sink is ready "
$ns_ set PAR $opt(nn)
$ns_ SetNodeCount
$ns_ set PAR $Pool
```

```
$ns_ SetPoolSize
$ns_ set PAR $RingSize
$ns_ SetRingSize
$ns_ set PAR $Grid
$ns_ SetGridSize
$ns_ set PAR 10
$ns_ SetRange
$ns_ Init

#Makes the configuration and Scattering of all nodes in C++ side
$ns_ Configure

if { $SECURE > 0 } {
for {set t 0} {$t < $opt(nn)} {incr t} {
    set bsrc_($t) [new Application/DiffApp/Broadcast]
    $ns_ attach-diffapp $node_($t) $bsrc_($t)
  $bsrc_($t) setNode $node_($t)
  $bsrc_($t) DBG 0

if { $SECURE < 3 } {
$ns_ at [expr 0.001 * [expr 0+$t]] "$bsrc_($t) start"

if { $SECURE == 2 } {
$ns_ at [expr 2  + [expr 0.001 * $t]] "$bsrc_($t) askHelp"
$ns_ at [expr 4  + [expr 0.001 * $t]] "$bsrc_($t) askHelp2"

set WaitForKD 6
} else {
set WaitForKD 2
}
} else {
set WaitForKD 0
}
}
} else {
set WaitForKD 0
}

for {set t 0} {$t < $opt(sndr)} {incr t} {
  set src_($t) [new Application/DiffApp/PingSender/OPP]

  $ns_ attach-diffapp $node_($t) $src_($t)
  $ns_ at $WaitForKD "$src_($t) publish"
}

for {set t 0} {$t < $opt(rcvr)} {incr t} {
  set snk_($t) [new Application/DiffApp/PingReceiver/OPP]
```

```
  $ns_ attach-diffapp $node_([expr $opt(nn)-1 -$t]) $snk_($t)
  $ns_ at $WaitForKD "$snk_($t) subscribe"
$node_([expr $opt(nn)-1 -$t]) makeSink
$node_([expr $opt(nn)-1 -$t]) freeEnergy
}

for {set t $CaptureRate} {$t <= $opt(stop)} {set t [expr $t +$CaptureRate]} {
$ns_ at $t "$ns_ Attack $t"
$ns_ at $t "$ns_ Throughput $t"
file delete -force -- routingfirst.tr
}

$ns_ at $opt(stop).000000001 "$ns_ Report"
#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).000000001 "$node_($i) reset";
}

# tell nam the simulation stop time
#$ns_ at  $opt(stop) "$ns_ nam-end-wireless $opt(stop)"
$ns_ at  $opt(stop).000000001 "puts \"NS EXITING...\" ; $ns_ halt"

puts "Starting Simulation..."
$ns_ run
```