RESILIENT AND HIGHLY CONNECTED KEY PREDISTRIBUTION SCHEMES FOR WIRELESS SENSOR NETWORKS

by
MURAT ERGUN

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
February 2010

RESILIENT AND HIGHLY CONNECTED KEY PREDISTRIBUTION SCHEMES FOR WIRELESS SENSOR NETWORKS

APPROVED BY

Assoc. Prof. Dr. Albert Levi .....

(Thesis Supervisor)

Assoc. Prof. Dr. Erkay Savaş .....

Asst. Prof. Dr. Hüsnü Yenigün .....

Assoc. Prof. Dr. Özgür Gürbüz .....

Assoc. Prof. Dr. Tonguç Ünlüyurt .....

DATE OF APPROVAL .....

RESILIENT AND HIGHLY CONNECTED KEY PREDISTRIBUTION SCHEMES FOR WIRELESS SENSOR NETWORKS

Murat Ergun
Computer Science and Engineering, MS Thesis, 2010
Thesis Supervisor: Assoc. Prof. Albert Levi
Keywords: Key Predistribution, Security, Key Transfer, Multi-Phase Wireless Sensor Networks

Abstract

Wireless sensor networks are composed of small, battery-powered devices called sensor nodes with restricted data processing, storage capabilities. Sensor nodes collect environmental data, such as temperature, humidity, light conditions, and transmit them using their integrated radio communication interface. In real life scenarios, the exact position of a node is not determined prior to deployment because their deployment methods are arbitrary.

Wireless sensor networks may be used for critical operations such as military tracking, scientific and medical experiments. Sensor nodes may carry sensitive information. In such cases, securing communication between sensor nodes becomes an essential problem. Sensor nodes may easily be impersonated and compromised by malicious parties. In order to prevent this, there is a need for some cryptographic infrastructure. Public key cryptography is infeasible for sensor nodes with limited computation power. Hence symmetric key cryptography mechanisms are applied in order to provide security foundations. Due to resource constraints in sensor nodes, best solution seems to be symmetric key distribution prior to deployment. For each node, a number of keys are drawn uniformly random without replacement from a pool of symmetric keys and loaded in the node's memory. After deployment, neighboring sensor nodes may share a key with a certain probability since all the keys are drawn from the same key pool. This is the basic idea of key predistribution schemes in wireless sensor networks.

Also there are more advanced deployment models that take the change of network in time into consideration. The nodes are powered by batteries and the batteries eventually deplete in time. However the network needs to operate longer than the lifetime of a single node. In order to provide continuity, nodes are deployed and integrated in the network at different times along the operation of the network. These networks are called *multiphase* wireless sensor networks. The main challenge of these networks is to provide connectivity between node pairs deployed at different times.

In this thesis, we proposed three different key predistribution schemes. In the first scheme, we introduce the concept of XORed key, which is the bitwise XOR of two regular (a.k.a single) keys. Sensor nodes are preloaded with a mixture of single and XORed keys. Nodes establish secure links by shared XORed keys if they can. If no shared XORed key exists between two neighboring nodes, they try single keys loaded in their memory. If node pairs do not have any shared XORed or single keys, they transfer keys from their secure neighbors in a couple of ways, and use them to match with their XORed keys. In this scheme, we aim to have a more resilient network to malicious activities by using XORed keys since an attacker has to know either both single key operands or the XORed key itself. We performed several simulations of our scheme and compared it with basic scheme [4]. Our scheme is up to 50% more connected as compared to basic scheme. Also it has better resilience performance at the beginning of a node capture attack and when it starts to deteriorate the difference between the resilience of our proposed scheme and basic scheme is not greater than 5%.

The second scheme that we proposed is actually an extension that can be applied to most of the schemes. We propose an additional phase that is performed right after shared keys between neighboring nodes are discovered. As mentioned above, neighboring node pairs share a common key with a certain probability. Obviously some neighboring node pairs fail to find any shared key. In our proposed new phase, keys preloaded in memories of secure neighbors of a node *a* are transferred to *a*, if necessary, in order for *a* to establish new links with its neighboring nodes that they do not share any key. In this way, we achieve the same connectivity with traditional schemes with significantly fewer keys. We compared the performance of our scheme with basic scheme [4] after shared-key discovery phase and our results showed that our scheme

achieved the same local connectivity performance with basic scheme, moreover while doing that, nodes in our scheme are loaded with three fourth of keys fewer than the keys loaded in nodes in basic scheme. In addition to that, our scheme is up to 50% more resilient than basic scheme with shared-key discovery phase under node capture attacks.

The last scheme that we proposed is designed to be used for multi-phase wireless sensor networks. In our model, nodes are deployed at the beginning of some time epochs, called *generations*, in order to replace the dead nodes. Each generation has completely different key pool. Nodes are predistributed keys drawn uniformly random from key pools of different generations in order to have secure communication with nodes deployed at those generations. In other words, in our scheme keys are specific to generation pairs. This makes the job of attacker more difficult and improves the resiliency of our scheme. We compared our scheme to another key predistribution scheme designed for multi-phase wireless sensor networks. Our results showed that our scheme is up to 35% resilient in steady state even under heavy attacks.

KABLOSUZ DUYARGA DÜĞÜMÜ AĞLARI İÇİN DAYANIKLI VE YÜKSEK BAĞLANTILI ANAHTAR ÖN DAĞITIM ŞEMALARI

Murat Ergun
Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2010
Tez Danışmanı: Doç. Dr. Albert Levi
Anahtar Kelimeler: Anahtar Ön Dağıtımı, Güvenlik, Anahtar Transferi, Çoklu Evreli Kablosuz Duyarga Ağları

Özet

Kablosuz duyarga ağları, duyarga düğümü adı verilen küçük, pille çalışan ve veriyi kısıtlı bir şekilde işleme ve saklama yeteneği olan aygıtlardan oluşur. Duyarga düğümleri sıcaklık, nem, ışık düzeyi gibi ortam bilgilerini toplar ve radyo iletişim arayüzünü kullanarak bu bilgileri iletirler. Gerçek hayatta dağıtım yöntemleri gelişigüzel olduğundan yerleştirilme öncesi kesin durumları belirlenemez.

Kablosuz duyarga ağları ordu takip sistemleri, bilimsel ve tıbbi araştırmalar gibi bir çok kritik öneme sahip operasyon için kullanılabilirler ve duyarga düğümleri hassas bilgileri taşıyabilirler. Bu gibi durumlarda duyarga düğümleri arasındaki iletişimi güvenlik altına almak bir zorunluluk haline gelir. Duyarga düğümleri kötü niyetli gruplar tarafından kolaylıkla taklit edilebilir ve ele geçirilebilirler. Bunu önlemek için kriptografik bir altyapıya ihtiyaç vardır. Sınırlı hesaplama gücü olan duyarga düğümleri için açık anahtar kriptografisi makul değildir. Bu yüzden, güvenliği tesis etmek için simetrik anahtar kriptografi mekanizmaları uygulanır. Duyarga düğümlerindeki kaynak sıkıntısı nedeniyle en iyi çözüm, yerleştirilme öncesi simetrik anahtar dağıtımı olarak gözükmektedir. Her düğüm için belli sayıda anahtar, bir simetrik anahtar havuzundan yerine konmaksızın rastgele bir şekilde çekilir ve düğümün belleğine yüklenir. Tüm anahtarlar aynı anahtar havuzundan seçildiğinden, yerleştirildikten sonra komşu duyarga düğümleri belli bir olasılıkla anahtar paylaşabilirler. Kablosuz duyarga düğümü ağlarındaki anahtar ön dağıtımı şemalarının temel fikri budur.

Ağın zaman içindeki değişimini ele alan daha gelişmiş yerleştirme yöntemleri de mevcuttur. Düğümler pille çalışır ve zaman içinde er geç pilleri tükenir. Diğer taraftan, ağın bir düğümün yaşam süresinden çok daha uzun süre çalışması gerekmektedir. Ağın devamlılığını sağlamak için düğümler ağın çalışması boyunca farklı zamanlarda yerleştirilir ve sisteme dahil edilirler. Bu tür ağlar *çoklu evreli kablosuz duyarga ağları* olarak adlandırılır. Bu ağlarda çözülmesi gereken sorun, farklı zamanlarda yerleştirilen düğüm çiftleri arasındaki bağlantıyı sağlayabilmektir.

Bu tezde üç farklı anahtar ön dağıtım şeması teklif edilmiştir. İlk şemada, iki sıradan anahtarın (tek anahtar) bit bazında XORlanmısıyla (dışlamalı ya da işlemine tabi tutulması) oluşan XORlu anahtar kavramını sunuyoruz. Duyarga düğümlerinin belleklerine tek ve XORlu anahtarların oluşturduğu bir karışım yüklenir. Düğümler ilk olarak XORlu anahtarları kullanarak güvenli bağlantı kurarlar. Eğer hiç ortak XORlu anahtar bulunmuyorsa tek anahtarları kullanmayı denerler. Eğer düğüm çifti arasında ortak herhangi bir XORlu veya tek anahtar bulunmuyorsa çeşitli yöntemlerle güvenli komşularından anahtar transfer eder ve XORlu anahtarlarıyla eşleştirmek için kullanırlar. XORlu anahtarları kullanmamızdaki amaç kötü niyetli faaliyetlere karşı daha dayanıklı bir ağ elde etmektir, çünkü bu durumda bir saldırganın ya XORlu anahtarı oluşturan iki tek anahtarı ya da XORlu anahtarın kendisini bilmesi gerekmektedir. Şemamızın çeşitli simülasyonlarını çalıştırdık ve temel şema [4] ile karşılaştırdık. Şemamız temel şemayla karşılaştırıldığında %50'ye kadar daha bağlantılıdır. Ayrıca düğüm ele geçirme saldırısının başında daha iyi performans sergilemekte ve kötüleşme durumunda temel şema ile arasındaki fark %5'i geçmemektedir.

Önerdiğimiz ikinci şema aslında çoğu şemaya uygulanabilecek bir eklentidir. Komşu düğümler arasında paylaşılan anahtarların keşfinden hemen sonra çalıştırılabilecek ek bir evre teklif ediyoruz. Komşu düğüm çiftlerinin belli bir olasılık ile ortak bir anahtar paylaştığından yukarıda bahsedildi. Açıkçası bazı komşu düğüm çiftleri herhangi bir ortak anahtar bulmakta başarısız olurlar. Teklif edilen yeni evremizde bir *a* düğümü, daha önceden bağlantı kurduğu güvenli komşularının belleğinde tutulan anahtarları gerekli görürse kendisine transfer edebilir ve ortak anahtar paylaşmayan komşularıyla bağlantı kurmak için kullanabilir. Bu yolla geleneksel

şemalardaki aynı yerel bağlantı değerlerini düğüm belleklerinde önemli ölçüde daha az anahtar tutarak sağlayabilmekteyiz. Şemamızın performansını paylaşılan anahtar keşfi evresinden sonraki temel şema [4] ile karşılaştırdık ve sonuçlarımız gösteriyor ki şemamız, temel şema ile aynı yerel bağlantıyı elde etmektedir. Daha fazlası, bunu yaparken şemamızdaki düğümler, temel şemadaki düğümlerden dörtte üç oranında daha az anahtar ile yüklenmektedir. Ayrıca şemamız, düğüm ele geçirme saldırılarında paylaşılan anahtar keşfi evreli temel şemadan %50'ye kadar daha dayanıklı kalmaktadır.

Önerdiğimiz son şema çoklu evreli kablosuz duyarga ağlarında kullanılmak üzere tasarlanmıştır. Tasarımımızda düğümler, ölenlerin yerini almak üzere *nesil* adı verilen zaman aralıklarının başında yerleştirilirler. Her neslin kendisine ait tamamen farklı bir anahtar havuzu vardır. Farklı nesillere ait havuzlardan rastgele seçilmiş anahtarlar düğümlere, o nesillerde yerleştirilmiş düğümlerle güvenli iletişim kurabilmeleri için önceden yüklenir. Başka bir deyişle, şemamızda anahtarlar nesil çiftlerine özgüdür. Bu saldırganın işini zorlaştırır ve şemamızın dayanıklılığını artırır. Şemamızı çoklu evreli kablosuz duyarga ağları için tasarlanmış başka bir anahtar ön dağıtım şemasıyla karşılaştırdık. Sonuçlarımız gösterdi ki, şemamız yoğun saldırılarda bile kararlı durumdayken %35'e kadar daha dayanıklıdır.

To my family

Acknowledgements

I would like to thank my thesis advisor, Albert Levi, for all his support throughout my education including giving advices about the life after university, answering my questions without caring about what time it is, at short, guiding me in all of my works.

I specially thank Erkay Savaş, for helping me through my projects and keeping me tight during my thesis preparation period by basketball.

I also thank Hüsnü Yenigün, Özgür Gürbüz, and Tonguç Ünlüyurt for devoting their time amongst their high volume schedule and joining my jury.

I thank Barış Altop, Can Berk Güder, Duygu Karaoğlan, Emre Kaplan, Ercüment Çiçek, Erman Pattuk, İsmail Fatih Yıldırım, Onur Durahim, Zekvan Yılmaz and all other classmates at FENS 2001 Lab for helping me out in my classes and giving me great time during graduate studies.

I thank my dearest Elif Yücelalp for her mental support during editing of my thesis for sure.

I specially thank my beautiful family for supporting me in every aspects of my life and growing me up to this day without any pay-back.

I also thank Scientific and Technological Research Council of Turkey (TÜBİTAK) for funding me by BİDEB scholarship and supporting this research under grant 104E071.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Wireless sensor networks [2] gained importance in the last decade due to widespread application areas from environmental monitoring to medical use, and from object tracking to military fields. The critical usage areas of sensor networks also provide point of attraction for researchers. Wireless sensor networks are composed of battery powered, small and resource constrained devices called sensor nodes. Sensor nodes are capable of collecting environmental information such as temperature, humidity, and light conditions. Moreover, they can process and store data, both in a limited way, and have short-range radio communication capabilities with integrated sensors. Although they are often called as mobile devices, once they are deployed in a field, most of the time they carry out their operation at the same position till their batteries are depleted. Hence, they usually have permanent set of neighbors. But in real life scenarios, the exact position of a node cannot be determined prior to deployment, because their deployment methods are arbitrary.

There exist some potential security risks in wireless sensor networks when they carry critical data. Sensors communicate via air. Wireless nature of communication provides some advantages to an intruder compared to wired communication. Intruder can surreptitiously listen to communication between sensors without being noticed. Another risk is caused by unconfined sensors in the field. When sensors are used in military fields, one cannot easily control them; they become open to physical attacks, such as impersonation and capture, by enemies. These problems are addressed in [3].

Due to known security problems encountered in wireless sensor networks, security mechanisms, such as encryption and authentication, should be maintained to overcome these threats. However, integration of security mechanisms into wireless sensor networks is not as intuitive as it is thought to be in the first sight because of

resource constraints of the nodes. Wireless sensor networks are usually lack of network topology information prior to deployment and they are formed by non-hierarchical sensor nodes. Unknown infrastructure and unreliable deployment zone make trusted third party solutions such as Kerberos [23] almost inapplicable for wireless sensor networks. Besides, key agreement solutions using public key cryptography such as, Diffie-Hellman [24], is also infeasible for wireless sensor networks that consist of sensor nodes with weak CPUs which are lack of capability of processing expensive public key encryption and decryption operations. Also energy consumption of public key encryption and decryption operations is too much for battery-operated devices and that shortens the life-time.

The remaining solution is conventional symmetric key encryption which does not require too much computational power and energy. Symmetric key cryptography is more CPU-efficient than public key cryptography and can be performed even by sensor nodes. On the other hand, even though data from sensor nodes is sent to the sink node most of the time, it is sometimes needed to be processed (e.g. aggregated) by the intermediate nodes along the path. Necessity of processing data in node-to-node basis requires security in link level. For this reason, symmetric keys are supposed to be distributed in link level instead of end-to-end (e.g. node-to-sink) fashion.

Symmetric key cryptography requires both ends of communication to share a secret key. Here, the main challenge is distribution of keying material in a secure way among the sensor nodes. Many research efforts have been spent to address this challenge. The best solution that survived is key predistribution prior to deployment.

In the literature, the problem of key distribution in wireless sensor networks is addressed by several probabilistic key predistribution schemes such as [4, 5, 6, 7, 8]. One of the earliest approaches is proposed by Eschenauer and Gligor [4]. In Eschenauer-Gligor's basic scheme, there is a global key pool of size P, this key pool contains symmetric keys and their unique key identifiers. k ($k << P$) keys are uniform randomly chosen from this key pool without replacement for each sensor node. These keys form the node's keyring to be stored in its memory. This phase is called *key predistribution phase*. After deployment, nodes broadcast identifiers of the keys in their

keyring. A node receiving these signals finds out which nodes are in its neighborhood as well as their keyring content. Since the keyrings of these nodes are produced from the same key pool, they share a common key with a certain probability. If there is a shared key between neighboring node pairs, this key is used to secure their communication. This phase is called *shared-key discovery phase*.

Eschenauer and Gligor [4] have also proposed a third phase for their basic scheme. This phase is called *path-key establishment phase*. If two neighbor nodes, *a* and *b* do not have a common key, they can try to find an intermediate node *c* such that *a* and *b* are both neighbors of *c* and they have previously established a secure link with *c* by sharing a predistributed key. One of the neighbor nodes, say *a*, generates a path-key and sends it to intermediate node *c* which will forward the path-key to *b*. The generated key is not compromised along the way, because it is sent over secure channels between uncompromised nodes all the time.

1.1. Contribution of the Thesis

In this thesis, we proposed three key predistribution schemes. Most of the works in the literature are compared to basic scheme, because it is one of the first key predistribution schemes and it is basis to the others. So we did compare first two of our proposed schemes with basic scheme. The first scheme consists of three phases. These phases are: (*i*) key predistribution phase, (*ii*) shared-key discovery phase and (*iii*) transfer phase. Our scheme differentiates from most of the key predistribution schemes with an important feature, which is the XORed key. XORed keys are bitwise XOR of two regular keys (will be called as single keys). Keyrings of nodes in this proposed scheme contain a mixture of XORed and single keys.

Neighboring sensor nodes try to find a shared XORed key in their keyring in the first place. If they cannot find a common XORed key, they try to find a shared single

key in their keyring. If both of these attempts fail, they transfer keys from their secure neighbors. The transferred keys are mostly single keys and nodes XOR transferred single keys in a couple of ways to produce an XORed key which is already found in the keyring of other neighboring party.

Our scheme increases the connectivity of basic scheme while keeping sizes of keyring and key pool fixed. Moreover instead of using single keys for securing communications, we encourage the usage of XORed keys, if available. This property constitutes the main frame of the scheme. XORed keys are used for the purpose of increasing resilience of the network, since an attacker has to compromise either both operand single keys used in an XORed key or the XORed key itself to capture a communication link. With a careful setting of the numbers of single keys and XORed keys in the keyrings (70 single keys / 30 XORed keys), local connectivity performance of our scheme increases up to 50% of connectivity achieved in basic scheme [4] and it slightly approaches 1.0 value. Resiliency metric in our scheme stays below that of the basic scheme up to the point where fraction of communications compromised reaches 0.6.

The second scheme that we proposed has a new improvement to basic scheme that does not require intermediate nodes to transfer the generated key. Again *a* and *b* are two neighboring nodes and suppose they do not have a shared key. One of the neighbor nodes, for instance *a*, looks for another neighbor node *c* which has already established secure link with *a* and share a common key with *b*. *a* asks *c* for transferring that common key directly to itself. Moreover *c* does not need to be neighbor with both of *a* and *b* as opposed to path-key establishment phase.

Our scheme consists of three primary phases akin to previous phase. These phases are: (*i*) key predistribution phase, (*ii*) shared-key discovery phase and (*iii*) transfer phase. First two phases are exactly same as the corresponding phases in basic scheme. Our contribution to basic scheme is in the transfer phase. Performance of our scheme is compared with basic scheme [4] after shared-key discovery phase and after path-key establishment phase separately. Our scheme achieves the same local connectivity with significantly fewer keys as compared to abovementioned phases. The keyring size of

nodes in our scheme is 4-fold smaller than the keyring size of nodes in basic scheme after shared-key discovery phase to achieve full connectivity. The keyring size of nodes in our scheme is also smaller than the keyring size of nodes in basic scheme after path-key establishment phase to achieve almost 1.0 local connectivity. Moreover the difference in fraction of communications compromised between our scheme and basic scheme after shared-key discovery phase increases up to 50% with abovementioned keyring sizes. Also we achieve the same resiliency performance with basic scheme after path-key establishment phase with smaller keyring size and same local connectivity.

As it is mentioned, sensor nodes are battery powered and they eventually go out of battery. However, WSNs should function for long periods. Therefore, as the nodes die, new nodes should be deployed in certain intervals, called *generations*, during operation of the network. This kind of WSNs is called *multi-phase wireless sensor networks*. Most of the key distribution studies in the literature are designed for single-phase wireless sensor networks. Even if some of these studies suggest dynamic node additions to the network, the key pools in these schemes contain static keys that do not change in redeployments. As a result, if the network encounters a long term attack and new nodes are added to the system dynamically after this attack, they will be integrated to the network with some already compromised keys in their keyrings. If the attacker continues his/her attack by capturing nodes and acquiring the keyrings of captured nodes, he/she will eventually discover all of the key pool and the network would totally collapse. However, periodic redeployments in multi-phase sensor networks present an important opportunity to reduce the effect of an attacker. In each redeployment, a fresh set of keys may be deployed. Thus after a temporary attack, key pool can recover itself and remove the effects of the compromised keys. In addition, in case of a continuous attack, key pool can keep the rate of damage within a certain level. In such schemes, the connectivity among the nodes in different deployment generations should also be sustained. There is limited work done in the literature about key distribution in multi-phase sensor networks. One of them, RoK scheme [9], is explained in the next section in detail.

As the last scheme, we propose a novel random key predistribution scheme for multi-phase wireless sensor networks which is called RGM (Random Generation

Material) scheme [1]. In our RGM scheme, each generation of deployment has its own random keying material. During shared-key discovery, unique pairwise keys are established between node pairs of particular generations. Here by uniqueness, we mean that nodes of other generations cannot know these keys. Therefore, a captured node cannot be used to obtain keys of other generations. This significantly improves the resiliency of RGM. We conducted simulative performance analyses and compared RGM scheme with RoK [9]. Our analyses show that RGM scheme is up to three-fold more resilient to node capture attacks as compared to RoK scheme. We also show that under heavy attacks, RoK scheme reveals 35% more secure link keys as compared to our RGM scheme. Moreover, with keyring size of 500 keys, our scheme provides 90% local connectivity, which is more than sufficient for a wireless sensor network.

1.2. Organization of the Thesis

The rest of this thesis is organized as the following. Section 2 gives general background information on key predistribution in wireless sensor networks and previous works in the literature. It also describes key predistribution schemes designed for more specialized wireless sensor networks. Sections 3, 4, and 5 are dedicated to three proposed schemes which are briefly mentioned above. Subsection organizations of 3, 4, and 5 are will be separately presented in the beginning of each section. Section 6 concludes the thesis.

2. LITERATURE ON KEY DISTRIBUTION IN WIRELESS SENSOR NETWORKS

There are two naïve solutions for distributing symmetric keys prior to deployment in wireless sensor networks. These are master key and pair-wise key methods. The first one is using a master key which all the nodes in the network share. The nodes use this master key to produce link keys. However, this method is very weak in terms of resilience. Capture of one node leads to the compromise of whole network. In the second method, each node keeps a unique key for every other node. This method gives the best resilience. The problem in this method is limited storage capacity of the sensor nodes. One node has to store n - 1 symmetric keys in its data memory, where n is the number of nodes in the network. That is why the network should be very small. Although both of these methods provide full connectivity in the network, their negative effects or inabilities of sensor nodes cause researchers to find new methods. One approach is using hybrid of these two naïve methods. In this approach, two nodes share a key with a probability. Thus, there is no guarantee that two nodes share a key. In the same way, an attacker can acquire a small portion of keys when a node is captured. This type of schemes creates a trade-off between *local connectivity* and *resilience*. Local connectivity is the probability of two neighboring nodes to share at least one common key. Resilience is the portion of indirectly compromised links over all links at the end of a successful attack of node captures. Indirectly compromised link is a link whose keys are known by the attacker, but none of the sensors in both ends is captured.

2.1. Key Predistribution for Single-Phase Wireless Sensor Networks

Research efforts on key predistribution have been concentrated on probabilistic key predistribution. They renounce connectivity in order to make network more resistant to attacks. In probabilistic key predistribution approaches, there is a pool which contains symmetric keys and their unique key identifiers. A number of keys are drawn without replacement for each sensor node and these keys are loaded into memory of the nodes prior to deployment. These keys form keyring of a sensor node. After deployment, two nodes can establish a secure connection if they have at least one shared key in their keyrings.

One of the earliest probabilistic key predistribution approaches is proposed by Eschenauer and Gligor [4] and this is also called the basic scheme. Basic scheme is composed of three phases: (*i*) key predistribution, (*ii*) shared-key discovery, and (*iii*) path-key establishment phases. In key predistribution phase, k keys are randomly drawn from a key pool of size P, where $k << P$, for each sensor node. These keys are loaded into data memory of sensor nodes prior to deployment. In this way each node forms its keyring.

Shared-key discovery phase starts after all sensor nodes are deployed and they discover neighbor nodes in their communication range. After deployment, nodes broadcast identifiers of the keys in their keyring. A node receiving these signals understands which nodes are in neighborhood as well as their keyring content. Since all keys are drawn from the same pool, two neighbor sensor nodes share a common key with a certain probability. This probability depends on the sizes of key pool and keyring. In other words, connectivity is directly proportional to keyring size and inversely proportional to key pool size. In this phase, all the nodes try to find a key shared between their neighbors. If there is such a key, it is used to secure communication between those two; otherwise, they run path-key establishment phase in which common secure neighbors help in key establishment.

In the path-key establishment phase, if two neighbor nodes, n_1 and n_2 do not have a common key, they can try to find an intermediate node i_1 in key sharing graph such that n_1 and n_2 are both neighbors of i_1 and they have previously established a secure link with i_1 by sharing a predistributed key. As long as the key sharing graph is connected, there is always a path from n_1 to n_2. Definition of key sharing graph is as follows. Suppose, V is the set of all nodes in the network. For any two nodes a and b in V, there exists an edge between them if and only if a and b have at least one shared key and they are in their radio communication range. One of the neighbor nodes, say n_1, will generate a key and send it to intermediate node i_1 which will forward it to n_2. The generated key is not compromised in this way, because it is sent over secure channels all the time. This is *one-hop* version of path-key establishment and it can also be generalized to *t-hop* versions that are using t intermediate nodes instead of one. In this way, generated key has to travel all along the nodes n_1, i_1, i_2, …, i_t, n_2. This extension may work for increasing connectivity; however it may also bring the burden of communication cost. The communication cost increases relative to the length of this multi-hop path.

In the basic scheme, it is likely that a particular key exists in several nodes' keyrings. This is actually a necessity, because otherwise local connectivity reduces. However, having multiple copies of a key is also a potential security problem. An attacker can capture some nodes and acquire their keyrings. Established links secured by using the same keys in acquired keyrings are automatically compromised by the attacker. This weakens the resilience of the network against node capture attacks.

Eschenauer and Gligor's basic scheme is also a framework for our first two proposed schemes. Our schemes improve the resilience and connectivity of the basic scheme.

Chan et al. [7] proposed another scheme called q-composite scheme to increase the resistance of basic scheme. Instead of using only one shared key, Chan et al. offered using two or more shared keys. It is known that using q ($q > 1$) keys instead of one key increase resilience of the network since attacker needs to capture more than one key to compromise a link. In this way, they achieve more durable system against attacks. Chan et al. also proposed an alternative scheme that uses a threshold number of shared keys to

establish a secure link. In other words, two neighbor nodes should have at least some number of shared keys, defined by threshold parameter, to create a secure link. This additional feature can lower connectivity value, but in some conditions it is preferable to have a robust and resilient system rather than a connected one.

Chan et al.'s *q*-composite scheme has also some similarities with two of our schemes proposed in Section 3 and 5, because in our schemes, nodes require more than one key to secure their communication. But *q*-composite scheme does not have a transfer phase which gives great advance in connectivity of the network as in the scheme proposed in Section 3 of the thesis. Moreover, it is not specifically designed for multi-phase wireless sensor networks as in RGM scheme [1] proposed in Section 5 of the thesis.

Differing from probabilistic schemes, in [10], Blom proposed a multipurpose deterministic key predistribution scheme which uses single key space. Each node is able to calculate a pairwise key by storing only $\lambda + 1$ keys in a network of size N $(\lambda << N)$. Compared to the naïve pair-wise scheme in which each node stores N keys, it is highly applicable to wireless sensor networks. In this scheme, there is a property that an attacker cannot compromise any link unless no more than λ nodes have been captured. Besides, if λ nodes have been captured, whole system gets compromised. This is called λ-secure property.

Du et al. [5] further improved Blom's scheme and transformed it to a general probabilistic case that is directly applicable to WSNs. Despite Blom's single key space, Du et al.'s scheme uses a multiple key space approach. This scheme has similar phases with basic scheme. When λ parameter is equal to 0, it is reduced to basic scheme. Basic scheme can be thought as a special case of Du et al.'s scheme in this aspect. In key predistribution phase, τ different key spaces are drawn for each node from a key space pool. After deployment of nodes and discovering neighbors, shared key space discovery starts and if any two nodes share information from the same key space, they can secure their communication link using this key space material. As in basic scheme, Du et al.'s scheme starts path-key space discovery phase in case of absence of shared-key space.

This modification in Du et al.'s scheme converts Blom's scheme to probabilistic key distribution scheme from pair-wise key distribution scheme. The idea behind using multiple key spaces instead of single key space is to make it more resilient than Blom's scheme. If an attacker compromises λ nodes, he/she does not certainly capture a key space, because all of the λ compromised nodes do not possibly share the same key space.

Du et al. [6] extended their scheme by integrating deployment awareness. This information is very valuable if it is known beforehand. Keys can be distributed using some heuristics that close neighbors have more shared keys than further neighbors. As a result, connectivity would have been dramatically increased.

Du et al.'s scheme with deployment knowledge [6] differs from their previous scheme [5] by key predistribution phase. Other phases are totally same as [5]. The scheme proposed in [6] leverages group based deployment model as deployment pattern of sensor nodes. In this model, a total of *N* sensor nodes are split into *t* x *n* groups containing the same amount of nodes. Each group of node is deployed in a geographical zone. Zones are designed as a grid. Hence, neighboring relations are provided in group basis.

It will be useful to mention about key space predistribution phase of [6] as it is different than [5]. In key space predistribution phase, not only sensor nodes but key space pools are split into groups. *Neighboring groups* are defined as groups laying close to each other geographically. Purpose of this phase is letting key space pools used by neighboring groups contain more shared keys and key space pools used by further groups contain fewer shared keys. After key space pools are arranged, τ key spaces are drawn uniformly random without replacement for each sensor node from their own key space pool. So, it is provided that sensor node pairs that are highly probable to be neighbors after deployment have more shared key spaces.

There are lots of works in the general literature of key management in sensor networks. Camtepe and Yener [3], Zhang and Varadharajan [11], Zhou et al. [12], Lee et al. [13] and Xiao et al. [14] provide good surveys and taxonomy about them. In all random key predistribution schemes, there is a trade-off between local connectivity and

resiliency against node capture attacks. Having a large keyring size increases the probability of direct key sharing (local connectivity), but this also gives more keys to the attacker when a node is captured.

2.2. Key Predistribution for Multi-Phase Wireless Sensor Networks

Sensor nodes operate using battery power that eventually depletes. Wireless sensor networks are set up to function for longer period of time as compared to the lifetime of sensor nodes. So, new nodes need to be deployed in some intervals to provide continuity of network. Such wireless sensor networks are called *multi-phase wireless sensor networks*. The intervals at which new nodes are deployed are called *generations*. In the beginning of each generation, dead nodes are replaced by fresh nodes.

Castelluccia and Spognardi [9] proposed a key management scheme called Robust Key Distribution (RoK) for multi-phase wireless sensor networks, in which predistributed keys have limited lifetimes. This is achieved by refreshing key pools for each generation of deployment. Refreshed key pools allow a network that is temporarily attacked to be self-healed in time.

If key pool is refreshed with random keys in each deployment, attacker cannot guess the upcoming pool by knowing previous keys or cannot learn former pools by knowing current one. However, in the same way, sensor nodes deployed at different generations cannot establish secure links. In order to achieve connectivity between nodes belonging to different generations, there should be some kind of relation between key pools at different generations.

RoK uses two key pools: *forward* and *backward key pools*, *FKP* and *BKP*. In order to provide connectivity between different generations, *FKP* is updated by hashing keys of previous generation and *BKP* is generated using Lamport hash chains [15].

Lamport hash chain is successive calculations of cryptographic hash function in order to produce one-time keys. Each key is derived from a hash function which takes next one-time key as an input.

Table 2.1 gives the symbols used in the explanations of RoK scheme. The same symbol table will be referred for the explanation of our proposed RGM scheme, which will be given in Section 5.

Table 2.1. Symbols used in RoK and RGM

FKP^j	Forward key pool at generation j in RoK scheme
BKP^j	Backward key pool at generation j in RoK scheme
GKP^j	Generation key pool of generation j in RGM scheme
P	Key pool size
FKR_A^j	Forward keyring of node A deployed at generation j in RoK scheme
BKR_A^j	Backward keyring of node A deployed at generation j in RoK scheme
KR_A^g	Generation keyring of node A deployed at generation g in RGM scheme
GKR_A^{fg}	Generation sub-keyring of node A deployed at generation f containing keys used to establish link with nodes deployed at generation g in RGM scheme. If $g > f$, then it is future generation sub-keyring. If $g = f$, then it is same generation sub-keyring.
fk_t^j	Forward key with index t at generation j in RoK scheme
bk_t^j	Backward key with index t at generation j in RoK scheme
gk_t^f	Generation key with index t to be used between the nodes deployed at the same generation f in RGM scheme
gk_t^{fg}	Generation key with index t to be used between the nodes deployed at generations f and g in RGM scheme
k_{AB}	Link key between nodes A and B
G_w	Generation window
$h(\cdot)$	Hash function that generates a non-repeating random number sequence to be used by RoK scheme. Each generated random number is in $[1...P]$ range.
$H(\cdot)$	Secure hash function
m	Number of current generation keys in a generation keyring
n	Number of future generation keys in a generation keyring for each next generation
p_{sg}	Key sharing probability of neighboring node pairs deployed at the same generation
p_{dg}	Key sharing probability of neighboring node pairs deployed at different generations

In RoK, the forward key pool at generation j is denoted as follows.

$FKP^j = \{fk_1^j, fk_2^j, ..., fk_P^j\}$, where P is the pool size. (2.1)

The forward key pool at generation $j+1$ is denoted as follows.

$FKP^{j+1} = \{fk_1^{j+1}, fk_2^{j+1}, ..., fk_P^{j+1}\}$, where $fk_t^{j+1} = H(fk_t^j)$. (2.2)

Similarly, the backward key pool at generation j is denoted as

$BKP^j = \{bk_1^j, bk_2^j, ..., bk_P^j\}$, where P is the pool size. (2.3)

The backward key pool at generation $j+1$ is denoted as follows.

$BKP^{j+1} = \{bk_1^{j+1}, bk_2^{j+1}, ..., bk_P^{j+1}\}$, where $bk_t^j = H(bk_t^{j+1})$. (2.4)

It is assumed that each node has an upper bound of lifetime and this upper bound defines generation window, G_w, which is a system parameter. A node may live at most as long as this generation window. A node A deployed at generation j is given two keyrings, forward keyring and backward keyring. Forward keyring, FKR_A^j, consists of forward keys of generation j drawn randomly from forward key pool at generation j, FKP^j. Backward keyring, BKR_A^j, consists of backward keys of generation $j+G_w-1$ drawn randomly from backward key pool at generation $j+G_w-1$, BKP^{j+G_w-1}. These keyrings are formally shown below.

$$FKR_A^j = \left\{ fk_u^j \mid u = h(id_A \| i \| j), i = 1,2,..., \lfloor m/2 \rfloor \right\} \quad (2.5)$$

$$BKR_A^j = \left\{ bk_u^{j+G_w-1} \mid u = h(id_A \| i \| j), i = 1,2,..., \lfloor m/2 \rfloor \right\} \quad (2.6)$$

Node A can produce a forward key fk_u^f if $f > j$ and backward key bk_u^b if $b < j+G_w-1$. Each node, deployed at generation j, have certain probability to share a common key with another node B which is deployed at generation i, where i is in interval $]j-G_w, j+G_w[$. The generations between which two nodes can produce the same forward and backward keys are called *overlapping generations*. Let's suppose $i \le j$, then their overlapping generations would be between j and $i+G_w-1$. If nodes A and B have common keys of indices $t_1, t_2, ..., t_z$, they compute their link key as the following.

$$k_{AB} = H(fk_{t_1}^j \| bk_{t_1}^{i+G_w-1} \| fk_{t_2}^j \| bk_{t_2}^{i+G_w-1} \| ... \| fk_{t_z}^j \| bk_{t_z}^{i+G_w-1}), \text{ where } i \le j. \quad (2.7)$$

Forward keys provide forward secrecy since the attacker cannot learn previous keys even if it learns a forward key at a generation. Similarly, backward keys provide

backward secrecy since the attacker cannot learn future keys even if it learns a backward key at a generation. When an attacker learns some forward and backward keys by capturing a sensor node, previous forward keys are not revealed since a forward key is calculated from previous forward key by a one-way hash function. Similarly, future backward keys are also protected. Sensor nodes cannot find out these previous forward keys and future backward keys even if they keep keys of same index in their keyrings. This property provides a lifetime to the keyring. The lifetime of a keyring also limits the capability of an attacker. He/she can use a compromised keyring for a short period of time. Since the keyrings have limited lifetime and key pools are refreshed periodically, compromised keys automatically expire like all the other keys as time passes. In this way, network gradually removes the traces of an attack and heals itself. If this attack is of temporary type, in a certain time network comes to the state before the attack has started. If it is a permanent type of an attack, network can keep the ratio of corrupted links within a certain limit.

There are a handful of works in the literature for key distribution in multiphase sensor networks that improves the RoK scheme [9]. Yilmaz et al. [16] leveraged generation time presented by RoK. They deployed nodes pre-loaded with keys belonging to future generations earlier; so that they reduced the period an adversary make use of compromised keys. Besides, there may be some discontinuity between the keyrings of sensor nodes as nodes of future generations are deployed earlier. Actually, they made a trade-off between resiliency and connectivity. Furthermore, Kalkan et al. [17] adjoined multiphase key predistribution scheme in RoK [9] scheme to deployment awareness discussed in [6]. They improved resiliency by leveraging deployment knowledge in key predistribution. Our RGM [1] scheme removed forward and backward key pools and reinvented a new key pool, which is called *generation key pool*. Unlike RoK scheme, there is no relationship between subsequent states of generation key pool. On the other hand, in RGM scheme, continuity between nodes at different generations is provided by giving each node keying material from key pools of other generations. Resiliency performance of RGM scheme is significantly better than RoK scheme with relatively small degradation in local connectivity.

3. A PROBABILISTIC KEY PREDISTRIBUTION SCHEME BASED ON XORED KEYING MATERIAL

In this section a random key predistribution scheme for wireless sensor networks is proposed. In this scheme, we use a novel key type called XORed key, which is bitwise XOR of two regular keys. In order to differentiate XORed and regular keys, we rename regular keys as single keys throughout this section. Our scheme uses a combination of XORed and single keys in the keyrings of sensor nodes. Since XORed keys are produced using two single keys, a secured link established using XORed keys is more resistant to attacks.

Our scheme also has a phase called transfer phase which increases the local connectivity of the network by transferring keys from secure neighbors in required conditions. Furthermore, transferred keys are not used directly, instead their XORed forms are used. In other words, keys are transferred to complete missing operands of XORed keys in the keyring.

As in Chan et al.'s scheme [7], our scheme uses two single keys contributed in the establishment of link key. However, in our scheme, keyrings of nodes are composed of variety of XORed and single keys, contrary to the homogeneous structure of keyrings in Chan et al.'s scheme [7]. Nodes XOR two single key operands or use XORed key in their keyring directly in order to secure their communications.

Table 3.1. List of symbols used in this section

SP	Key pool of single keys
XP	Key pool of XORed keys
xk_i	XORed keyring of node i
sk_i	Single keyring of node i
$\|SP\|$	Global single key pool size
$\|XP\|$	Global XORed key pool size
m	Keyring size
$\|sk\|$	Single keyring size
$\|xk\|$	XORed keyring size
N	Set of all nodes
NL_i	List of neighbors of node i
$SecureNL_i$	List of neighbors of node i with at least one shared key
n	Number of neighbors in communication range
$A \xrightarrow{key} B$	A sends *key* to B in a secure way

The rest of this section is organized as follows. Section 3.1 describes the proposed scheme. Section 3.2 explains performance metrics and gives comparative simulation results. The running times of the methods are given in Section 3.3. Final discussions and conclusions are made in Section 3.4.

3.1. Our Contribution

Our scheme is based on probabilistic key predistribution, like basic scheme [4]. It includes three main phases which are (*i*) key predistribution, (*ii*) shared-key discovery, and (*iii*) key transfer phases. Although key predistribution and shared-key discovery phases in our scheme have similar characteristics with corresponding phases of the basic scheme, our scheme differs from basic scheme with one important feature, which is the keying material. There are two types of keys stored in the keyrings of nodes.

Transfer phase is a novel phase in our scheme. In this phase, some manipulations are performed to improve the local connectivity of the proposed scheme.

Workflow of abovementioned phases are shown in Figure 3.1. Key predistribution phase and shared-key discovery phase are split into two parts as single key and XORed key subphases since the key types involved are different.

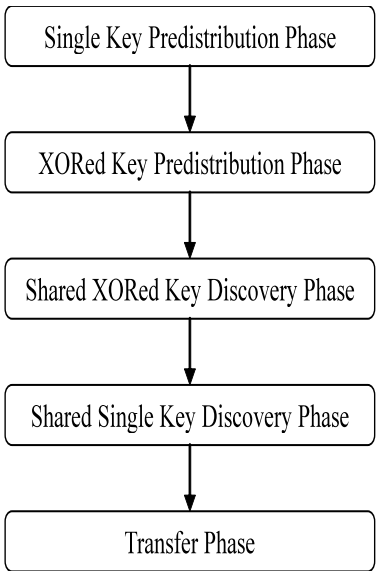

Figure 3.1. Workflow of the proposed scheme

3.1.1. Key Predistribution

Key predistribution phase starts with offline generation of a large single key pool of $|SP|$ keys. Each single key is assigned a unique key identifier. In addition to that, XORed key pool, which is composed of keys obtained by XORing distinct single keys in single key pool, is prepared automatically. The total number of XORed keys derived by XORing single keys is the number of binary combinations of all single keys in the single key pool. Thus, the size of the XORed key pool is the following:

$$|XP| = |SP| \times (|SP| - 1) / 2 \quad (3.1)$$

Key identifier of an XORed key x is defined as $i \,||\, j$, where i and j are key identifiers of corresponding single keys from which x is derived.

$|sk|$ single keys from *SP* are drawn uniformly random without replacement for each node to form single keyrings. After this process, XORed keyrings of sensor nodes are established by selecting $|xk|$ XORed keys from XORed key pool *XP* for each node uniformly random without replacement. If one of the single key operands of a selected XORed key is already in the single keyring of the node, selected XORed key is ignored and new one is drawn. Thus, the node can derive as many new XORed keys as possible from available single keys in its keyring. We will explain the reason why the nodes need to produce XORed keys from the single keys in their keyrings in shared-key discovery phase. Total keyring memory size of sensor nodes is the sum of single keyring size and XORed keyring size as given in Equation 3.2.

$$m = |sk| + |xk| \quad (3.2)$$

3.1.2. Shared-Key Discovery

After key predistribution phase is completed, sensor nodes with preloaded keyrings in memory are deployed in the field. After settlement, nodes scan their neighborhood independently. Each sensor node broadcasts node identity and the identifiers of single and XORed keys they have. At the end of transmission of own key identifiers and reception of respective keyring contents in the neighborhood, nodes try to find common XORed keys with their neighbor nodes in the first step. If no common XORed key with a neighbor is found, corresponding node tries to find a common single key.

The reason behind searching for XORed keys initially is that resiliency of an XORed key is more than a resiliency of a single key. An attacker has to obtain two operands of an XORed key in order to capture a link secured by that XORed key, while it is enough to capture just one single key in single link key case. In our scheme using XORed key as a link key is promoted; if there is a chance to use XORed key or single key to secure a link, XORed key usage is preferred. However, we should admit that it is harder to find out a shared XORed key in keyrings of neighboring nodes, because the size of XORed key pool is much larger than the size of single key pool. This reduces the probability of an XORed key to be shared by any two nodes.

If both of these attempts fail, nodes try to derive new XORed keys using single keys pre-loaded in their memory. A node XORs two single keys in its single keyring to match an XORed key in the XORed keyring of its neighbor. For example, suppose *a* and *b* are two neighbors with no secure link yet established. If node *a* has single keys *t* and *u* in its keyring such that $x = t \oplus u$ and *x* is in *b*'s XORed keyring, then *a* XORs *t* and *u* to obtain *x* and the nodes *a* and *b* can use *x* to establish a secure link. If neighboring nodes still cannot find a common key, they start key transfer phase which is described in the next subsection.

```
for all a∈N
  for all b∈NL_a
    for all x∈xk_b
      if x∈xk_a
        SecureNL_a = SecureNL_a ∪ {b}
        SecureNL_b = SecureNL_b ∪ {a}
      end
      if b∉SecureNL_a
        for all s∈sk_b
          if s∈sk_a
            SecureNL_a = SecureNL_a ∪ {b}
            SecureNL_b = SecureNL_b ∪ {a}
          end
for all a∈N
  for all b∈NL_a
    if b∉SecureNL_a
      for all t∈sk_a
        for all u∈sk_a
          if t⊕u∈xk_b
            SecureNL_a = SecureNL_a ∪ {b}
            SecureNL_b = SecureNL_b ∪ {a}
          end
```

Figure 3.2. Pseudo-code of shared-key discovery phase.

The algorithm of shared-key discovery phase is depicted in Figure 3.2. In this figure, it is shown that for each neighboring node *a* and *b*, if they have at least one shared XORed key *x* in their XORed keyring, they establish secure link by using this key. If they fail to find a shared XORed key, they search for a shared single key. If they find at least one shared single key *s*, they establish secure link by using this key. If a secure link is established in one of these ways, *a* and *b* add each other to their secure neighbor list.

In the second loop of the algorithm in Figure 3.2, neighboring nodes, which do not have any common XORed or single key, XOR single keys in their keyring to

produce an XORed key that is found in the keyrings of their neighbors. If there is a match, they use this XORed key in their communication.

3.1.3. Key Transfer

Main contribution of our scheme is in the key transfer phase. When shared-key discovery phase ends, there would possibly be some neighboring nodes that failed on finding common key and some succeeded on setting up secure communication links. Successfully established secure links are used to help creating secure links between insecure neighbor nodes in key transfer phase.

If two neighboring nodes a and b cannot establish a secure link in shared-key discovery phase, they apply a number of methods that will help them to agree on a key.

In the first method, a node searches its own single keyring and the single keyrings of its neighbors with which they have a secure communication link (from now on mentioned as direct secure neighbor). If there are two single keys, one in its single keyring and one in its secure neighbor's single keyring, such that XOR of these single keys matches an XORed key in its unsecure neighbor's XORed keyring, it transfers the single key from its direct secure neighbor and establishes communication with its unsecure neighbor using XOR of those single keys. For example, suppose b has the XORed key $x = s \oplus t$ in its XORed keyring xk_b, and one of its operands t is found in node a, the other operand s is found in direct secure neighbor node c of a. In this method, a transfers the single key s from node c and XORs it with its single key t. In this way, it obtains a new XORed key x which a shares with b. For this operation, node a needs to seek the identifiers of the single keys in direct secure neighbors' single keyrings. These key identifiers are already supplied in shared-key discovery phase, so there is no need to resend them in this phase. Since there is no extra messages sent

through nodes, this method (and also the upcoming methods) has minimal effect on communication cost. Algorithm for this method is given in Figure 3.3.

```
Method 1 (a, b):
for all c∈SecureNL_a
  for all s∈sk_c
    for all t∈sk_a
      if s⊕t∈xk_b
        c --s--> a
        sk_a = sk_a ∪ {s}
        SecureNL_a = SecureNL_a ∪ {b}
        SecureNL_b = SecureNL_b ∪ {a}
      end
```

Figure 3.3. Method 1, nodes try to transfer single keys from their direct secure neighbors and XOR them with existing single keys in their keyring to produce an XORed key that is found in the keyrings of their neighbors.

If the first method does not work, the nodes use the second method. In method 2, a node transfers an XORed key from its direct secure neighbor, and XOR the transferred key with another XORed key in its keyring to derive a new XORed key that can be used for securing a communication link. In order to derive a new XORed key, transferred XORed key and existing XORed key in the keyring should have one common operand; when two XORed keys are XORed, this common operand cancels out and new XORed key becomes XOR of two single keys. For instance, suppose nodes a and b are two neighboring nodes with no shared key and node c is a's direct secure neighbor. c has an XORed key x and a has an XORed key y in their XORed keyrings such that x and y have a common single key operand. Furthermore, XOR of x and y produces another XORed key which is found in the XORed keyring of node b. In this condition, a transfers x from c and creates XOR of x and y. Afterwards nodes a and b establish secure link using this XORed key. Algorithm for method 2 is given in Figure 3.4.

Method 2 (a, b):
for all $c \in SecureNL_a$
  for all $x \in xk_c$
    for all $y \in xk_a$
      if $x \oplus y \in xk_b$
        $c \xrightarrow{x} a$
        $xk_a = xk_a \cup \{x\}$
        $SecureNL_a = SecureNL_a \cup \{b\}$
        $SecureNL_b = SecureNL_b \cup \{a\}$
      end

Figure 3.4. Method 2, nodes try to transfer XORed keys from their direct secure neighbors and XOR them with existing XORed keys in their keyring also to produce an XORed key that is found in the keyrings of their neighbors.

If the previous methods do not help to establish a secure link, as the last chance, the nodes try method 3. In this method, a node searches for two single keys from distinct direct secure neighbors, transfer and XOR them to obtain a new XORed key. If this new XORed key is also found in the XORed keyring of the neighbor node with which the node wants to establish a secure link, then this secure link is created. Transferred XORed keys should have one common operand again for the reasons mentioned above. For example, suppose nodes c and d are direct secure neighbor of a and a and b are two neighboring nodes that do not have a shared key. c has a single key s and d has a single key t such that their XOR produces a new XORed key which also exists in the node b's XORed keyring. a requests to transfer single keys s and t from direct secure neighbors c and d and upon receipt a XORs these two single keys and produces the XORed key which is common with b. Then, they establish secure link. Algorithm of this method is given Figure 3.5.

Method 3 (a, b):
for all $c \in SecureNL_a$
  for all $d \in SecureNL_a$
    for all $s \in sk_c$
      for all $t \in sk_d$
        if $s \oplus t \in xk_b$
          $c \xrightarrow{s} a$
          $sk_a = sk_a \cup \{s\}$
          $d \xrightarrow{t} a$
          $sk_a = sk_a \cup \{t\}$
          $SecureNL_a = SecureNL_a \cup \{b\}$
          $SecureNL_b = SecureNL_b \cup \{a\}$
        end

Figure 3.5. Method 3, nodes try to transfer two single keys from two distinct direct secure neighbors and XOR them in order to produce an XORed key that is found in the keyrings of their neighbors.

If a node runs these methods in the transfer phase and establishes new secure communication links with neighbor nodes, it adds these nodes into its direct neighbors list and to re-runs these methods for the purpose of deriving new common keys with the neighbors that do not share one. This provides an additional optimization to transfer phase, and it incrementally improves local connectivity.

3.2. Performance Evaluations

For the performance evaluation of the proposed scheme, we conduct simulations. The simulations are done in MATLAB environment in a 2.4 GHz Intel Core 2 Quad desktop PC running 32-bit Windows Vista and 64-bit Linux operating systems. In simulations, 10000 nodes are uniformly random distributed in a square field of size 1000 m × 1000 m. Nodes can communicate with others in an area of 40 m radius. In

this configuration, average number of neighbors becomes 48.64 nodes. Single key pool size is 10000 keys. Thus, XORed key pool size automatically becomes 49,995,000 keys, although this key pool is not actually generated.

Local connectivity is the probability that any two sensor nodes share at least one key. In Figure 3.6, basic scheme line indicates the local connectivity of basic scheme [4] with 100 single keys. This line serves as a performance threshold to our scheme. For our scheme, single keyring sizes increase by 10 keys, while XORed keyring sizes decrease by 10 keys such that single keyring and XORed keyring sizes add up to 100 keys. When single keyring size is small, local connectivity of our scheme is lower than the basic scheme. The reason is that probability of sharing at least one XORed key is so small as the XORed key pool is huge in practice. As the single keyring size increases and XORed keyring size reduces, the local connectivity of our scheme increases and eventually exceeds the basic scheme threshold. The highest connectivity of our scheme occurs when 70 single keys and 30 XORed keys are used. At this point, local connectivity value of our scheme is 0.97; whereas the local connectivity of basic scheme with 100 keys is 0.63. This result shows that our scheme provides almost full connectivity when the keyring size ratio is selected appropriately.

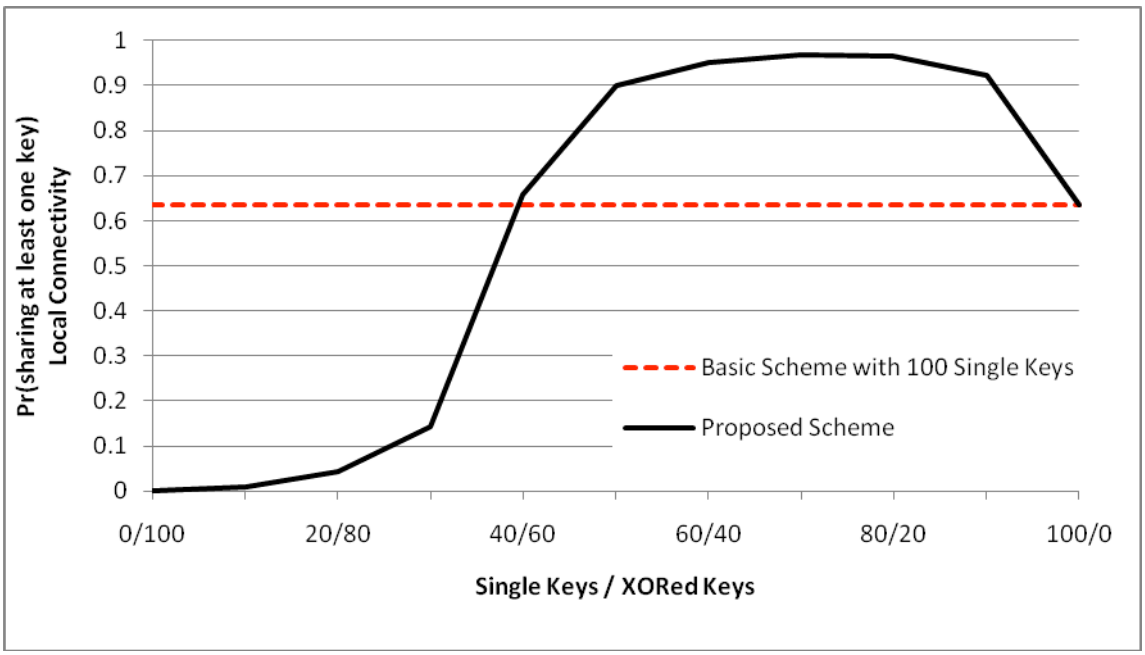

Figure 3.6. Local connectivity of our scheme compared to basic scheme

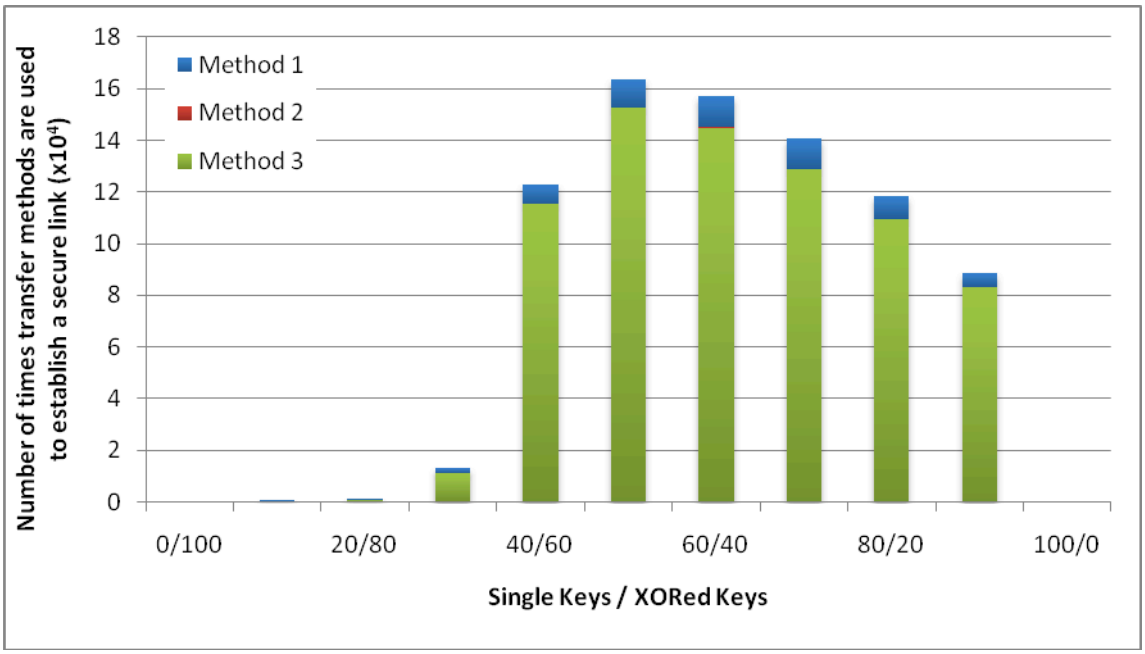

Figure 3.7. Number of times the nodes make use of transfer methods in transfer phase as a whole (units in vertical axis are multiples of 10^4).

Figure 3.7 is the graph of number of times transfer methods are used to establish a secure link versus number of keys in single and XORed keyrings of the nodes. Units in the vertical axis are multiples of 10^4. As shown in Figure 3.7, the greatest benefit of our key transfer phase is obtained when there are 50 single keys and 50 XORed keys in corresponding keyrings. In this distribution, more than 160,000 links are established securely by our transfer methods. In the figure it is seen that method 3 is dominant amongst others, because method 3 gives nodes great flexibility to produce new XORed keys. The other methods somehow depend on the content of the node's own keyring; however, method 3 gives nodes the chance to use the keyrings of two distinct direct secure neighbors. The second mostly used method is method 1. Method 2 is very similar to method 1 except for transferring XORed keys instead of single keys. This difference is the reason of the gap between the usage numbers of these methods, because it is harder to find appropriate XORed keys, as the XORed key pool size is much larger.

Transfer cost defines the number of keys exchanged between sensor nodes in order to establish link keys in transfer phase. Transferred keys are usually the missing operands of XORed link keys. In method 1 and method 2, one key (single key for method 1 and XORed key for method 2) is transferred from direct secure neighbors in

order to produce a new XORed link key. Meanwhile in method 3, two single keys are transferred from two distinct direct secure neighbors. Here one should notice that keys are transferred whenever needed to establish a new secure link. In other words, they are transferred deterministically, not probabilistically. The number of keys transferred between secure direct neighbors is 269,210 for the case that 70 single keys and 30 XORed keys are used which gives the highest local connectivity. For this case, 1.915 keys are transferred on the average for each new established links during transfer phase.

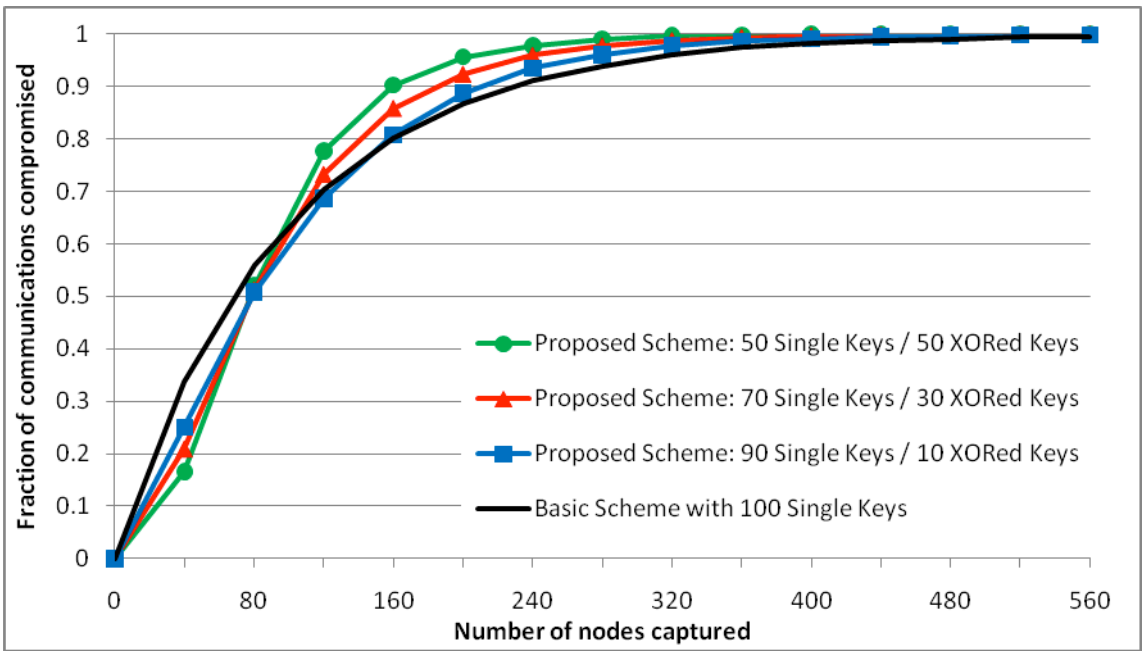

Figure 3.8. Resilience of our scheme compared to basic scheme.

Figure 3.8 shows the resilience of our scheme compared to basic scheme. Resilience is the fraction of communication links compromised to all links at the end of a node capture attack. Lower values indicates more resilient network. In the figure, it is seen that as the fraction of XORed keys increase, the network becomes more resilient at the beginning of an attack. However, as the attack continues, basic scheme becomes slightly more resilient starting from the point where 0.6 fractions of links are compromised. Nevertheless when the optimal case of our scheme, which is the case 70 single keys and 30 XORed keys are used, is compared to the basic scheme, the difference in ratio of links compromised is always lower than 5%. This means our scheme provides better connectivity with a marginal cost of resilience.

3.3. Complexity Analysis

In this section, we analyze the running times of the methods that take place in shared-key discovery phase and transfer phase of our proposed scheme. In the second part of shared-key discovery phase, nodes search their single keyrings to find two single keys that match an XORed key in the keyring of their unsecure neighbor when XORed. The complexity of this algorithm is $O\left(n \times |sk|^2 \log(|xk|)\right)$, where n is the number of neighbor nodes in communication range, $|sk|$ is the single keyring size and $|xk|$ is the XORed keyring size.

In the first method of transfer phase, nodes search the keys in the single keyrings of their secure neighbors and the keys in their single keyrings to find a match that XOR of those single keys is in the XORed keyring of their unsecure neighbor. They repeat this method for each unsecure neighbor. Hence, the complexity is $O\left(n^2 \times |sk|^2 \log(|xk|)\right)$.

In the second method, nodes search the XORed keys in the keyrings of their secure neighbors and the XORed keys in their keyrings to find a combination that XORing of those XORed keys produces another XORed key that is in the XORed keyring of their unsecure neighbor. They run the method for each unsecure neighbor. Thus, the complexity of this method is $O\left(n^2 \times |xk|^2 \log(|xk|)\right)$.

Nodes search the single keyrings of each pair of their secure neighbors to find two single keys that matches an XORed key in the keyring of their unsecure neighbor when they are XORed. Nodes iterate this method for each of their unsecure neighbors. Thereby, the complexity of the last method is $O\left(n^3 \times |xk|^2 \log(|xk|)\right)$, since three different neighbor nodes take place.

3.4. Discussions and Conclusions

All of the methods used in the transfer phase aim to set up new secure communication links. However, none of them directly use a transferred key, but modify them by performing XOR operations. This property improves resilience such that if a direct neighbor, from which a node transferred a key, is captured by an attacker, the node's communication link may still remain safe. If transferred keys were used directly, then capture of one node would mean compromise of all link keys that are transferred from this capture node.

Our scheme cannot be directly implemented into Du et al.'s scheme [5], because it uses matrices for distributing keys. But if we degrade λ-secure property to 0-secure property, which means basic scheme, then we can easily engage deployment knowledge into our scheme together with mentioned modifications to basic scheme above.

We proposed a scheme that achieves high connectivity values as compared to basic scheme with same keyring sizes. In addition to this, our scheme uses XORed keys for link keys, if available. This plays a role to increase the resilience of the network, since an attacker needs to compromise two single keys instead of one. Our scheme uses a novel phase called transfer phase, in which the nodes transfer single or XORed keys directly from their secure neighbors. Our scheme achieves high local connectivity with carefully selected number of single and XORed keyring sizes. In our proposed scheme with 70 single keyring size and 30 XORed keyring size, there is an increase in local connectivity more than 50% as compared to basic scheme [4]. The cost of this improvement in local connectivity to the resilience is also minimal. The difference between resilience of our proposed scheme and basic scheme does not exceed 5%. Communication cost is minimized since keyring indices of neighbor nodes are already sent out in shared-key discovery phase. Nodes transfer only required keys from their neighbors and no pre-discovery of keys is needed during this phase.

4. IMPROVING CONNECTIVITY OF KEY PREDISTRIBUTION VIA TRANSFERRED KEYS

In this section of the thesis, we propose a scheme that is an extension to basic scheme [4]. Our scheme is composed of three phases. These are (*i*) key predistribution, (*ii*) shared-key discovery phase, and (*iii*) transfer phase. First two phases are identical to the corresponding phases of the basic scheme. As a distinct feature, the third phase of our scheme, which is the transfer phase, is different from the path-key establishment phase of the basic scheme.

In transfer phase, missing keys are asked from the secure neighbors in order to establish more secure links. As the indices of keys in the keyrings are sent out in shared-key discovery phase, there is no need to an extra effort to learn the keyring contents of the neighbors to find out which of them have necessary keys.

The method in the transfer phase of this scheme is tried in the XORed keying material scheme, which is proposed in Section 3, in the first place. Though, this method suppressed other methods used in XORed keying material scheme. Consequently, we removed the method from XORed keying material scheme and considered it as an independent scheme.

The rest of this section is organized as follows. Our contribution is explained in Section 4.1. Section 4.2 discusses the analytical computations of the performance of our scheme and shows comparative performance evaluation. Discussions and conclusions of this scheme are given in Section 4.3.

Table 4.1 Symbols used in this section

A_f	Area of the sensor field
P	Global key pool size
m	Keyring size
m'	Extended keyring size
k	Number of neighbors in communication range
d	Number of neighbors with at least one shared key
p	Probability of two nodes share at least one key, i.e. local connectivity after shared-key discovery phase
p'	Connectivity after transfer phase
N	Set of all nodes
NL_i	Neighbor list of node i
Kr_i	Keyring of node i
$a \xrightarrow{key} b$	a sends *key* to b in a secure way
r	Radius of a node's communication range
t	The average distance between any two neighboring nodes
$A(x, y)$	The union area of the communication ranges of two neighboring nodes located at positions (0, 0) and (x, y)
$n(x, y)$	The total number of the nodes that are neighbors to nodes a and/or b which are located at positions (0, 0) and (x, y)
$f_{xy}(x, y)$	The probability distribution function of the position of node b with respect to its neighboring node a located at position (0, 0)

4.1. Our Contribution

Our solution is taking advantage of previously secured communication links between neighboring nodes. As it is known, unique identifiers of keys in the keyrings of all neighboring sensor nodes are sent out in shared-key discovery phase. This data can be kept in the memory of sensor nodes for the purpose of searching keyring content of neighbors while needed. If two neighboring sensor nodes a and b cannot find any common key to secure their communication link, they can individually search the keyring data of their neighbor nodes with which they already established secure communication. If one of the secure neighbors of a has already a shared key with b, node a asks that secure neighbor to transfer this shared key. After that, nodes a and b

establish a secure link using this key. Same situation is valid for node *b* also. For example, suppose node *c* is a neighbor of *a* and it shares a key with *a*, meaning that they have secured their communication in shared-key discovery phase. Moreover suppose that *c* and *b* have a common key, say *K*, in their keyrings. Since *a* knows the key indices of the keyrings of both *c* and *b*, *a* concludes that *c* can help to establish a secure link between *a* and *b*. To do so, *a* requests *K* from *c* and *c* transfers this shared key *K* to *a* via their secure link. Then the neighboring nodes *a* and *b* can use *K* to secure their communication. This algorithm is expressed in Figure 4.1 and it is also visualized in Figure 4.2.

```
for all a∈N
  for all b∈NL_a
    if Kr_a ∩ Kr_b = ∅
      for all c∈NL_a s.t. Kr_a ∩ Kr_c ≠ ∅
        for all key∈Kr_c
          if key∈Kr_b
            c --key--> a
            Kr(a) = Kr(a) ∪ {key}
          end
```

Figure 4.1. Pseudo-code of transfer phase

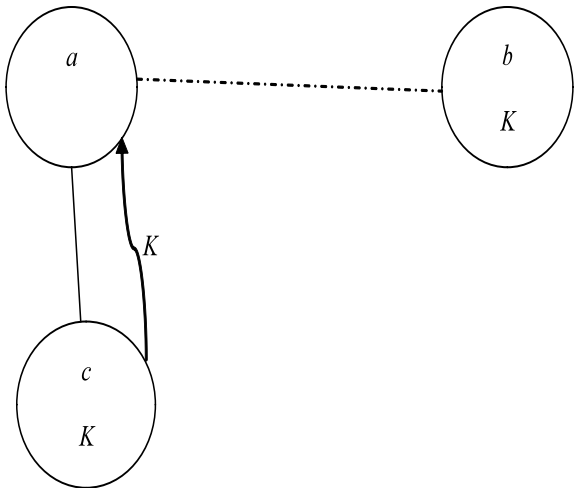

Figure 4.2. Visualization of transfer phase

This improvement to the basic scheme makes a situation as if the keyring size of a sensor node is the amount of unique keys in its keyring and in the keyrings of its neighbor nodes with a secured link. This keyring is called *extended keyring* and its size, m', has boundary conditions as specified below.

$$m \le m' \le (d+1) \times m \tag{4.1}$$

In Equation 4.1, $(d+1) \times m$ is the maximum value for m'. This condition occurs when all the keys in the keyrings of d secure neighbors and node's own keyring are unique. m is the minimum value for m'. This condition occurs when all the nodes in the neighborhood have the same keyring with the node's own keyring. On the other hand, m' is only the theoretical extended keyring size, not the actual one. Actual extended keyring includes only the predistributed keys in the node's keyring and keys actually downloaded via secure channels during the transfer phase. As a result, if a sensor node is captured by an attacker, he/she can obtain only this amount of keys.

Our scheme has significant advantages over basic scheme [4] with path-key establishment phase. In path-key establishment phase, a node has to find a secure path from itself to its unsecure neighbor in order to generate a link key. This is not handled intuitively, it broadcasts its secure neighbors if they know a path to its unsecure neighbor and its secure neighbors also broadcast this message to their corresponding neighbors and so on. Hence, this leads to a message flooding in the network. In our scheme there is no need to find that kind of path between unsecure neighbors. Moreover, the keyring contents of neighboring nodes are already discovered in shared-key discovery phase. Therefore, there is no significant communication cost during transfer phase in our scheme.

4.2. Performance Evaluation

Further analysis of performance metrics are given in this section. We compared local connectivity and resilience performance of our scheme to the basic scheme after shared-key discovery and path-key establishment phases in the second part of this section. Communication cost analysis of transfer phase of our scheme and path-key establishment phase of basic scheme are also given in this part. Comparison of the scheme based on XORed keying material, which is proposed in Section 3, and the scheme proposed in this section is made in the third part of this section.

4.2.1. Analytical Formulations

In this section, we give analytical formulation for the local connectivity of our scheme. We will use these formulations to support simulation results which will be given in the next subsection. Before giving the local connectivity calculations, we need to find the total number of sensor nodes that are neighbors to one or both of two neighboring nodes *a* and *b*.

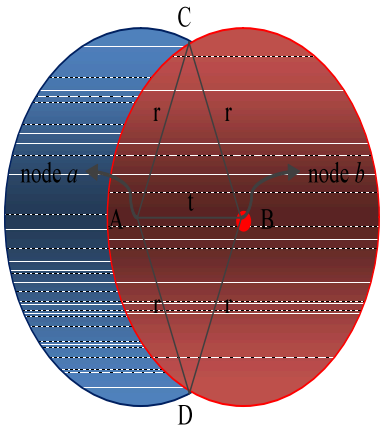

Figure 4.3 The coverage areas of two neighboring nodes *a* and *b*

We try to find the expected value of the colored area, which is the union of the communication ranges of neighboring nodes *a* and *b*, shown in Figure 4.3. In this figure, *r* is the radius of the communication range and *t* is the average distance between nodes *a* and *b*.

Area of *ABCD* rhombus is formulated in Equation 4.2. Area of the union of coverage areas of nodes *a* and *b*, which is the overall colored region in Figure 4.3, is the difference between the sum and the intersection of two coverage areas. The intersection region is the darker zone in Figure 4.3 and it is calculated as the subtraction of the area of *ABCD* rhombus from the areas of *CAD* and *CBD* sectors. Area of the union region is formulated in Equation 4.3.

$$Area_{ABCD} = r \times t \times \cos\left(\arcsin\left(\frac{t}{2r}\right)\right) \tag{4.2}$$

$$\begin{aligned} Area_{union} &= 2\pi r^2 - Area_{intersection} \\ &= 2\pi r^2 - \left(2 \times \arccos\left(\frac{t}{2r}\right) \times r^2 - Area_{ABCD}\right) \\ &= 2\pi r^2 - \left(2 \times \arccos\left(\frac{t}{2r}\right) \times r^2 - r \times t \times \cos\left(\arcsin\left(\frac{t}{2r}\right)\right)\right) \\ &= 2\pi r^2 - 2r^2 \times \arccos\left(\frac{t}{2r}\right) + r \times t \times \cos\left(\arcsin\left(\frac{t}{2r}\right)\right) \end{aligned} \tag{4.3}$$

The position of a neighboring node *b*, which is in the communication range of another node *a*, is assumed to be a bivariate uniformly distributed random variable with the following probability distribution function.

$$f_{xy}(x, y) = \frac{1}{\pi r^2} \tag{4.4}$$

Without loss of generality, we can assume that *a* is located at (0, 0) position. Therefore, the distance between *a* and *b,* namely *t*, is formulated as follows.

$$t = \sqrt{x^2 + y^2} \tag{4.5}$$

By substituting Equation 4.5 in 4.3, we obtain Equation 4.6.

$$Area_{union} = A(x,y)$$
$$= 2\pi r^2 - 2r^2 \times \arccos\left(\frac{\sqrt{x^2+y^2}}{2r}\right) + r \times \sqrt{x^2+y^2} \times \cos\left(\arcsin\left(\frac{\sqrt{x^2+y^2}}{2r}\right)\right) \quad (4.6)$$

The expected value of $A(x,y)$ is calculated in Equation 4.7.

$$E(A(x,y)) = \iint_R A(x,y) f_{xy}(x,y)\,dxdy$$
$$= \iint_R \left(\left(2\pi r^2 - 2r^2 \times \arccos\left(\frac{\sqrt{x^2+y^2}}{2r}\right) + r \times \sqrt{x^2+y^2} \times \cos\left(\arcsin\left(\frac{\sqrt{x^2+y^2}}{2r}\right)\right)\right) \times \frac{1}{\pi r^2} \right) dxdy$$
$$= \iint_R \left(2 - \frac{2}{\pi} \times \arccos\left(\frac{\sqrt{x^2+y^2}}{2r}\right) + \frac{\sqrt{x^2+y^2}}{\pi r} \times \cos\left(\arcsin\left(\frac{\sqrt{x^2+y^2}}{2r}\right)\right)\right) dxdy \quad (4.7)$$

where R is the circular coverage area of a node as shown in Figure 4.4.

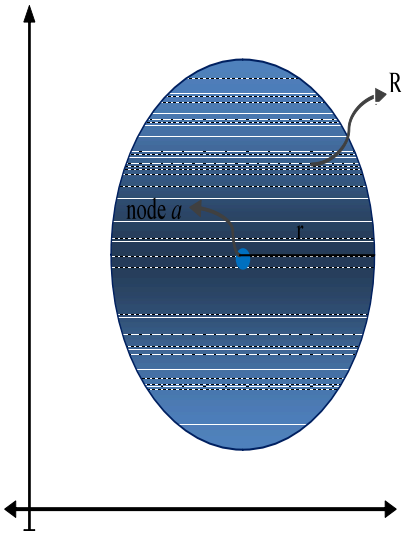

Figure 4.4 The coverage area of a node *a*

The total number of the nodes that are neighbors to nodes *a* and/or *b* is calculated as the total number of nodes in the intersection area minus 2 (for *a* and *b*). More formally, the total number of neighbors of *a* and/or *b* is calculated in Equation 4.8.

$$n(x,y) = A(x,y) \times \frac{N}{A_f} - 2 \tag{4.8}$$

Expected value of $n(x,y)$ is the following.

$$E(n(x,y)) = \iint_R \left(A(x,y) \times \frac{N}{A_f} - 2 \right) f_{xy}(x,y) dxdy \tag{4.9}$$

where R is the circular coverage area of a node as shown in Figure 4.4.

When we substitute Equation 4.7 in Equation 4.9, we obtain the following equation.

$$E(n(x,y)) = \\ -2 + \frac{N}{A_f} \times \iint_R \left(2 - \frac{2}{\pi} \times \arccos\left(\frac{\sqrt{x^2+y^2}}{2r} \right) + \frac{\sqrt{x^2+y^2}}{\pi r} \times \cos\left(\arcsin\left(\frac{\sqrt{x^2+y^2}}{2r} \right) \right) \right) dxdy \tag{4.10}$$

Local connectivity formulation of the basic scheme has been given in [4]. The probability that two nodes do not share any key after shared-key discovery phase, in other words their keyrings are composed of completely different keys, is given as follows.

$$p[\text{two nodes do not share any key}] = \frac{\binom{P}{2m}\binom{2m}{m}}{\binom{P}{m}^2} \tag{4.11}$$

Local connectivity is the probability that any two nodes share at least one key in their keyrings. This can be found as the following.

$$p = 1 - p[\text{two nodes do not share any key}] = 1 - \frac{\binom{P}{2m}\binom{2m}{m}}{\binom{P}{m}^2} \tag{4.12}$$

In our transfer phase, neighboring nodes *a* and *b* individually try to find a secure neighbor that has a common key with the other one. If one of them succeeds, then a secure link between *a* and *b* is established. The candidate node that can help *a* and *b* in this process: (*i*) must be a secure neighbor of at least one of *a* and *b*, (*ii*) must have common key(s) in its keyring with both *a* and *b*. If the candidate node is in the union of coverage areas of *a* and *b* and has common keyring element with both, then it is already a secure neighbor of at least one of them. Therefore, the requirements expected from the candidate node is reduced to being in the union of the coverage areas of *a* and *b* and having common element in its keyring that is also in the keyring of *a* and common element in its keyring that is also in the keyring of *b*.

The probability that a node has at least one common key with both *a* and *b* is p^2. Therefore, the probability that this node does not have a common key with *a* and/or *b* is $1-p^2$. There are on the average of $E(n(x,y))$ nodes with independent keyrings in the union area. The probability that none of those nodes can help to establish a secure link between *a* and *b* in the transfer phase is $(1-p^2)^{E(n(x,y))}$. Multiplying this value with the failure probability after shared key discovery phase yields overall failure probability of $(1-p^2)^{E(n(x,y))}\times(1-p)$. Therefore the probability that *a* and *b* establish a secure link after the transfer phase, p', is given as follows.

$$p' = 1-(1-p^2)^{E(n(x,y))}\times(1-p) \tag{4.13}$$

4.2.2. Simulation Results

We performed simulations to analyze the performance of our scheme and compare it with existing schemes. Results of our scheme are compared with results of basic scheme [4] and basic scheme with one-hop path-key establishment phase. Basic scheme is composed of two fundamental phases. These phases are key predistribution phase and

shared-key discovery phase. One-hop path-key establishment phase is an add-on performed after shared-key discovery phase. Our transfer phase is another add-on, which is alternative to one-hop path-key establishment phase. It is performed after shared-key discovery phase.

Simulation code is written using MATLAB language in Windows 32-bit operating system. In our simulations, 10000 nodes are distributed over a field of 1000 m × 1000 m. Nodes have a communication range of 40 m. In this setting, the expected value of the number of nodes in the union area of the coverage areas of two neighboring nodes is calculated as 69.05 using Equation 4.10. Two sets of simulations are performed for two different key pool sizes. In the first set, key pool contains 10,000 unique keys and in the second set, key pool contains 100,000 unique keys.

In Figures 4.5 and 4.6, local connectivity values of pure basic scheme (i.e. shared-key discovery phase), one-hop path-key establishment phase and transfer phase are compared. Horizontal axis indicates the keyring size of nodes, key pool size is set to 10,000 in Figure 4.5 and 100,000 in Figure 4.6. In Figure 4.5, it is seen that basic scheme with shared-key discovery phase has ~1.0 local connectivity[1] when 220 keys are preloaded to nodes' keyrings. Path-key establishment phase raises local connectivity to ~1.0 with keyring size of 75 keys. Our contribution, transfer phase, does the same with 55 keys in the keyring. Transfer phase gives the same local connectivity performance with one fourth of keyring size as compared to basic scheme with shared-key discovery phase.

[1] ~1.0 local connectivity means having a local connectivity value >0.99 in the rest of this section.

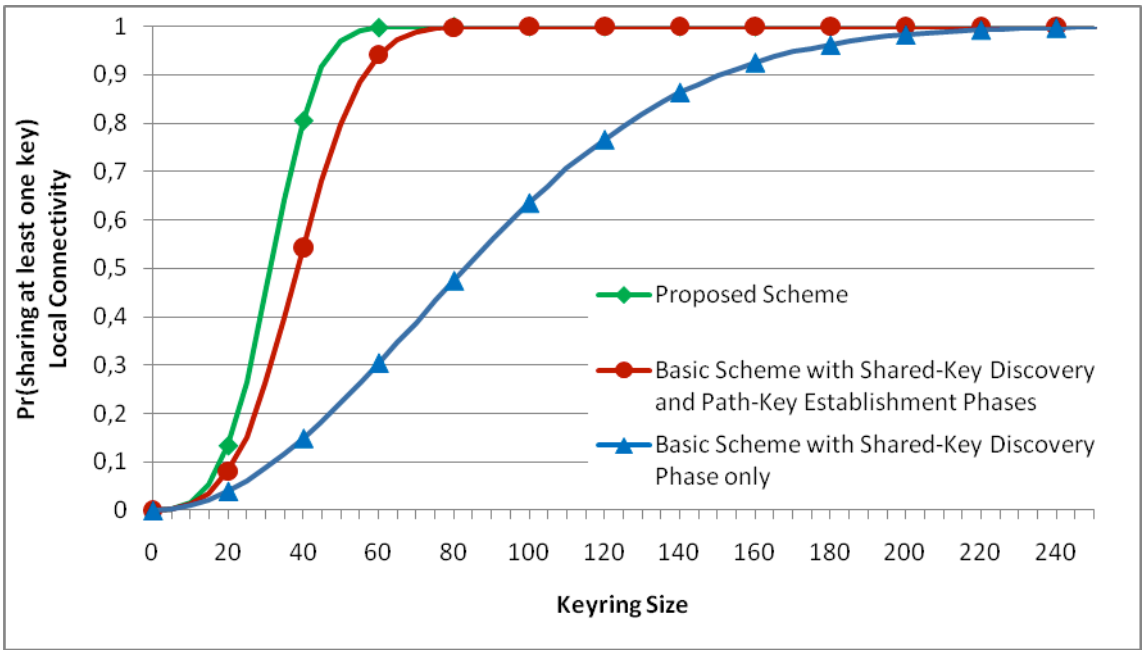

Figure 4.5. Comparison of local connectivities of basic scheme with shared-key discovery phase, basic scheme with path-key establishment phase and proposed scheme. Key pool size is 10,000.

In Figure 4.6, local connectivity value of basic scheme with shared-key discovery phase approaches 0.5 when the keyring size is 250 with 100,000 key pool size setting. ~1.0 local connectivity is achieved by basic scheme with path-key establishment phase when the keyring size is 230, rather, the same connectivity is achieved by our proposed scheme when the keyring size is 180.

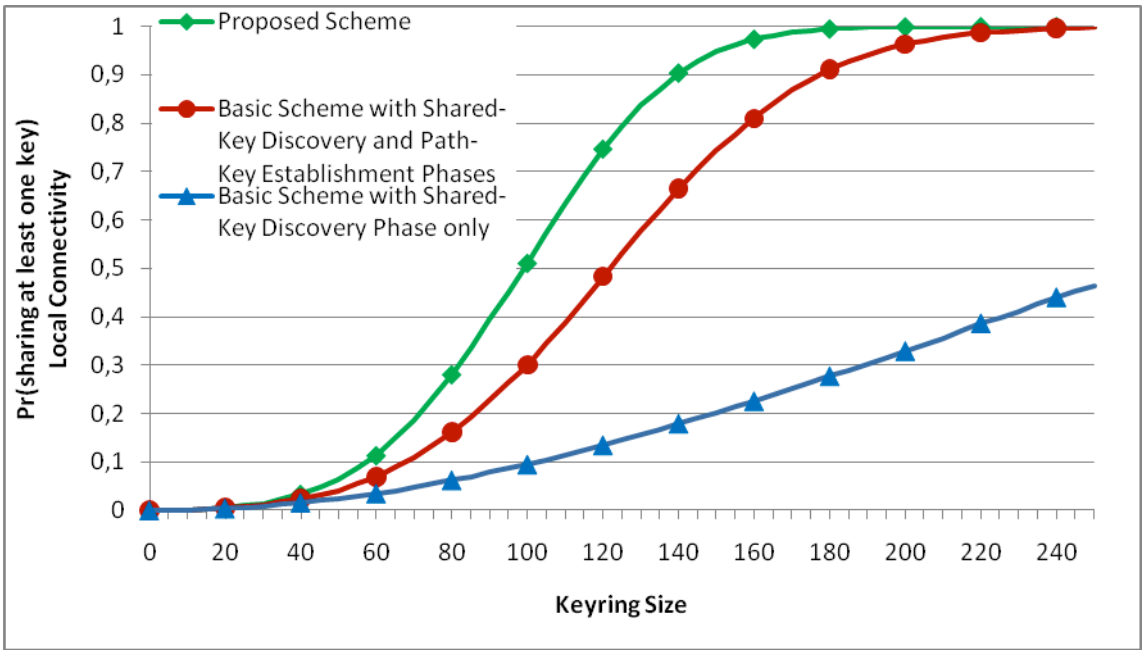

Figure 4.6. Comparison of local connectivities of basic scheme with shared-key discovery phase, basic scheme with path-key establishment phase and proposed scheme. Key pool size is 100,000.

Analytical formulations of local connectivity of transfer phase, which are given in Section 4.2.1, are compared with simulation results in Figures 4.7 and 4.8. We apply Equations 4.10 and 4.13 to calculate local connectivity. As can be seen from Figures 4.7 and 4.8 with key pool sizes of 10,000 and 100,000 respectively, simulation and analytical results definitely match each other. This verifies the correctness of our simulations.

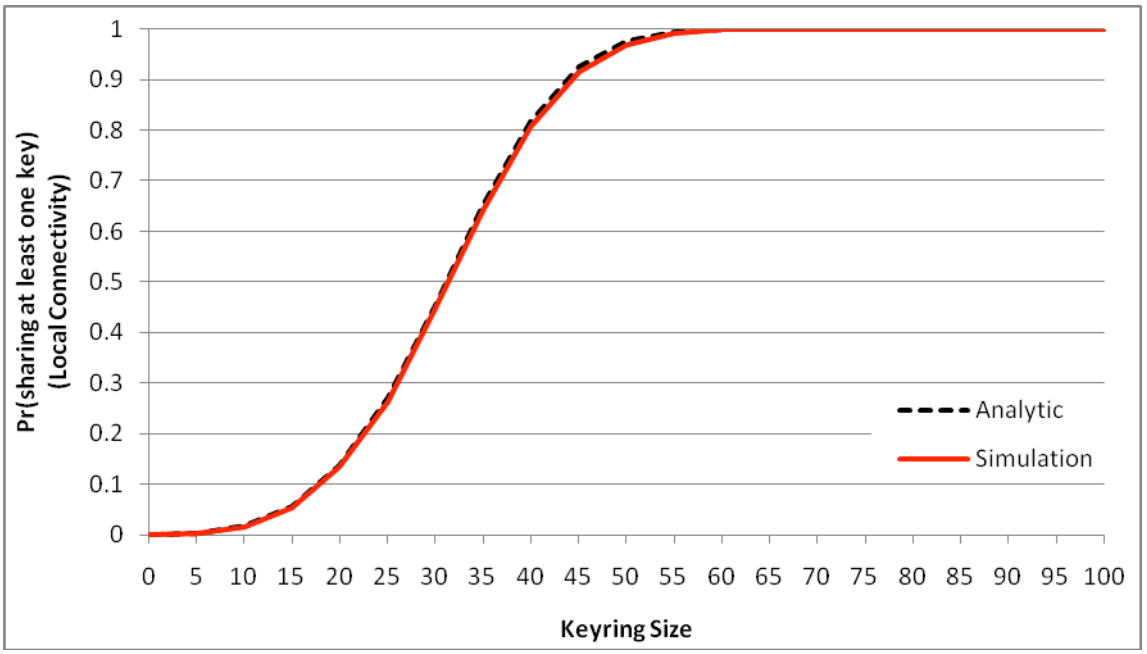

Figure 4.7. Comparison of simulative and analytical local connectivity of our scheme. Key pool size is 10,000.

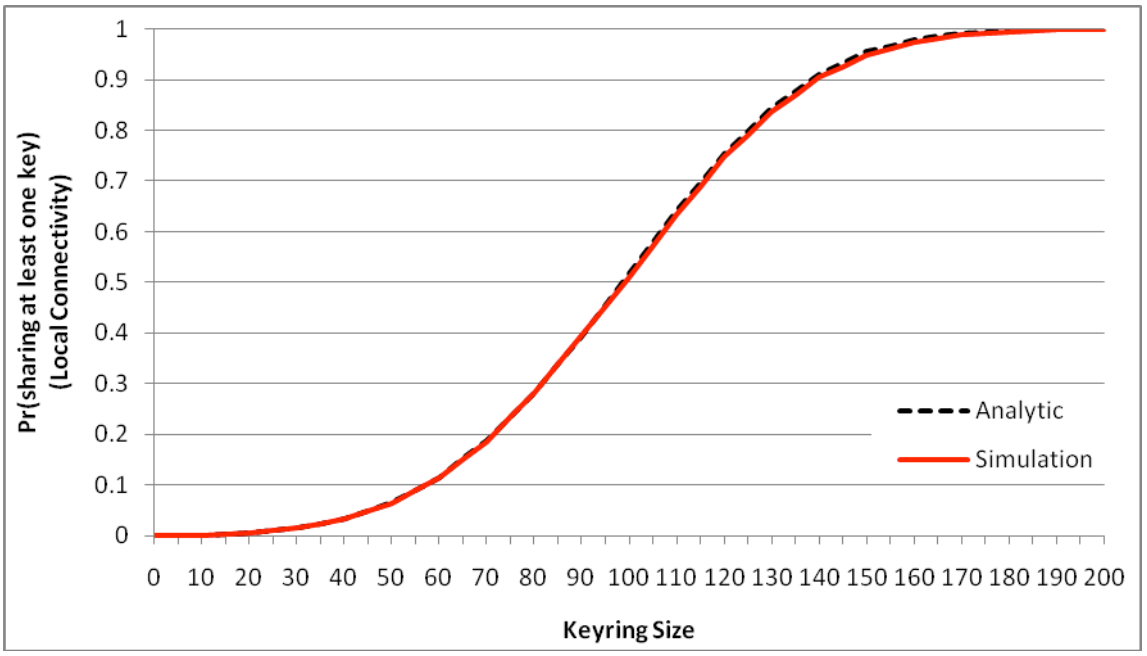

Figure 4.8. Comparison of simulative and analytical local connectivity of our scheme. Key pool size is 100,000.

Figures 4.9, 4.10, and 4.11 show comparative resilience of basic scheme with shared-key discovery phase and our proposed scheme for different key pool sizes.

Resilience is measured as the fraction of indirectly compromised links to all established links. Smaller this ratio is better the resilience. In a network with key pool size of 10,000 keys, shared-key discovery phase with 220 keys preloaded in keyrings of each node has a local connectivity of value ~1.0. The same local connectivity value can be achieved with the transfer phase with only 55 keys preloaded in nodes' keyrings. As can be seen from Figure 4.9, our scheme with transfer phase performs much better than basic scheme in this setting. Resiliency performance of our scheme is up to 35% better than basic scheme with shared-key discovery phase when the number of nodes captured is 120.

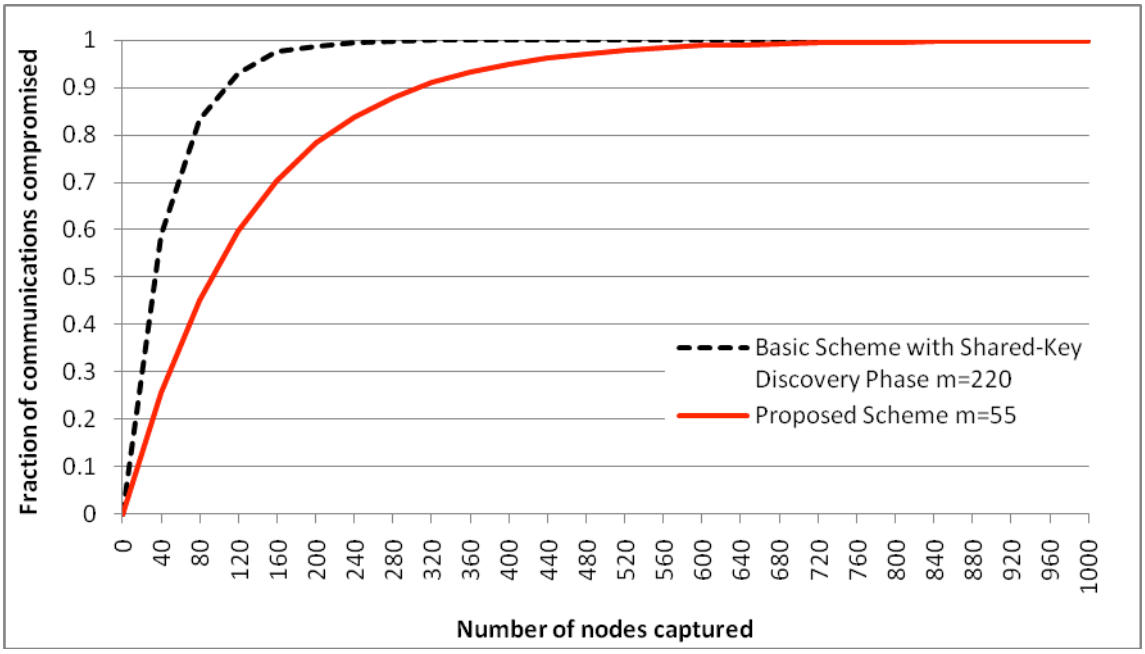

Figure 4.9. Fraction of communications compromised vs. number of nodes captured by the attacker for basic scheme with shared-key discovery phase and our proposed scheme. Local connectivity is set to ~1.0 for both cases, key pool size is 10,000.

In Figure 4.10, local connectivity value is set to 0.45 for both of basic scheme with shared-key discovery phase and our proposed scheme and key pool size is set to 100,000. Our scheme is more resilient up to 40% as compared to basic scheme with shared-key discovery phase when the number of nodes captured is 240.

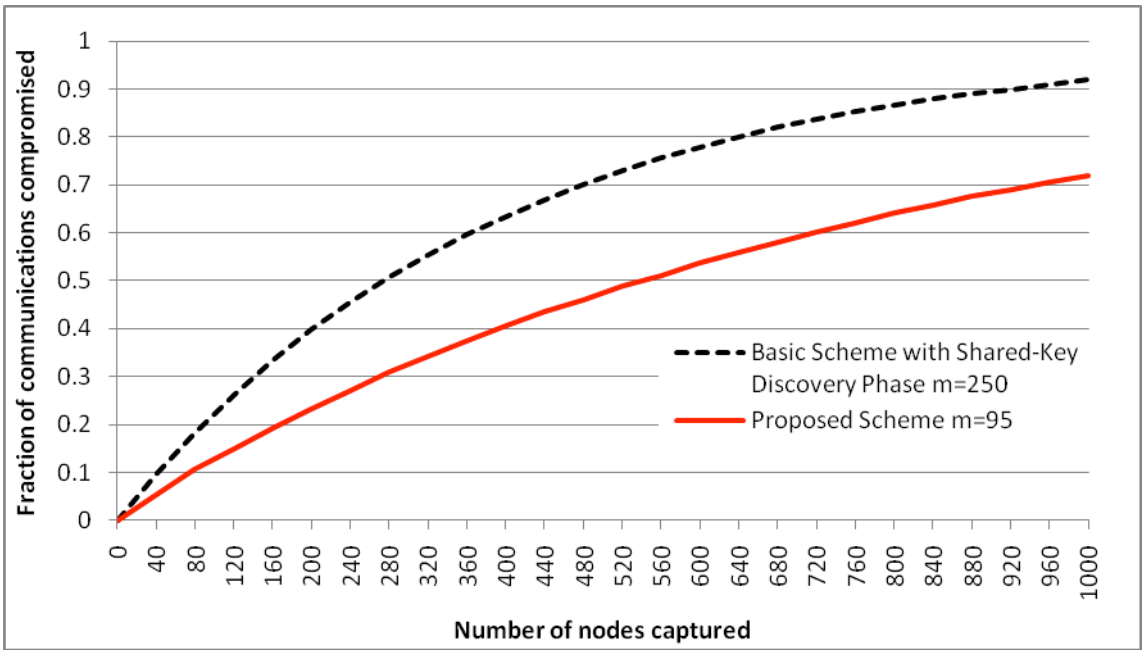

Figure 4.10. Fraction of communications compromised vs. number of nodes captured by the attacker for basic scheme with shared-key discovery phase and our proposed scheme. Local connectivity is set to 0.45 for both cases, key pool size is 100,000.

Figure 4.11 shows resilience of both schemes with local connectivity value of ~1.0. In order to catch up ~1.0 local connectivity in basic scheme with shared-key discovery phase with a key pool size of 100,000 keys, the keyring size of nodes is set to 680. The keyring size of nodes in our scheme is only 180. In this setting, the performance boost of our scheme is clearer. Our scheme is 50% better than basic scheme with shared-key discovery phase, when the number of nodes captured is 240.

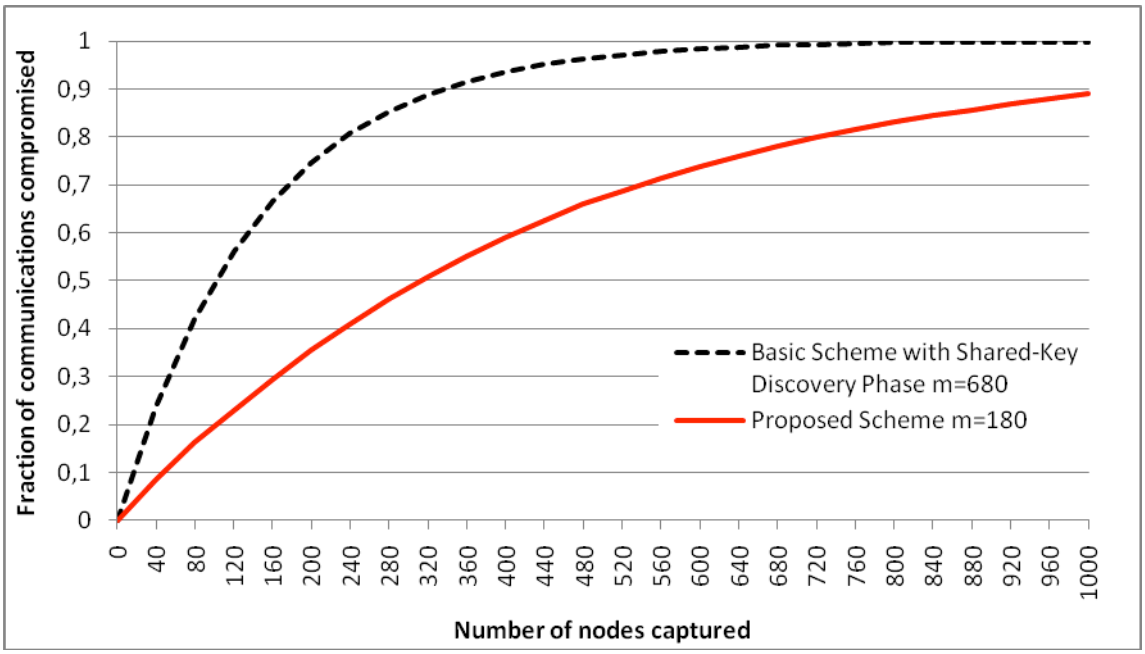

Figure 4.11. Fraction of communications compromised vs. number of nodes captured by the attacker for basic scheme with shared-key discovery phase and our proposed scheme. Local connectivity is set to ~1.0 for both cases, key pool size is 100,000.

Figures 4.12 and 4.13 show comparative resilience results of basic scheme with one-hop path-key establishment phase and our scheme with transfer phase. Path-key establishment phase and transfer phase are alternatives of each other and they are both performed after shared-key discovery phase. Our scheme achieves ~1.0 local connectivity with keyring size of 55 keys, whereas one-hop path-key establishment phase needs 75 keyring size when the key pool size is 10,000. This parameters are realized as 230 keys for the keyring size in basic scheme with one-hop path-key establishment phase and 180 keys for the keyring size in our proposed scheme when the key pool size is set to 100,000. Figures 4.12 and 4.13 show the resilience performance of both schemes when the local connectivity is ~1.0. As can be seen from both figures, those two phases give the same performance more or less. But the keyring size is 20 and 50 keys fewer in our transfer phase as compared to path-key establishment phase.

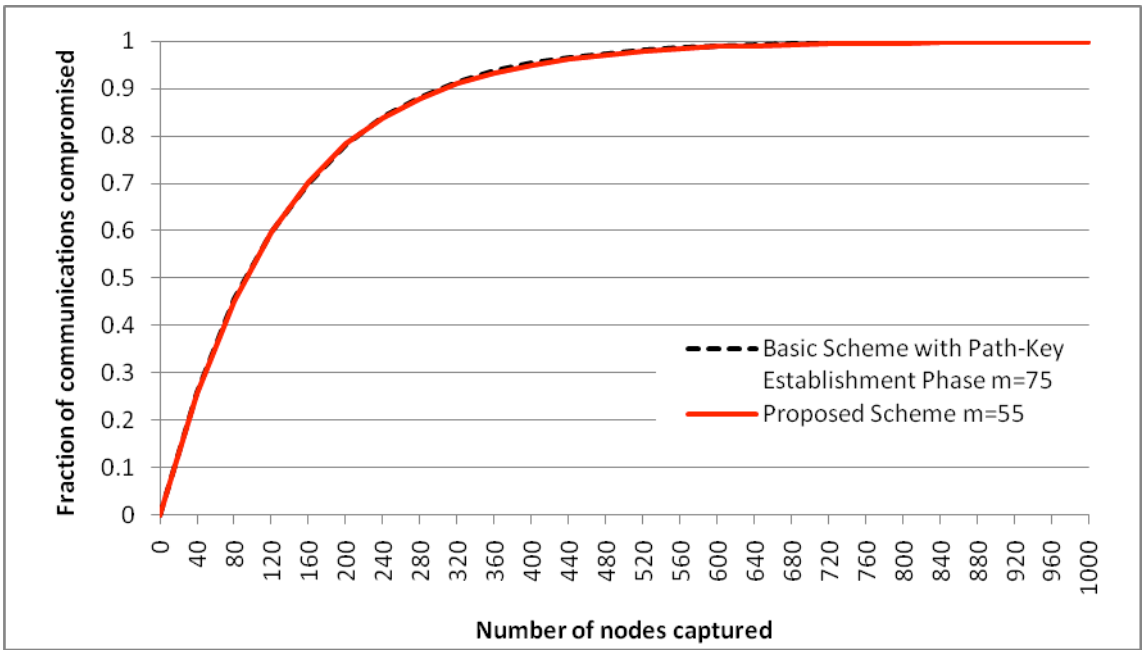

Figure 4.12. Fraction of communications compromised vs. number of nodes captured by the attacker for basic scheme with path-key establishment phase and our proposed scheme. Local connectivity is set to ~1.0 for both cases, key pool size is 10,000.

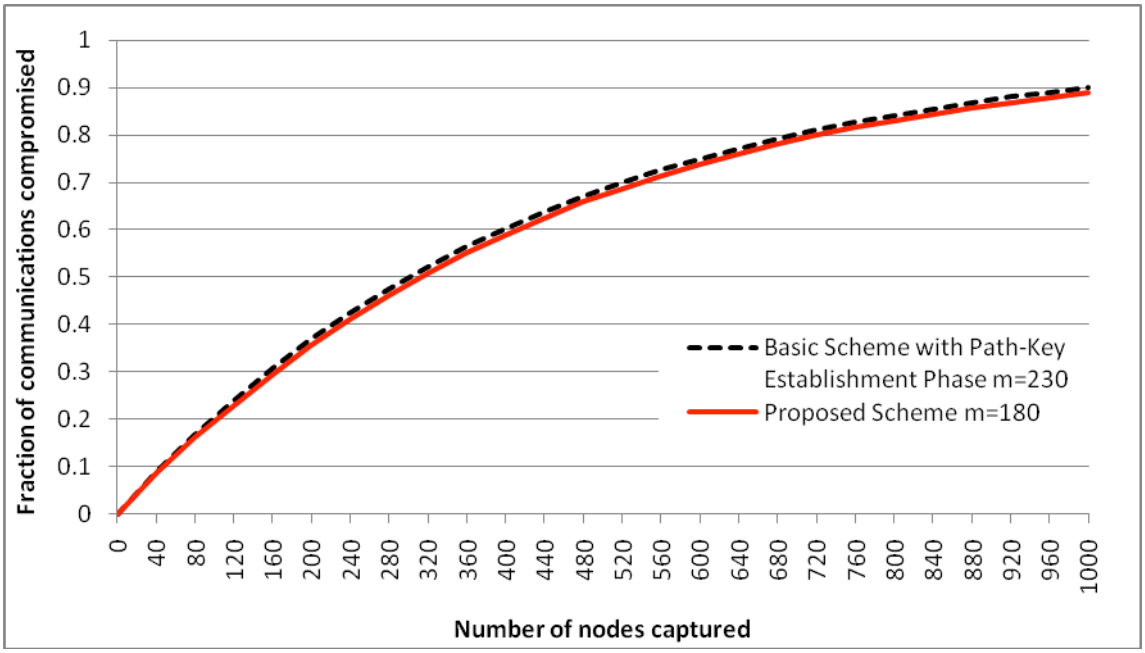

Figure 4.13. Fraction of communications compromised vs. number of nodes captured by the attacker for basic scheme with path-key establishment phase and our proposed scheme. Local connectivity is set to ~1.0 for both cases, key pool size is 100,000.

Figures 4.14 and 4.15 show additional communication cost per link that arises from secure link establishment activities in one-hop path-key establishment phase and our transfer phase. Communication cost per link is the average number of messages sent in order to establish a new secure link after shared-key discovery phase. It should be noted that communication cost resulted from shared-key discovery phase is not taken into account since this cost is same in both path-key establishment phase and our transfer phase.

In transfer phase, a node tries to find keys in the keyrings of its secure neighbors that are common with its unsecure neighbors. If there are such keys, it sends messages to transfer those keys, if there is no such a key, it does not send any message. At the same time, it informs its unsecure neighbors that it is able to establish secure link with them, then the secure neighbors transfer the common keys. If we optimize the messaging even further, nodes can concatenate the request messages to transfer common keys from secure neighbors and information messages that are sent to the unsecure neighbors in one broadcasting message. After those broadcasting messages, secure neighbors send each common key in separate messages. Additional communication cost for our transfer phase is calculated as the sum of the number of nodes initiate transfer phase messaging and number of transferred keys. Additional communication cost per link is the fraction of additional communication cost to the number of secure links established in transfer phase.

The scenario is different in path-key establishment phase. Each node broadcasts a message in order to find an intermediate node that will help to establish a secure link with its unsecure neighbor. Each node sends this broadcasting message regardless of the existence of such an intermediate node, because nodes do not have any knowledge of secure neighbor list of nodes. If there is such an intermediate node, the broadcasting node generates a link key and sends it to its unsecure neighbor through the intermediate node. As a result, additional communication cost is calculated by the number of nodes that broadcast message, all the nodes in this case, plus number of link keys generated times two for the messages from the originator to the intermediate node and from the intermediate node to the destination. Additional communication cost per link is the ratio of additional communication cost to the number of new established secure links.

In Figures 4.14 and 4.15, additional communication cost per link is depicted in logarithmic scale. Key pool size is set to 10,000 in Figure 4.14 and 100,000 in Figure 4.15. Additional communication cost per link for basic scheme with path-key establishment phase is very high at the beginning. The reason is that all the nodes broadcast message regardless of establishing secure links or not. Afterwards, additional communication cost per link value for basic scheme with path-key establishment phase stabilizes at 2.1. Additional communication cost per link value for our scheme is stabilized at a lower level, at 1.1.

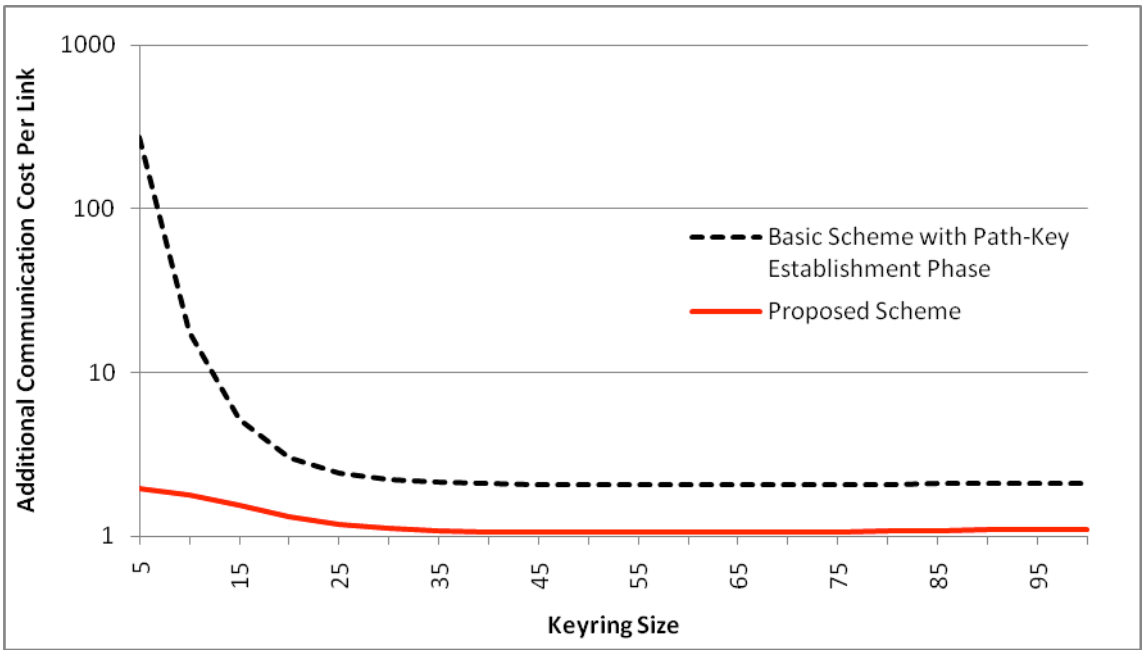

Figure 4.14. Additional number of messages sent in order to establish a new secure link in basic scheme with path-key establishment phase and our proposed scheme. Key pool size is 10,000.

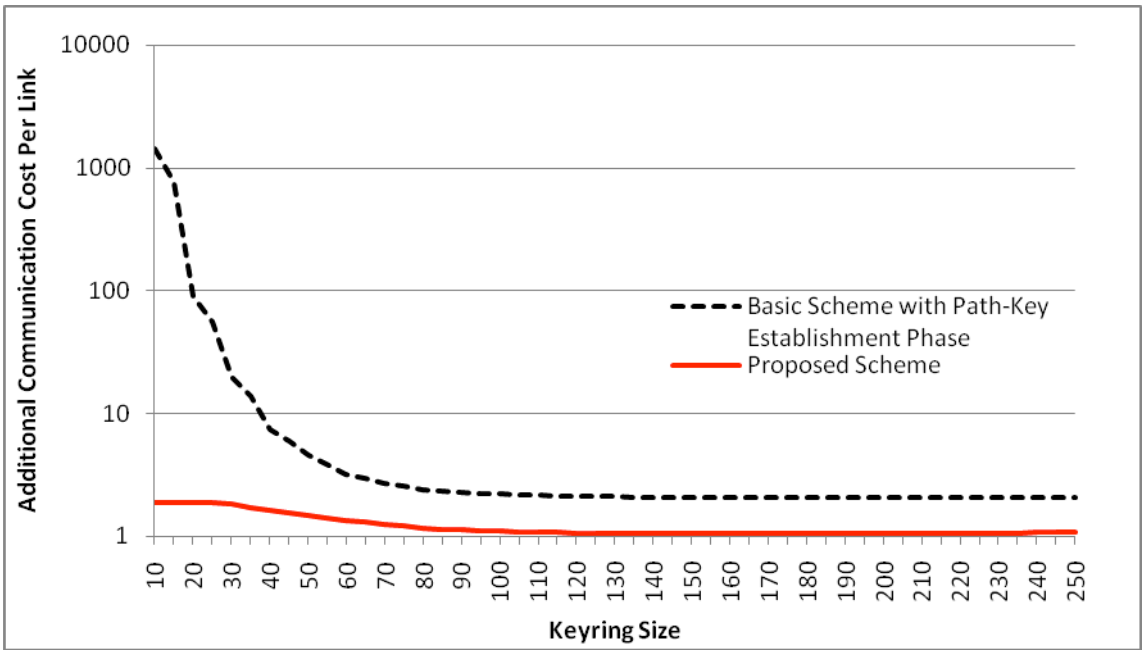

Figure 4.15. Additional number of messages sent in order to establish a new secure link in basic scheme with path-key establishment phase and our proposed scheme. Key pool size is 100,000.

In transfer phase, nodes search the keyrings of their secure neighbors if there is a match with the keys in the keyrings of their unsecure neighbors. They repeat this operation for each unsecure neighbor. The complexity of this algorithm is $O\left(k^2 \times m \log(m)\right)$ where k is the number of neighbor nodes in communication range and m is the keyring size.

4.2.3. Comparison of the Schemes Proposed in Last Two Sections

The scheme proposed in Section 3, namely XORed keying material, can reach local connectivity up to 0.97 when the optimal keyring mixture of single and XORed keys are used. This optimal case is the single keyring size of 70 keys and XORed keyring size of 30 keys when the total keyring size is 100 and key pool size is 10,000.

The same connectivity is achieved by the scheme proposed in this section when the keyring size is 50 and key pool size is 10,000.

Figure 4.16 shows the resiliency performance of the proposed scheme in Section 3 (XORed keying material) and the proposed scheme in this section in the same connectivity value. The scheme proposed in this section has better resiliency performance than the XORed keying material up to 23% when the number of nodes captured is 120.

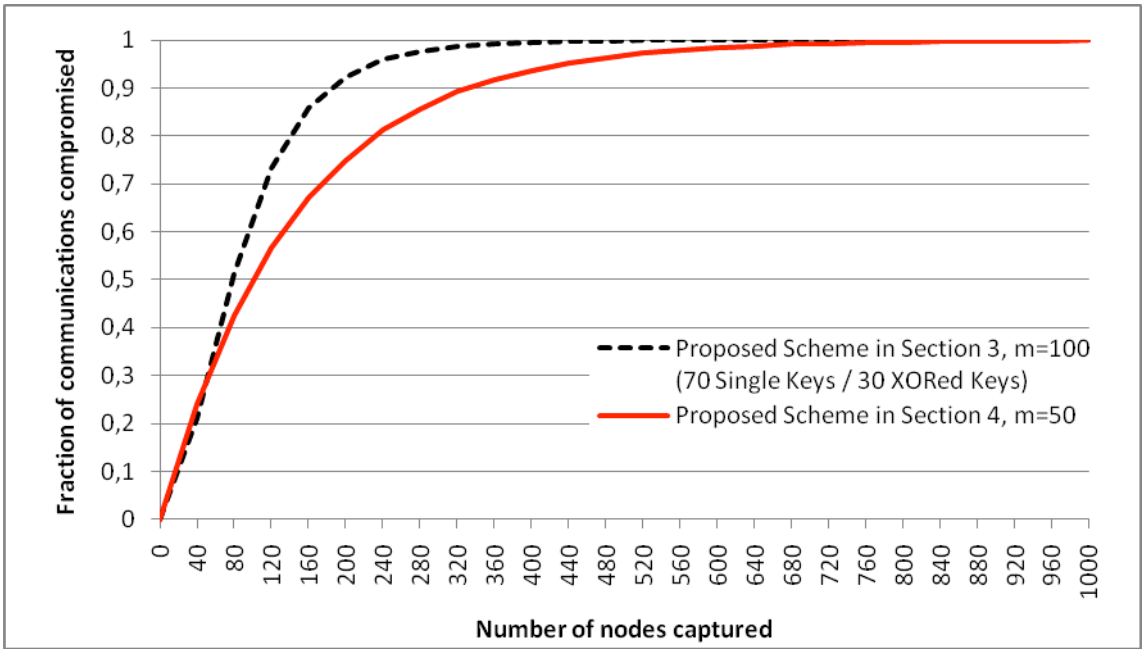

Figure 4.16. Fraction of communications compromised vs. number of nodes captured by the attacker for the proposed scheme in Section 3 (XORed keying material) and the proposed scheme in this section. Local connectivity is set to 0.97 for both schemes, key pool size is 10,000.

4.3. Discussions and Conclusions

In this section, we introduced a scheme and compared its performance to the shared-key discovery phase and path-key establishment phase of basic scheme [4]. Our

scheme achieved the same high local connectivity with one fourth of the size of the keyring as compared to basic scheme after shared-key discovery phase when the key pool size is 10,000. Our scheme has almost full connectivity while nodes are loaded with 20 and 50 fewer keys than the nodes in basic scheme after path-key establishment phase in networks with key pool sizes of 10,000 and 100,000 keys, respectively. Moreover, our scheme increases local connectivity without any degradation in resilience. Resilience performance of our scheme is up to 50% better than basic scheme after shared-key discovery phase and more or less the same with basic scheme after path-key establishment phase. The keyring indices of neighbors which are broadcasted in shared-key discovery phase are used in our scheme, thus communication cost is also minimal as compared to path-key establishment phase. Additional communication cost per link is 1 message fewer in our transfer phase.

Our contribution to basic scheme is such a generic framework that it can be applied to most specialized schemes using probabilistic key predistribution. Our contribution can be also implemented into schemes using deployment knowledge such as Du et al.'s scheme [6]. In Du et al.'s case [6], intermediate node can be used to transfer necessary shares of key matrix to establish communication with a neighbor which does not share a common key.

5. GENERATIONWISE KEY PREDISTRIBUTION APPROACH FOR MULTIPHASE WIRELESS SENSOR NETWORKS

In this section a resilient key predistribution scheme for multi-phase wireless sensor networks is proposed. We compared its performance with another well-known multi-phase key predistribution scheme, RoK [9], and we saw performance boosts in resilience up to 35%.

The rest of this section is organized as follows. The RGM scheme, our contribution, is explained in Section 5.1. Threat model and resiliency metrics are described in Section 5.2. Section 5.3 discusses the comparative performance evaluation. Finally, Section 5.4 concludes the scheme.

List of symbols used in this section is given in Table 2.1.

5.1. Our Contribution

In this subsection, the proposed RGM (Random Generation Material) key predistribution scheme for multi-phase wireless sensor networks is designed.

5.1.1. Motivation

Our aim in this study is to develop a key predistribution scheme for multi-phase sensor networks with improved resiliency as compared to RoK scheme [9]. In order to understand the resiliency behavior of RoK scheme, we analyze resistance of forward and backward keyrings. We performed simulations in order to see how many active attacks are thwarted by the use of forward and backward keyrings. The results are shown in Figure 5.1. As can be seen from Figure 5.1, number of failed attacks due to backward keyring linearly increases in time. This shows that backward keyring is playing an important role in preventing the communication links from being compromised. On the other hand, the number of failed attacks due to forward keyring remains constant at around generation 5, when initially deployed nodes are still alive. This observation shows that forward keyring is ineffective to resist against attacks after a certain point. The reason of this behavior of RoK is due to the fact that once an attacker learns a forward key, it can repeatedly compute forward keys of upcoming generations by simply hashing. Therefore, the forward key pool is totally revealed to the attacker after some generations.

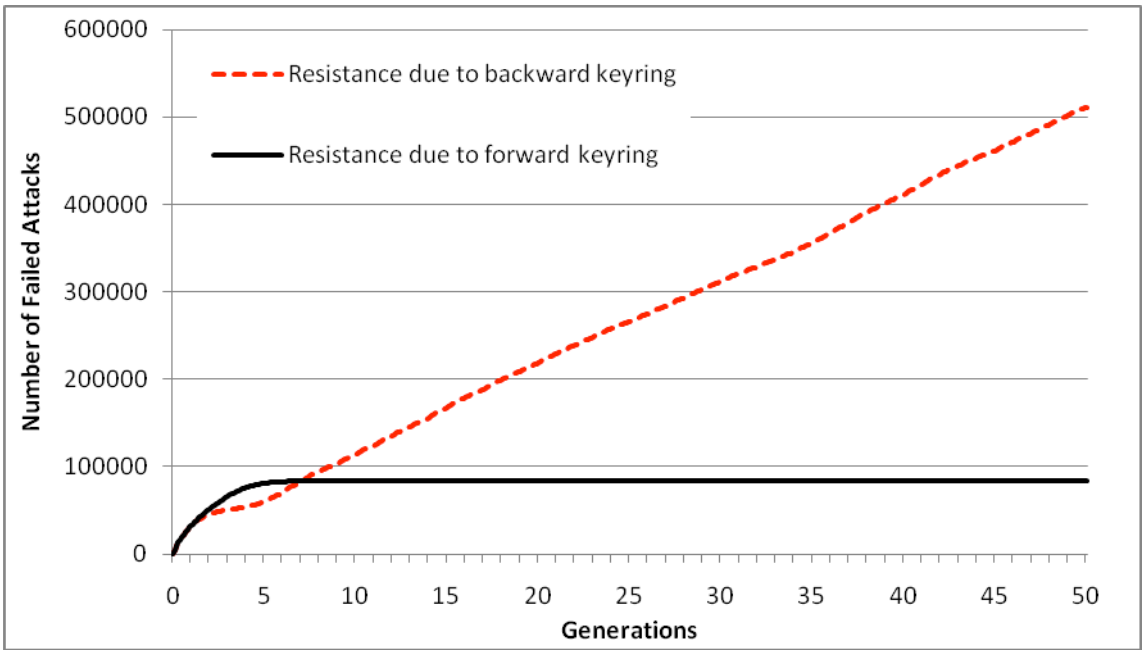

Figure 5.1. Resistance of forward and backward keyrings of RoK in case of an attack. Pool size is 10000 for forward and backward key pools; keyring size is 500. The attacker randomly captures 30 nodes per generation.

Abovementioned analyses on the resistance of forward and backward keyrings show that RoK's double hash chain approach reveals too much information to an active attacker. In order to reduce the information that an attacker obtains out of a captured node, we envision to use pairwise keys between node pairs of particular generations instead of using chain of keys lasting as long as the lifetime of the network. This is the base of our RGM approach that will be detailed in the next subsections.

5.1.2. Overview

In our RGM method, the concept of overlapping generations is not used. Instead of forward and backward keyrings, one keyring, called the *generation keyring*, is used. Also one key pool, called *generation key pool*, is used for each generation. In contrast to RoK, key pool in our RGM scheme is reconstructed randomly by a distribution center at

each generation. On the other hand, keys that are used to establish links evolve in a different way, independent of evolution of generation key pool. This will be discussed in detail below.

In our scheme, generation key pool randomly refreshes in time. Therefore, there is no relation between past and future states of the pool in our scheme. This property is very important in order to limit attacker's capability. On the other hand, we need to provide connectivity between nodes deployed at different generations. Connectivity will be obtained by evolving not the key pool itself, but evolving keys individually.

Restricting the use of a particular key for specific generations is another resiliency improving factor for our method. In our RGM scheme, the keys are designed to be used between node pairs that are deployed at particular generations. For instance, the link key used between nodes A deployed at generation f and B deployed at generation g is confined to nodes deployed at one of these generations if they store the keys with same indices. Nodes deployed at generations other than f and g have no access to this link key. This has an advantage over RoK [9] in restricting the information an attacker acquires if he/she captures a node. In such a case, he/she can compromise the keys used between generation at which captured node has been deployed and other generations within the generation window. Hence if the attacker wants to compromise a predefined link indirectly, he/she must compromise nodes which have keys in their keyring with the same shared indices and have been deployed at the same generations with one of the ends of that predefined link. What our method tries to do is to increase the number of constraints that an attacker should obey. In this way, we improve the resiliency of the wireless sensor network.

5.1.3. Predistribution of Generation Material

In multi-phase WSNs, each node tries to establish secure links with its neighboring nodes even if they belong to different generations. Therefore, nodes deployed both at the same and different generations should share keys.

In RGM, each generation has its own key pool, denoted as GKP^j for the nodes deployed at generation j.

$$GKP^j = \left\{ gk_1^j, gk_2^j, ..., gk_P^j \right\} \tag{5.1}$$

In RGM, keyring of a sensor node A deployed at generation j, is split into several sub-keyrings in order to establish link with nodes deployed at same or different generations. One of the sub-keyring is reserved for current generation keys, which are used to establish secure communication with other nodes deployed in same generation. Keys in this sub-keyring are randomly selected from the node's own generation key pool without replacement. More formally, for a node A deployed at generation j, same generation sub-keyring, GKR_A^{jj}, is as shown below.

$$GKR_A^{jj} = \left\{ gk_{t_{0,i}}^j \mid i = 1,2,...,m \right\} \tag{5.2}$$

where each $gk_{t_{0,i}}^j$ is a key selected at random from GKP^j without replacement (i.e. a particular member of GKP^j cannot appear in this sub-keyring of a particular node more than once). Here $t_{0,i}$ denotes the index of the randomly selected keys for this sub-keyring.

The same generation sub-keyring is also needed to establish secure links with nodes deployed at past generations. A secure hashing algorithm is used for that. Details will be discussed in the following subsection.

Other sub-keyrings of a node are used to establish secure links with nodes deployed at future generations. These future generation sub-keyrings contain a transformed version of generation keys selected at random from the key pools of future generations as explained below. For a node A deployed at generation j, future generation

sub-keyring containing random keys used to secure communication with nodes deployed at a future generation $j+q$, where $1 \le q < G_w - 1$ is given as follows.

$$GKR_A^{j(j+q)} = \left\{ gk_{t_{q,i}}^{j(j+q)} \mid gk_{t_{q,i}}^{j(j+q)} = H\left(gk_{t_{q,i}}^{(j+q)} \,||\, j\right), i = 1,2,...,n \right\} \tag{5.3}$$

where each $gk_{t_{q,i}}^{(j+q)}$ is a key selected at random from $GKP^{(j+q)}$ without replacement and $t_{q,i}$ denotes the index of the randomly selected keys for this sub-keyring.

This process can be detailed as follows. In order to form the future generation sub-keyring $GKR_A^{j(j+q)}$, the node *A* first picks *n* random keys from the key pool of the future generation $j+q$ without replacement. Each of these keys is appended with the generation number of *A*, which is *j*, and hashed using a secure hash function like *SHA-1* [18] or *SHA-256* [19] depending on the key size. These hashed values are stored in the sub-keyring. In this way, we customize the keys belonging to a future generation to be used in another generation without storing the actual keys, thanks to the one-way property of the secure hash functions. The way these transformed (hashed) keys are used to establish link keys will be explained in the next subsection.

To sum up, in RGM, keyring of a sensor node contains random keys picked from the current generation key pool and transformed random keys for up to $G_w - 1$ future generations picked from future generation key pools. More formally, for a node *A* deployed at generation *j*, its keyring KR_A^j is shown as follows.

$$KR_A^j = \left\{ \begin{array}{l} gk_{t_{0,1}}^{j}, gk_{t_{0,2}}^{j}, gk_{t_{0,3}}^{j}, ..., gk_{t_{0,m}}^{j}, \\ gk_{t_{1,1}}^{j(j+1)}, gk_{t_{1,2}}^{j(j+1)}, gk_{t_{1,3}}^{j(j+1)}, ..., gk_{t_{1,n}}^{j(j+1)}, \\ gk_{t_{2,1}}^{j(j+2)}, gk_{t_{2,2}}^{j(j+2)}, gk_{t_{2,3}}^{j(j+2)}, ..., , gk_{t_{2,n}}^{j(j+2)}, \\ \vdots \\ gk_{t_{G_w-1,1}}^{j(j+G_w-1)}, gk_{t_{G_w-1,2}}^{j(j+G_w-1)}, gk_{t_{G_w-1,3}}^{j(j+G_w-1)}, ..., gk_{t_{G_w-1,n}}^{j(j+G_w-1)} \end{array} \right\} \tag{5.4}$$

In RGM, sensor nodes store *m* current generation keys and *n* future generation keys for each upcoming generation. Because a sensor node may communicate with at most $G_w - 1^{st}$ next generation, the maximum value for the total amount keys in the

generation keyring of a particular node (say node *A* deployed at generation *j*) is calculated as follows.

$$\left|KR_A^j\right| = (G_w - 1) * n + m \tag{5.5}$$

However, in RGM, it is possible not to share keys with less future generations in order to save the memory. We will detail this case in Section 5.3.2.

5.1.4. Calculation of Link Keys

After the generation of the keyrings, the nodes are deployed over the sensor field. At the beginning of each generation, each node broadcasts the indices of the keys in its keyring together with the corresponding generation information. After that, the nodes decide whether they have common keys or not. In our RGM scheme, as in RoK scheme, all the shared generation keys contribute to the link key. Contribution of as many keys as possible increases the resistance of link against attacks.

If two nodes, say *A* and *B*, belong to the same generation *j* and if they have common generation keys with indices $v_1, v_2, ..., v_z$ (i.e. the keys $gk_{v_1}^j, gk_{v_2}^j, ..., gk_{v_z}^j$), they compute their link key as follows.

$$k_{AB} = H\left(gk_{v_1}^j \,||\, gk_{v_2}^j \,||\, ... \,||\, gk_{v_z}^j\right) \tag{5.6}$$

If the nodes belong to different generations, the link key creation process is different. Let's suppose, node *A* deployed at generation *f* and another node *B* deployed at generation *g* have common generation keys with indices $v_1, v_2, ..., v_z$. They compute their link key as follows:

$$k_{AB} = H\left(gk_{v_1}^{fg} \,||\, gk_{v_2}^{fg} \,||\, ... \,||\, gk_{v_z}^{fg}\right), \text{ where } f < g \tag{5.7}$$

Node *A* already has each of these generation keys gk_v^{fg} since they are in the future generation sub-keyrings of *A*. By using these keys, node A can easily calculate k_{AB}. However, from the point of view of node *B*, the generation keys gk_v^{fg} are past generation keys and therefore they are not the part of its keyring. Fortunately, node *B* can calculate gk_v^{fg}. As discussed in the previous subsection, $gk_v^{fg} = H(gk_v^g \,||\, f)$, where *H* is a secure and one-way hash function. Node *B* is deployed at generation *g*, therefore its same generation sub-keyring contains the current generation key gk_v^g. By using gk_v^g, it calculates the keys gk_v^{fg} and then k_{AB}.

The generation key gk_v^{fg} with index *v* can be computed by a node if and only if this node has been deployed at generation *g* and has current generation key gk_v^g with index *v* in its keyring. As a result, a sensor node *C*, which is deployed at another generation *e* cannot compute generation key gk_v^{fg} even if it has generation key gk_v^{eg} with the same index *v*. The reason is that the node *C* needs gk_v^g in order to compute gk_v^{fg} and it cannot obtain gk_v^g using gk_v^{eg} due to the one-way property of the secure hash function *H*.

It is clear that a generation key gk_v^{fg} with index *v* may be known only by nodes of generations *f* and *g*. No node deployed at another generation can compute the key that is unique to generation *f* and *g*. So an attacker has to spend extra effort in order to capture the nodes belonging to generations *f* or *g* if he/she wants to acquire the link key between nodes *A* and *B* that are deployed at generations *f* and *g*, respectively. This is the main factor of improved resiliency in our RGM scheme. However, storing generation keys for particular generation pairs causes increased memory usage in order to reach acceptable level of connectivity. Such performance metrics are analyzed in more detail in Section 5.3.

5.2. Threat Model and Resiliency Metrics

In this subsection, we describe the threat model used in this thesis for both RoK and our proposed RGM schemes. We assume the existence of an attacker who is capable of capturing nodes at random locations in the sensor field. The rate at which the attacker captures nodes is a system parameter. As the attacker captures nodes, it learns keys that exist in the captured nodes' keyrings. These keys are not only used to compromise the communications of the captured nodes, but also used to compromise the secure links among non-captured nodes since the keys can be reused throughout the networks. A resilient design aims reduce the effect of captured nodes over such innocent nodes.

We employ two attacker types, eager attacker and temporary attacker, as in [9]. Eager attacker identifies an attacker who starts his/her activity at the beginning of network and continues till the end. Temporary attacker attacks on the network for a fixed amount of time which is relatively smaller than the lifetime of the network.

As in [9], the resiliency of the sensor network against node capture attacks will be measured via *compromised link ratio* in this section. Compromised link ratio is defined as the fraction of indirectly compromised links. That is, if this ratio is low, the network is more resilient. Indirectly compromised link is a link whose keys are known by the attacker, but none of the sensors in both ends is captured.

We consider two resiliency metrics: *active resiliency* and *total resiliency*.

Active resiliency is the resiliency due to current alive links. A link is said to be alive if both ends are alive; otherwise, that link is said to be dead. Active resiliency is measured with the metric of *active compromised links ratio* which is defined as the ratio of compromised alive links to all established alive links.

Total resiliency is the resiliency due to all links (dead and alive) and it is measured with *total compromised links ratio* metric. Total compromised links ratio is defined as the ratio of the all indirectly compromised dead or alive links

over all established links since the beginning of the network. Total resiliency is important if the attacker keeps log of communications in the network. When a link is compromised even if its endpoints are dead, an attacker may have a look at the logged messages and learn the contents. Of course, the content of the message may not be as valuable as when it was sent.

Actually, both active and total compromised links ratio metrics are inversely related to the resiliency. As these ratios go low, the network's resiliency becomes high. In the rest of this section, we will use both *resiliency* and *compromised links ratio* terms. The readers are advised to interpret the inverse relationship between these two terms.

5.3. Performance Evaluation

We performed simulative performance evaluation to compare the performances of RoK [9] and the proposed RGM methods. Some results are also verified analytically. Different scenarios based on different keyring sizes are analyzed. We first give the analytical formulations for the proposed RGM scheme. Then, the simulation setting is described. After that we discuss performance results of three different scenarios.

5.3.1. Analytical Formulations

In this subsection we formulate upper and lower bounds of the *local connectivity* metric in RGM. Local connectivity is defined as the probability that two neighboring nodes sharing at least one key in their keyrings. In RGM scheme, key sharing

probabilities of the node pairs deployed at the same generations and at different generations are different since different parts of the keyring are utilized for these two types of connections.

We first formulate the key sharing probability of neighboring node pairs deployed at the same generation, which is denoted as p_{sg}. Current generation sub-keyrings are used for same generation nodes. There are m current generation keys in the keyrings of the nodes deployed at the same generation. Such a case is similar to Eschenauer and Gligor's basic scheme [4]. As described in [7, 9], the probability that two nodes sharing at least one key in their sub-keyrings of m keys is given as follows.

$$p_{sg} = 1 - \frac{\binom{P}{2m}\binom{2m}{m}}{\binom{P}{m}^2} \tag{5.8}$$

We now formulate the key sharing probability of neighboring node pairs deployed at different generations, which is denoted as p_{dg}. A node can produce m past generation keys by making use of its m current generation keys. Besides, a node retains n future generation keys for each future generation in its window. In order for a node pair deployed at different generations to establish a pairwise key, there should be at least one common key between the current generation sub-keyring of one of the nodes, say A, and the future generation sub-keyring of the other node, say B, that corresponds to the generation of A. This is equivalent to the fact that a sub-keyring with m elements and another sub-keyring with n elements, which are picked from a global key pool of P elements, should have a non-null intersection set. There are total of $\binom{P}{m}\binom{P}{n}$ such sub-keyring pairs out of which $\binom{P}{m+n}\binom{m+n}{m}$ sub-keyring pairs have no common elements. Thus the probability that two nodes deployed at different generations sharing a common key, p_{dg}, is formulated as follows.

$$p_{dg} = 1 - \frac{\binom{P}{m+n}\binom{m+n}{m}}{\binom{P}{m}\binom{P}{n}} \tag{5.9}$$

In our RGM scheme, we take $m \geq n$. In this setting, equations 5.8 and 5.9 imply that $p_{sg} \geq p_{dg}$, since key sharing probability is higher for larger keyring size. At any given time, a mix of current generation and different generation neighbors exist in the network. Therefore, the overall local connectivity is between these two probability values. In conclusion, it can be said that p_{sg} is the upper bound and p_{dg} is the lower bound for local connectivity, i.e. $p_{dg} \leq P_{local} \leq p_{sg}$. Due to node depletions and new deployments, local connectivity may change at each generation. However, it should remain within that range. In the beginning, all the nodes are deployed at the same initial generation, that's why local connectivity is expected to be equal to p_{sg} at generation 0. This is the maximum value of the local connectivity. As the nodes start to die and new nodes replace the dead ones in upcoming generations, the nodes in the network become heterogenic with respect to deployment generations. As a result, local connectivity value is expected to decrease and approach to the lower bound p_{dg}, but we do not expect that the local connectivity reaches p_{dg}. This is due to the fact that there will always be some neighboring node pairs belonging to the same generation, which will cause increased local connectivity. We will simulatively analyze how the local connectivity changes in time in Section 5.3.3. At this section, we will also show that the lower and upper bounds, and abovementioned expected behavior hold.

Another performance metric is about the resiliency of the network in the presence of an attacker. This metric is the probability that a link is compromised when a set of nodes are captured. Analytical formulation of this metric is given in [9]. However, instant changes at generation level cannot be observed due to approximations made in the formulas. Therefore, we decided not to use analytical formulation for resiliency metrics, but conduct extensive simulations that will be detailed in the next subsections. We compared our simulation results for RoK with the ones given in the original paper [9]. This verifies the accuracy and correctness of our simulation code.

5.3.2. Simulation Setup

All the simulations are implemented in Python programming language using Eclipse Classic environment for Microsoft Windows Vista 32-bit operating system. We worked on an Intel Q6600 powered machine running at 2.4 GHz clock speed.

In our simulations of both RGM and RoK schemes, 400 sensor nodes are initially deployed in a square field using two-dimensional uniform random distribution. One side of this square field is 634 meters that makes average number of neighbors of a sensor node approximately 4 nodes with the assumption that communication range of a node is 40 meters. Generation window (G_w) is set to 6. Sensor nodes have a random lifetime assigned according to Gaussian distribution with mean $G_w/2$, and standard deviation $G_w/6$. Whenever a node is deployed, it starts to function immediately. In our simulation model, the sink determines the amount of nodes that are going to stop functioning before each generation. After that, at the beginning of each generation, that amount of new nodes are deployed over the sensor field using two-dimensional uniform random distribution.

Simulations are run along 50 generations. For the sake of accuracy, all simulations are run 25 times and their average values are shown in the figures.

We have analyzed three simulation scenarios. In all of them, they key pools size of RGM scheme is set to 10000 keys. Moreover, the total keyring size of the RGM scheme is taken as 500 keys. As discussed in Section 5.1, RGM scheme has different *m* and *n* values for current generation sub-keyring and future generation sub-keyrings. In our simulations, *m* is set to 200 and *n* is set to 100. Normally, a node contains future generation sub-keyrings for up to $G_w - 1$ upcoming generations. However, in order to optimize keyring size and use the memory efficiently in RGM, we do not store last two future generation sub-keyrings in the nodes' keyrings since the probability of encountering two neighboring nodes with ages differing more than 3 generations is very low (this result has been obtained experimentally, however, for the sake of brevity we

will not discuss the details of these experiments). In this way, the size of the keyring of a sensor node in RGM becomes $(G_w - 3) \cdot n + m = (6-3) \cdot 100 + 200 = 500$.

The key pool and keyring size for RoK in each simulation scenario are different. The details about how the key pool size is set and keyrings are arranged will be given for each scenario separately.

Our attack model assumes an attacker who can capture nodes at random locations. In the simulations, the attacker's capture rate is taken as 1, 3 and 5 nodes per round. In our scheme, *round* is the time unit and one generation consists of 10 rounds.

5.3.3. Scenario 1: Same Key Memory Usage Case

In the first group of simulations, our RGM scheme and the RoK scheme are compared in exactly same key memory and key pool sizes. For both RoK and RGM schemes, the key pool size is set to 10000 keys and the keyring memory size is set to 500 keys. For the RoK scheme, the keyring size of 500 is the minimum size to reach 1.0 local connectivity when the key pool has 10000 keys. RoK utilizes memory by dedicating half of the memory to forward keyring, and the other half to backward keyring. In the following simulations, each RoK keyring has 250 keys. In the RGM scheme, $m = 200$ keys of the total 500 keys are used for same generation sub-keyring. Moreover, there are $G_w - 3 = 3$ future generation sub-keyrings in RGM and $n = 100$ future generation keys are predistributed for each upcoming generation.

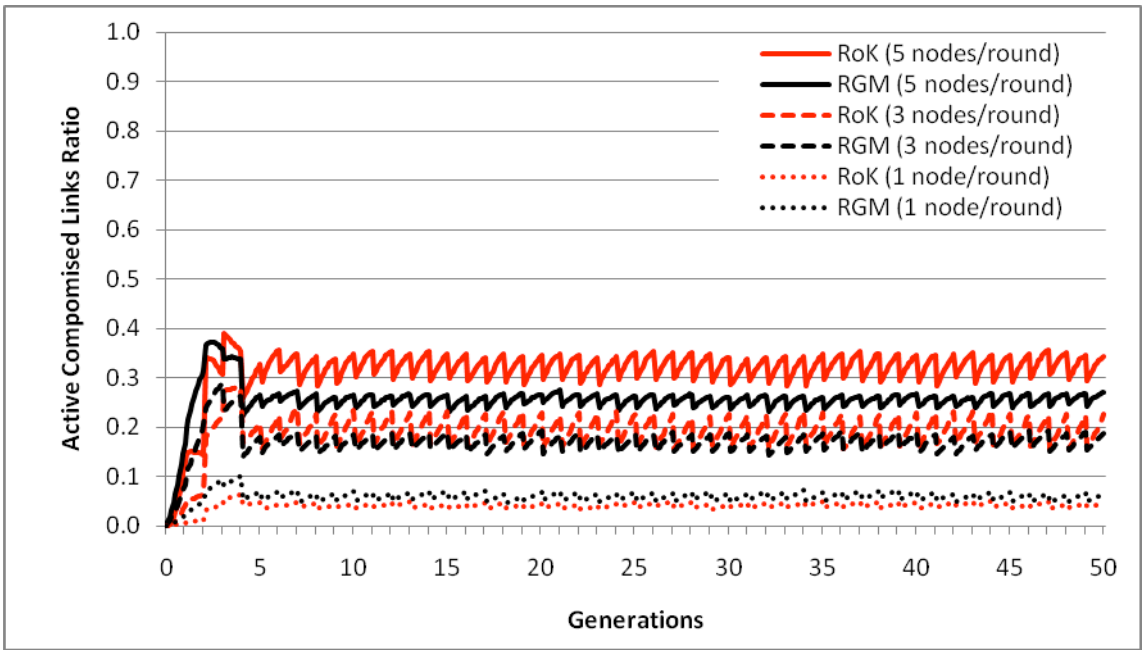

Figure 5.2. Active compromised links ratio of RoK and RGM in case of an eager attacker with capture rates of 1, 3, and 5 nodes per round, memory size is 500 and key pool size is 10000 for both of RoK and RGM.

Figures 5.2 and 5.3 show resiliency of the sensor network against an eager attacker. The resiliency metrics used here are active and total compromised links ratios as defined in Section 5.2. Eager attacker starts capturing nodes at the beginning of the network and continues indefinitely.

As can be seen from Figure 5.2, active compromised links ratio reaches its highest value in around generation 3, when most of nodes deployed at the initial phase are alive. After this time on, first nodes start to die, and resiliency stabilizes together with arrival of new nodes. Our results show that in the steady state our RGM scheme performs 35% better than RoK scheme at capture rate of 5 nodes per round. At capture rates of 1 and 3 nodes per round, in the steady state both schemes perform almost equally.

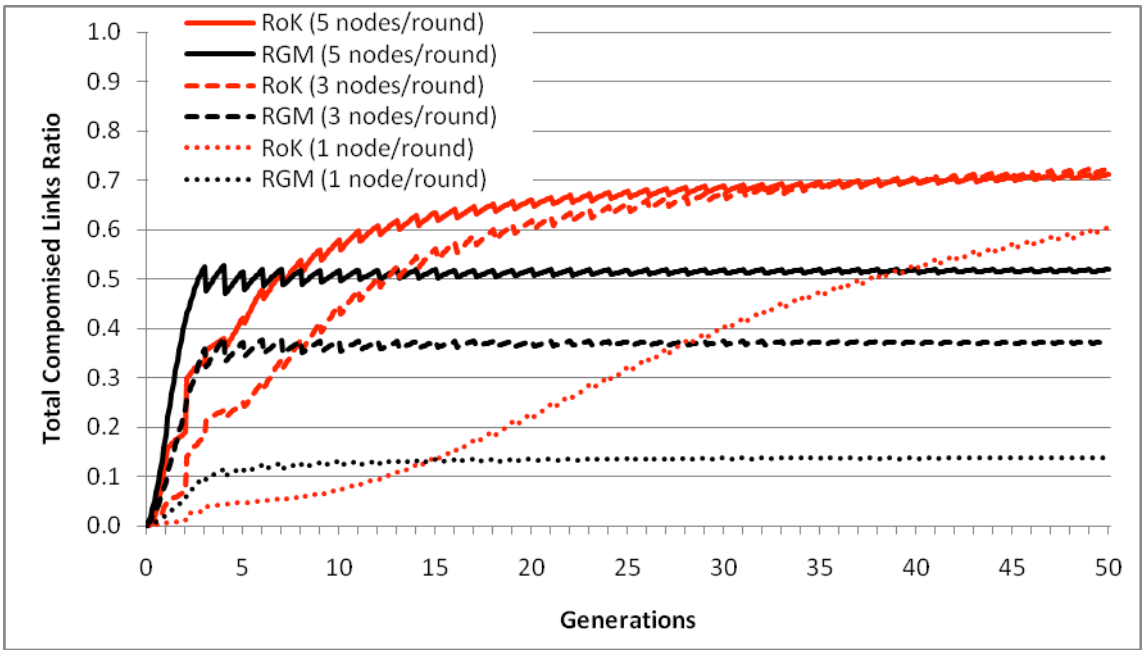

Figure 5.3. Total compromised links ratio of RoK and RGM in case of an eager attacker with capture rates of 1, 3, and 5 per round, memory size is 500 and key pool size is 10000 for both of RoK and RGM.

Figure 5.3 shows total resiliency of RoK and our RGM schemes. As can be seen from Figure 5.3, RGM scheme, as opposed to RoK scheme, reaches its steady-state at the beginning of network lifetime, around 3rd generation. The reason of this behavior of RGM is that the generation key pools are independent of each other and are useful only for small period of time. On the other hand, RoK's total compromised links ratio has tendency to increase as the network gets older. Although, RoK initially performs better, in the later generations (after 5th generation for 5 nodes/round; after 9th generation for 3 nodes/round, after 15th generation for 1 node/round) RGM starts performing better than RoK in all capture rates. In RoK, total compromised links ratio values tend to converge around 0.7 in high capture rates, meaning that 70% of the links are compromised. However, in our RGM scheme, even if the network is attacked with the highest rate of 5 nodes per round, the total compromised links ratio value slightly goes over 0.5. For lower capture rates, performance improvement of RGM over RoK is clearer. When the capture rate is 1 node per round, at the end of the simulations the number of links that the attacker captures is approximately five-fold more in RoK scheme as compared to our RGM.

Figure 5.4 comparatively shows active resiliency of RoK and RGM in case of a temporary attacker who starts his/her activity in generation 5 and ends in generation 14. When the attack starts, compromised links ratio raises up to an upper-bound which is different for all corruption rates. After attack stops, network starts to heal itself. As it can be seen from Figure 5.4, both schemes completely heals (i.e. ratio of compromised links come to zero) almost at the same time (generation 18). Figure 5.4 also shows that during the attack, RGM performs as much as 30% better than RoK for capture rate of 5 nodes per round.

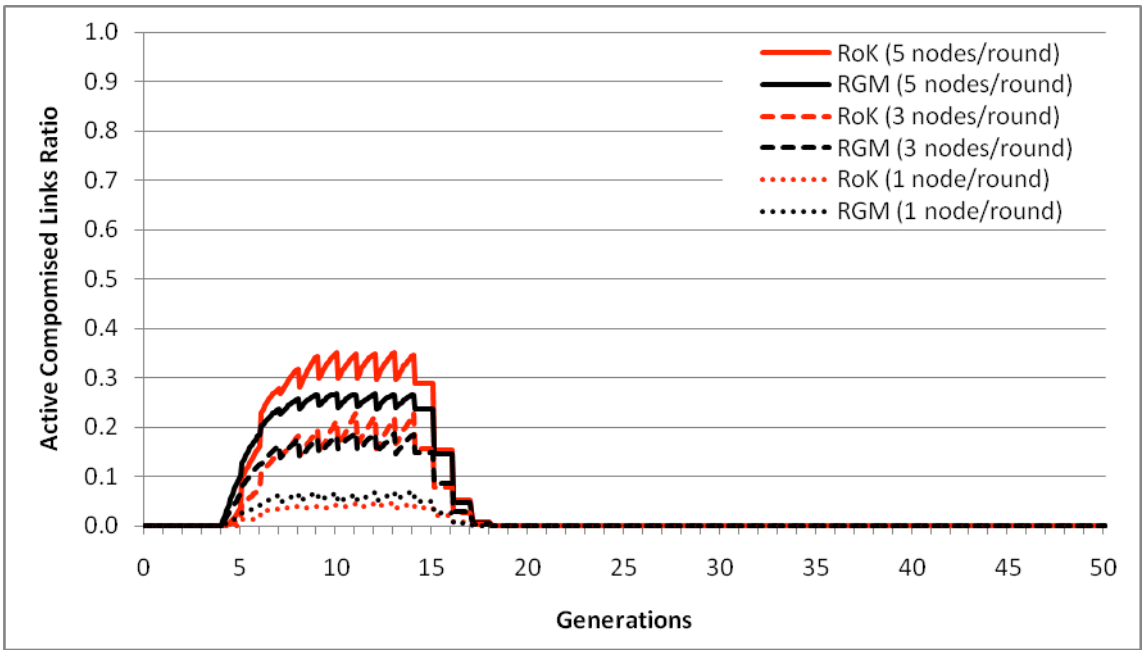

Figure 5.4. Active compromised links ratio of RoK and RGM in case of a temporary attacker with capture rates of 1, 3, and 5 per round, memory size is 500 and key pool size is 10000 for both of RoK and RGM.

One drawback of RGM scheme is that *local connectivity* of RGM tends to be less than in RoK. As described previously, *local connectivity* is defined as the probability of any two neighbors to share at least one common key. The local connectivity values of RoK and RGM for the cases that we analyze are given in Figure 5.5. As can be seen from this figure, in RGM neighboring nodes share at least one key with a probability of around 0.9, while in RoK this value is 1.0. That means our scheme performs 10% worse than RoK. However, this does not mean that 10% of the nodes are completely disjointed from the network. 0.9 local connectivity means 10% of all possible links are insecure.

However, the end points of these insecure links have at least one other secure neighbor and via these secure neighbors they are connected to the network securely. That is why 90% local connectivity, as in RGM scheme, is considered more than enough for the secure operation of the network.

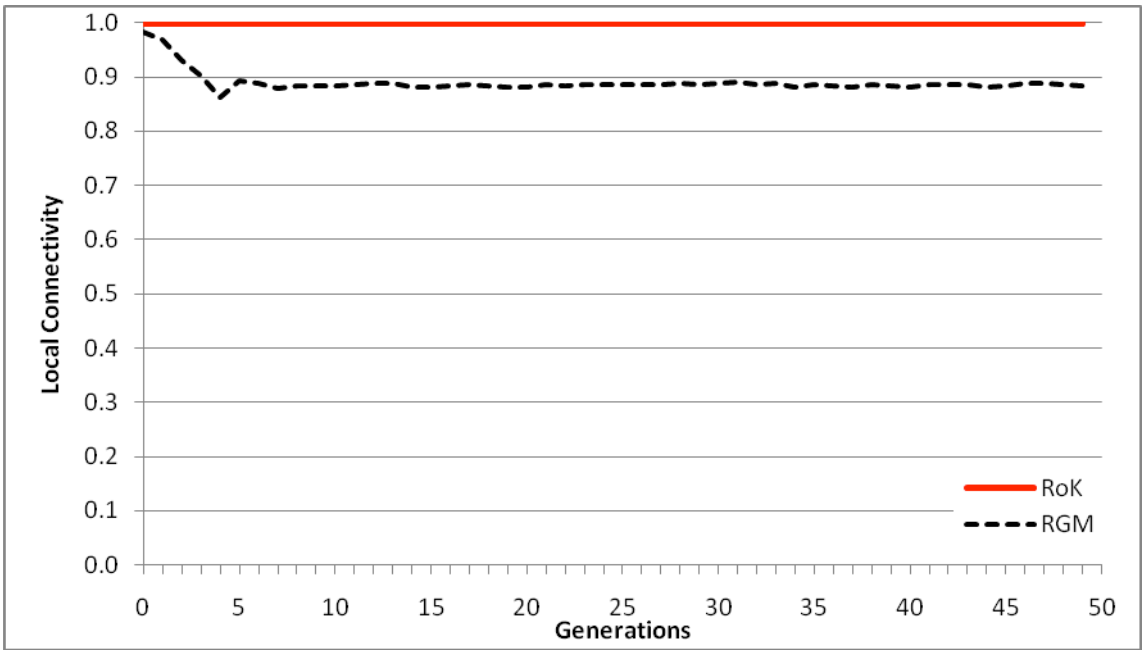

Figure 5.5. Local connectivity of RoK and RGM, memory size is 500 and key pool size is 10000 for both of RoK and RGM.

In this scenario, where P is 10000, m is 200, and n is 100 for RGM, Equations 5.8 and 5.9 yield that $p_{sg} = 0.983$ and $p_{dg} = 0.869$. It can easily be seen from Figure 5.5 that local connectivity is almost equal to p_{sg} at generation 0 since all nodes are same generation nodes initially. As time goes along, connectivity decreases as the nodes belonging different generations are deployed. Around generation 5, local connectivity reaches steady state. At this steady state, local connectivity is greater than the lower bound p_{dg}, but never reaches this lower bound due to same generation nodes existing in the network. These observations are consistent with the analytical discussions given in Section 5.3.1.

5.3.4. Scenario 2: Same Local Connectivity Case

In this scenario we compare our RGM scheme at the same local connectivity value as the RoK scheme. We selected 0.9 as this connectivity value. To reach this value, the memory size of a node in RoK is set to 300 keys, where half of this memory is used for forward keyring, and the other half is used for backward keyring. The memory size for RGM is the same as Scenario 1 (total memory is 500 keys, m = 200, n = 100 and the number of future generation sub-keyrings is 3). The key pool size is set to 10000 keys. In this configuration, both RoK and RGM have 0.9 local connectivity as shown in Figure 5.6. Initially, local connectivity of RGM is higher than 0.9, but it falls down in the first 3 generations, and then it stabilizes at 0.9. The reason for this behavior has been discussed in Section 5.3.3.

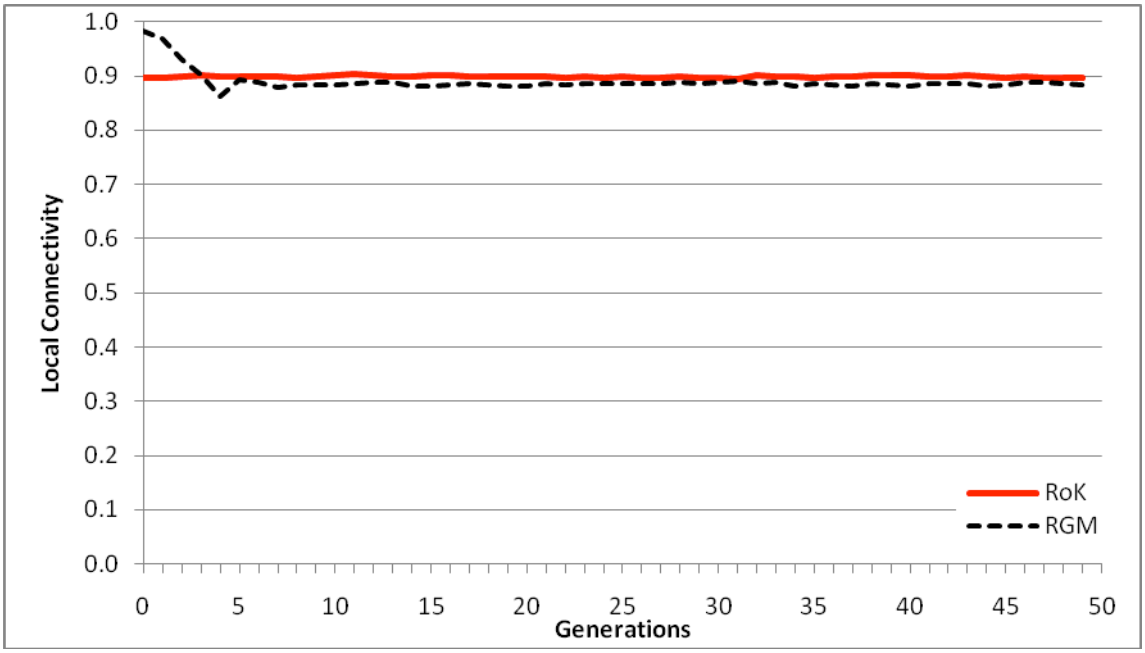

Figure 5.6. Local connectivity of RoK and RGM, memory size is 300 for RoK and 500 for RGM, key pool size is 10000 for both of them.

Figure 5.7 and 5.8 show resiliency of RoK and RGM in case of an eager and temporary attacker infiltrates into the network, respectively. RGM configuration and consequently the resiliency behaviors are same as before, but this time memory size of nodes in RoK is lowered while keeping the pool size fixed. This causes a bit less active

resiliency (i.e. a bit higher compromised links ratio) for RoK in both eager and temporary attack cases as compared to Scenario 1. In general, we can say that in this scenario active resiliency of our proposed RGM scheme is 35% - 50% better than RoK for capture rates of 3 and 5 nodes per round.

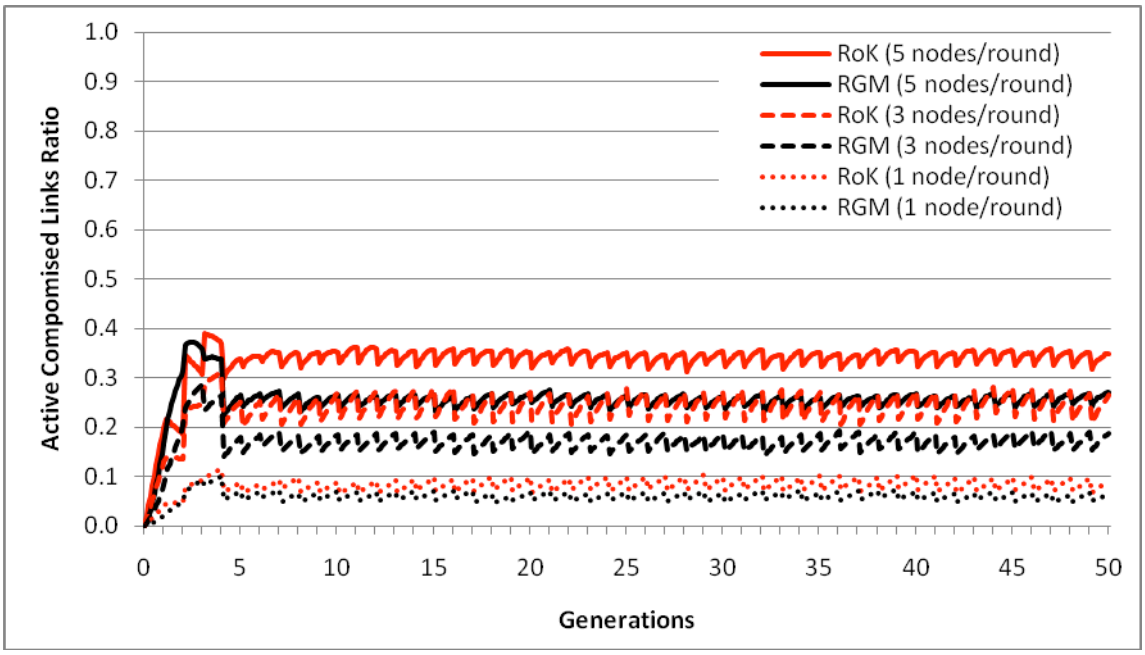

Figure 5.7. Active compromised links ratio of RoK and RGM in case of an eager attacker with capture rates of 1, 3, and 5 nodes per round, memory size is 300 for RoK and 500 for RGM, key pool size is 10000 for both of them.

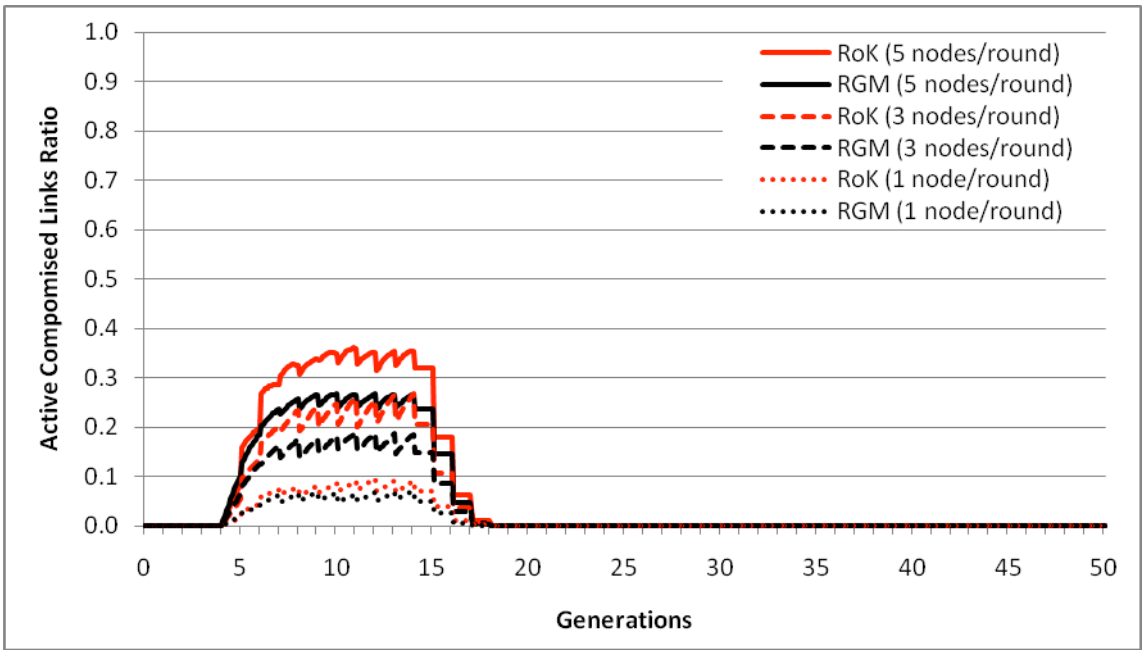

Figure 5.8. Active compromised links ratio of RoK and RGM in case of a temporary attacker with capture rates of 1, 3, and 5 per round, memory size is 300 for RoK and 500 for RGM, key pool size is 10000 for both of them.

Figure 5.9 shows the total resiliency performance in the eager attack case. The performance of RGM scheme is the same as Scenario 1 since the same parameters are used for RGM. Although the keyring size is reduced in this scenario for RoK, we do not observe any significant difference in RoK scheme. Thus, as described in Section 5.3.3 for Scenario 1, total resiliency performance of our RGM scheme is clearly better than RoK in all capture rates.

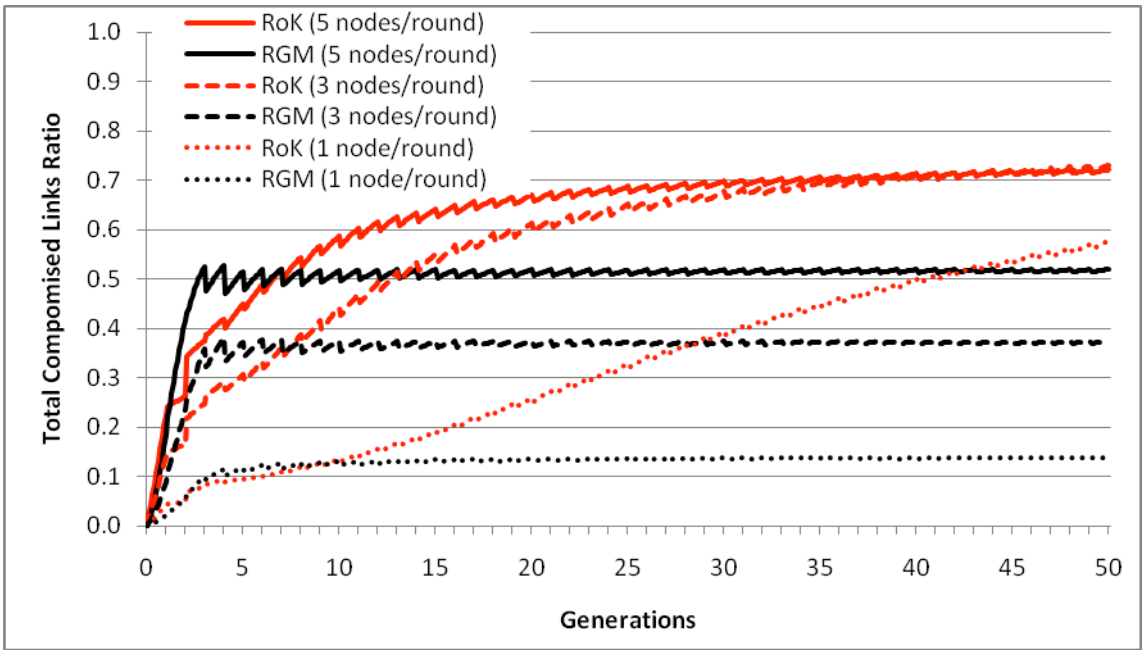

Figure 5.9. Total compromised links ratio of RoK and RGM in case of an eager attacker with capture rates of 1, 3, and 5 per round, memory size is 300 for RoK and 500 for RGM, key pool size is 10000 for both of them.

5.3.5. Scenario 3: Same Keyring Memory Size and Same Local Connectivity Case

In this last scenario, we consider same memory usage and same local connectivity cases for both RoK and RGM. To have equality in these two parameters, we used different key pool sizes for both schemes. Unlike previous scenarios, forward and backward key pool size in RoK is set to 28000 keys each. In our RGM scheme, the generation key pool size is set to 10000 keys as in other scenarios. Keyring size is set to 500 keys in both RoK and RGM schemes. As before, RoK utilizes its memory by reserving its half of memory to backward keyring and the other half to forward keyring; in our RGM scheme, nodes are loaded with current generation sub-keyring of size $m = 200$, and three future generation sub-keyrings of size $n = 100$ each.

Figure 5.10 and 5.11 show active resiliency against eager and temporary attackers, respectively. As can be seen from these figures, in this scenario both RoK and RGM perform equally. However, our RGM scheme still performs better in total resiliency metric which is depicted in Figure 5.12.

Although RoK improves its own total resiliency in this scenario as compared to previous scenarios due to increased key pool size, Figure 5.12 shows that our RGM scheme is still better than RoK in all capture rates up to three-fold. Total compromised links ratio of RoK scheme becomes greater than that of RGM around 15th generation and RGM scheme preserves this better performance until the end of network life.

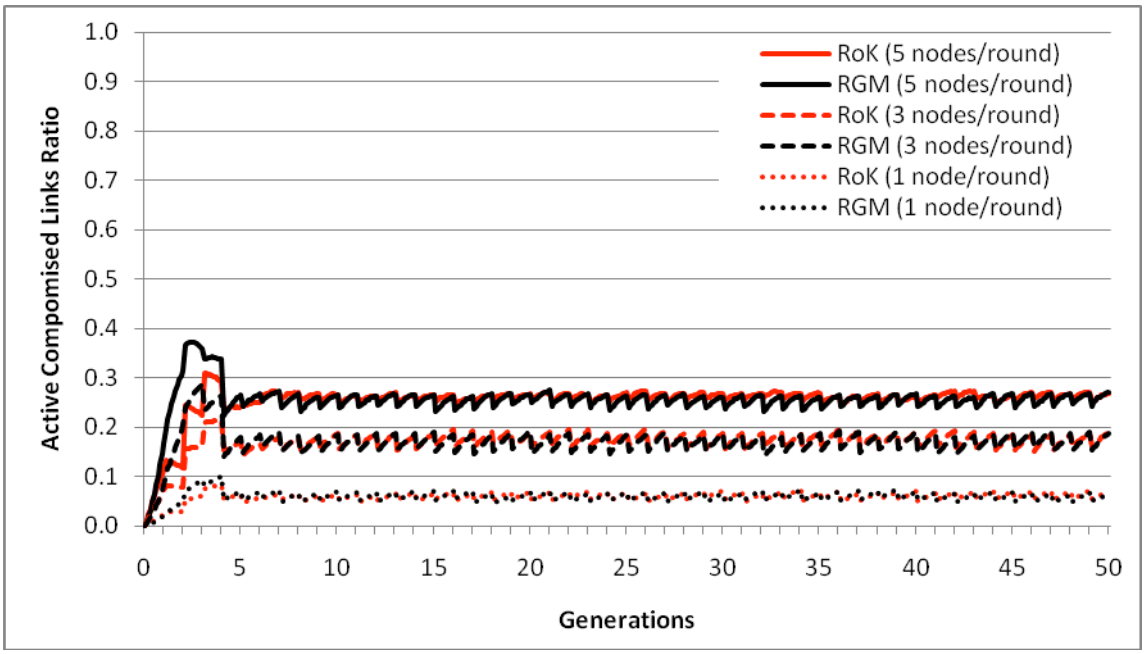

Figure 5.10. Active compromised links ratio of RoK and RGM in case of an eager attacker with capture rates of 1, 3, and 5 nodes per round, memory size is 500 for both of RoK and RGM, key pool size is 28000 for RoK, 10000 for RGM.

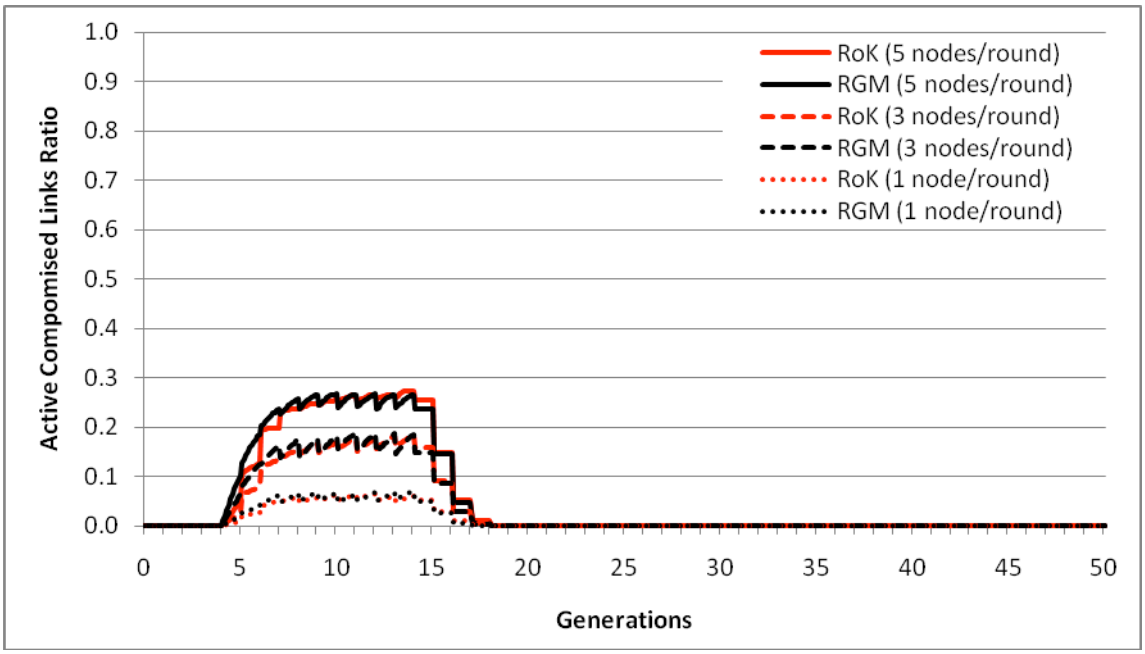

Figure 5.11. Active compromised links ratio of RoK and RGM in case of a temporary attacker with capture rates of 1, 3, and 5 per round, memory size is 500 for both of RoK and RGM, key pool size is 28000 for RoK and 10000 for RGM.

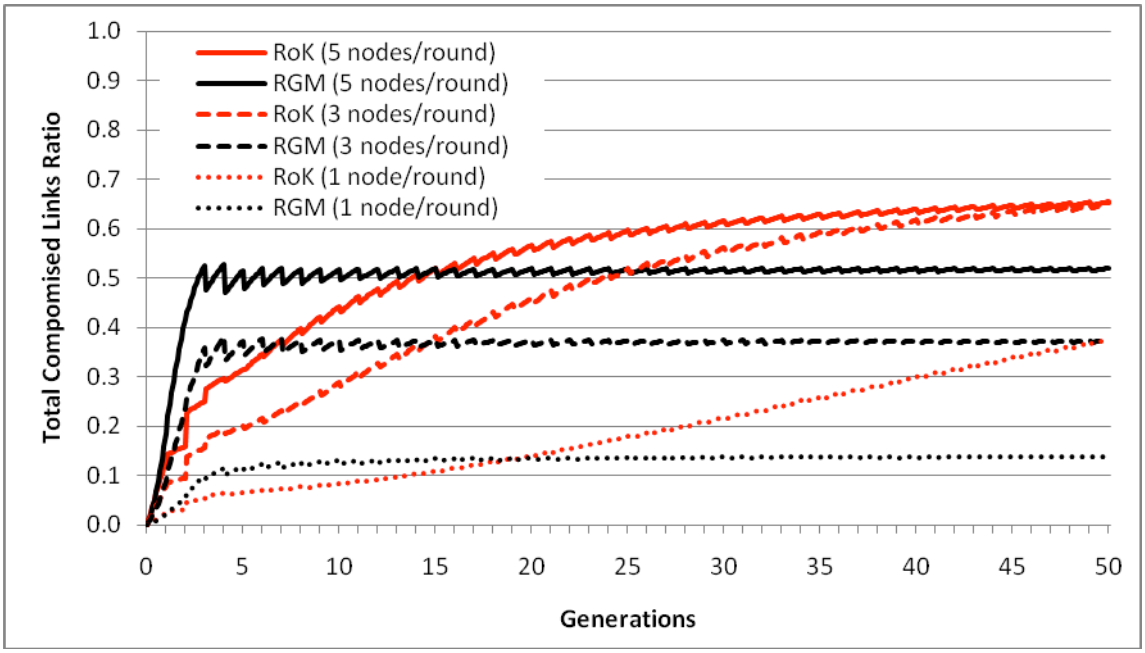

Figure 5.12. Total compromised links ratio of RoK and RGM in case of an eager attacker with capture rates of 1, 3, and 5 per round, memory size is 500 for both of RoK and RGM, key pool size is 28000 for RoK and 10000 for RGM.

Figure 5.13 depicts local connectivity of both schemes in this scenario. Although the keyring size of RoK scheme is increased as compared to previous scenario, the key pool size is also enlarged so that the local connectivity stays the same as RGM scheme at 0.9.

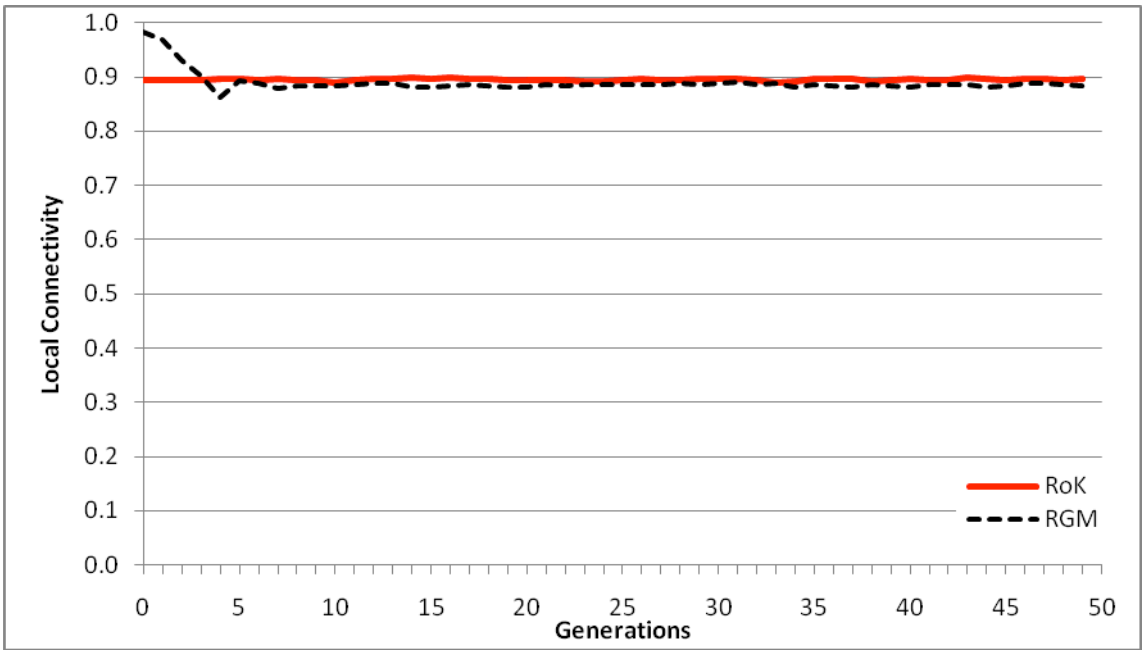

Figure 5.13. Local connectivity of RoK and RGM, memory size is 500 for both of RoK and RGM, key pool size is 28000 for RoK and 10000 for RGM.

5.3.6. Discussions of Memory Requirements in RGM

Abovementioned scenarios show that the proposed RGM scheme has better resiliency than RoK, but it requires more keys in the keyring. Especially Scenario 2 (Section 5.3.4) clearly states this difference; in order to have 0.9 local connectivity, RoK needs 300 keys, whereas RGM needs 500 keys in the keyring. However, as analyzed below, the amount of key memory that RGM uses covers only a small portion of the data memory of the state-of-the-art sensor nodes. Assuming that 128-bit keys are used and another 16 bits are employed for key identification, the amount of key memory

used for RGM is 9000 bytes. In parallel with latest advances in sensor node technology, storage capabilities are increased. For example, MICAz [20] and IRIS [21] have 128 Kbytes of flash memory that can be used for key storage. The flash memory capacity of TinyNode584 [22] is 512 Kbytes. In our RGM scheme, the memory requirement for keys is less than 10 Kbytes that comprises of a small portion of the flash memory of state-of-the-art sensor nodes.

5.4. Discussions and Conclusions

In this section we proposed a random key predistribution scheme, called RGM, for multi-phase wireless sensor networks. In our RGM scheme, each redeployed node comes with a refreshed set of generation keys so that capture of a node that belong to a particular generation has minimal effect on the keys between nodes of other generations. In this way, the value of the information learned by the attacker is reduced and, therefore, the resiliency of the network is improved. Our scheme also takes care of the cryptographic connectivity of the newly deployed nodes with their physical neighbors. The simulative performance analyses show that our scheme performs up to 35% better resiliency under heavy attacks as compared to well-known RoK scheme [9] with the cost of 10% degradation from perfect cryptographic connectivity. Thus, our scheme provides a good tradeoff in favor of security. It is also worthwhile to mention that the connectivity value of 0.9 does not make any node disconnected from the network. The network still operates securely with this rate.

Our extensive analyses also show that even with the same keyring size and the same connectivity level, total resiliency of our RGM scheme is still better than RoK in all capture rates as the attack continues.

We also analyzed *key pool utilization* of both RGM and RoK schemes in all three scenarios in order to further justify the memory requirement of the RGM scheme. Key

pool utilization metric shows the percentage of the keys in the key pools that are used in one or more link keys. The results are given in Table 5.1. As can be seen from this table, key pool utilization of RGM is lower than that of RoK scheme in all scenarios. That is why RGM scheme needs more keys in the keyrings in order to reach an acceptable level of local connectivity. Lower key pool utilization also shows that the attacker learns less valuable information out of the captured nodes in our RGM scheme as compared to rival RoK scheme. Thus, even with more keys per node, the resiliency of RGM is generally better than RoK.

Table 5.1. Key pool utilization

RGM Scenario 1, 2 and 3	RoK Scenario 1	RoK Scenario 2	RoK Scenario 3
11.2%	100%	99.5%	85.9%

6. CONCLUSIONS

In this thesis we proposed three different random key predistribution schemes for wireless sensor networks. Our first scheme uses a combination of XORed and single keys, therefore it has two different key pools and two different keyrings. In our scheme, we introduced the transfer phase which increases the local connectivity. The keys transferred in this phase are not used directly; but, they are XORed prior to usage as link keys. Hence, capture of a neighbor node from which the keys are transferred does not mean compromise of the link. Attacker may still have not enough information about the link key. In this way, we keep the resilience of our scheme under control. Simulation results show that this proposed scheme achieves higher local connectivity than basic scheme while using the same total keyring sizes. However, the mixture of single and XORed keys must be somehow balanced. Resilience performance of our scheme is better at the beginning of the attack as compared to basic scheme. As the attack continues, resilience of our scheme starts to deteriorate after 60% of the links are compromised. However, the resiliency of our scheme never becomes more than 5% worse than the basic scheme.

Our second scheme is an extension to the basic scheme with its additional transfer phase. Our proposed transfer phase increases the cryptographic local connectivity of the network drastically after shared-key discovery phase is performed. When keyring size is 220 keys in basic scheme [4], local connectivity becomes 1.0 after shared-key discovery phase is performed. The same local connectivity is achieved by using keyring size of only 55 keys in our scheme. Moreover, the fraction of communications compromised in our scheme stays lower than basic scheme with shared-key discovery phase up to 50%. Our transfer phase can also be thought as a better alternative to the path-key establishment phase of basic scheme. Our analysis shows that our scheme has the same 1.0 local connectivity with basic scheme with path-key establishment phase although it

uses 20 and 50 fewer keys when the key pool size is 10,000 and 100,000 respectively. It also has the same resiliency performance with basic scheme with path-key establishment phase in this setup. Furthermore, the number of messages sent in order to establish new secure links after failure in shared-key discovery phase is 1 message fewer on the average in transfer phase as compared to path-key establishment phase.

Finally, we proposed a key predistribution scheme for multi-phase wireless sensor networks called RGM (Random Generation Material). In RGM, generation key pool is refreshed in each generation randomly and independent of other generations, so an attacker having information about previous or future states of the pool cannot guess the current state. This is a good property for restricting the amount of knowledge that the attacker knows and, therefore, improving resilience performance. Resilience performance of our RGM scheme is up to 35% better as compared to RoK scheme [9]. While having this success in resilience, local connectivity performance drops by 10%. Despite the deterioration, the network is still connected via secure neighbors and operates without any problem.

7. REFERENCES

[1] Ergun, M., Levi, A., Savas, E. (2009) A resilient key predistribution scheme for multiphase wireless sensor networks. *Proceedings of 24th International Symposium on Computer and Information Sciences, 2009 (ISCIS 2009)*.

[2] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002) Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

[3] Çamtepe, S. A. and Yener, B. (2007) *Key Distribution Mechanisms for Wireless Sensor Networks: a Survey*. Technical Report TR-05-07 Rensselaer Polytechnic Institute, Computer Science Department, March 2005. Book Chapter: Key Management in the book Wireless Sensor Networks Security, IOS Press.

[4] Eschenauer, L. and Gligor, V. D. (2002) A key-management scheme for distributed sensor networks. *Proceedings of the 9th ACM conference on Computer and Communications Security (CCS)*.

[5] Du, W., Deng, J., Han, Y., and Varshney, P. (2003) A pairwise key pre-distribution scheme for wireless sensor networks. *Proceedings of the 10th ACM conference on Computer and Communications Security (CCS)*.

[6] Du, W., Deng, J., Han, Y., Chen, S., and Varshney, P. (2004) A key management scheme for wireless sensor networks using deployment knowledge. *Proceedings* of *IEEE Infocom'04*.

[7] Chan, H., Perrig, A., and Song, D. (2003) Random key predistribution schemes for sensor networks. *Proceedings of the 2003 IEEE Symposium on Security and Privacy*.

[8] Levi, A., Tasci, S. E., Lee, Y. J., Lee, Y. J., Bayramoglu, E., Ergun, M. (2009) Simple, Extensible and Flexible Random Key Predistribution Schemes for Wireless Sensor Networks using Reusable Key Pools. accepted to *Journal of Intelligent Manufacturing*, 2009.

[9] Castelluccia, C. and Spognardi, A. (2007) RoK: A robust key pre-distribution protocol for multi-phase wireless sensor networks. *Proceedings of SecureComm 2007 – 3rd International Conference on Security and Privacy in Communications Networks*.

[10] Blom, R. (1984) An optimal class of symmetric key generation systems. *Proceedings of Eurocrypt 84*.

[11] Zhang, J. and Varadharajan, V. (2009) Wireless sensor network key management survey and taxonomy. *Journal of Network and Computer Applications*, doi:10.1016/j.jnca.2009.10.001.

[12] Zhou, Y., Fang, Y. and Zhang Y. (2008) Securing Wireless Sensor Networks: A Survey. *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 6-28.

[13] Lee, J. C., Leung, V. C. M., Wong, K. H., Cao, J. and Chan, H. C. B. (2007) Key management issues in wireless sensor networks: current proposals and future developments. *IEEE Wireless Communications*, vol. 14, no. 5, pp. 76-84.

[14] Xiao, Y., Rayi, V. K., Sun, B., Du, X., Hu, F., and Galloway, M. (2007) A survey of key management schemes in wireless sensor networks. *Comput. Commun.* 30, 11-12 (Sep. 2007), 2314-2341.

[15] Lamport, L. (1981) Password authentication with insecure communication. *Proceedings of Commun. ACM, vol. 24, no. 11*.

[16] Yilmaz, O. Z., Levi, A., and Savas, E. (2008) Multiphase deployment models for fast self healing in wireless sensor networks. *Proceedings of SECRYPT 2008 - International Conference on Security and Cryptography,* Porto, Portugal.

[17] Kalkan, K., Yilmaz, S., Yilmaz, O. Z., Levi, A. (2009) A highly resilient and zone-based key predistribution protocol for multiphase wireless sensor networks. In *Proceedings of Q2SWinet '09 - 5th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 29-36.

[18] FIPS PUB 180-1. (1995) http://www.itl.nist.gov/fipspubs/fip180-1.htm, retrieved on 09/10/2009.

[19] FIPS PUB 180-2. (2002) http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf, retrieved on 09/10/2009.

[20] MICAz datasheet. (2009) http://www.xbow.com/Products/productdetails.aspx?sid=164, retrieved on 09/10/2009.

[21] IRIS datasheet. (2009) http://www.xbow.com/Products/productdetails.aspx?sid=264, retrieved on 09/10/2009.

[22] TinyNode 584 fact sheet. (2009) http://www.tinynode.com/index.php?id=104, retrieved on 09/10/2009.

[23] RFC 1510, The Kerberos Network Authentication Service (V5). (1993) http://www.ietf.org/rfc/rfc1510.txt, retrieved on 01/28/2010.

[24] RFC 2631, Diffie-Hellman Key Agreement Method. (1999) http://www.ietf.org/rfc/rfc2631.txt, retrieved on 01/28/2010.