Exploration of Key Management Strategies for Modern Internet of Things

Graduate Students: Ayishwarya Narasimhan, Vaishnavi Yekklur Balaji,
Faculty Advisor: Dr. Wei Wang
Department of Computer Science, San Diego State University, USA

Abstract—**In this paper we present different key management schemes used in the wireless sensor networks delineating their similarities and differences and we also present a list of applications where it plays a major role. Internet of Things (IoTs) have attracted intensive interest from both academia and industry due to their wide application in civil and military scenarios. In hostile scenarios, it is very important to protect IoTs from malicious attacks. Securely designing wireless sensor networks is significantly challenging owing to its salient features and resource limitations. The security of a cryptographic system relies mainly on the secrecy of the key it uses. If an attacker is able to find the key, the entire system is broken because the attacker can use the key to decrypt the intercepted ciphertexts to find the original plaintexts. Hence numerous key management schemes have been proposed. The objective of key management is to dynamically establish and maintain secure channels among communicating nodes. The main features of key management in sensor networks include utilization of energy, localized impact of attacks, and scaling to a large number of nodes. The main challenge of key management is to manage the trade-off between acceptable security levels and conserving energy which is needed for network operations.**

***Index Terms*— Key management, Survey, Wireless Sensor Networks, Security**

I. Introduction

The use of sensor networks in a wide variety of sensitive applications ranging from healthcare to warfare is stimulating numerous efforts to secure these networks from malicious attacks. Sensor networks consist of a large number of tiny sensor nodes that collect and process data from the surrounding environment.

Most security protocols used in the sensor networks are based on cryptographic operations that involve keys. To provide confidentiality, an encryption operation requires a key to be fed into an algorithm so that plaintexts can be transformed into ciphertexts. To guarantee packet authenticity, the source node can attach a MAC to each packet, where usually, the MAC is computed by hashing the concatenation of the packet and a key.

The security of a cryptographic system relies mainly on the secrecy of the key it uses. If an attacker can find the key, the entire system is broken because the attacker can use the key to decrypt the intercepted ciphertexts to find the original plaintexts. The attacker can achieve the goal by cryptanalysis on the eavesdropped packets that are transmitted over the wireless medium. Due to the existence of the redundancy of the message source in the real world, the attacker may know more or less information about the key used. Therefore, the sender and receiver may be required to update the key used between them from time to time. In a IoT, some sensor nodes may be captured by an attacker; thus, the key information is accessible to the attacker and can be used to launch other serious attacks. Therefore, a very important issue is how to securely manage the keys between the sender and the receiver.

The objective of key management is to dynamically establish and maintain secure channels among communicating parties [1]. Typically, key management schemes use administrative keys or encryption keys for the secure and efficient redistribution and, at times, generation of the secure channel communication keys (data encryption or session keys) to the communicating parties. Communication keys may be pair-wise keys used to secure a communication channel between two nodes that are in direct or indirect communications [1–4], or they may be group keys shared by multiple nodes [5,6]. Both administrative and communication keys needs to be changed or rekeyed to maintain secrecy and resilience to attacks, failures, or network topology changes. Key management entails the basic functions of analysis, generation, assignment, and distribution of network keys.

In this paper we are going to describe a general key management process and present different key management schemes used in the Wireless Sensor Networks.

II. KEY MANAGEMENT PROCESS

The key management has four basic functions. They are namely analysis, assignment, generation, and distribution of network keys. Key functions are triggered by keying events. The key events include network deployment, node addition, node eviction, or periodic key refresh. Entities with key management responsibilities may include a key server, base stations, and gateway nodes [6].

Manuscript received April 2015. The authors are with the Department of Computer Science, San Diego State University, CA 92182 USA (email: yekklurb@rohan.sdsu.edu, narasimh@rohan.sdsu.edu).

A. Key Analysis

The key requirements are first analyzed in order to determine the required number of keys for each nodes and number of keys for the entire network.

B. Key Assignment

The next step in the key management process is the key assignment. It refers to the mapping of keys to different parties. Key assignment may be static or dynamic depending upon the key management solutions employed. If it is static then each node is assigned with same set of keys throughout the network lifetime whereas if it is dynamic the keys will changed periodically. Mapping decisions are very important as it significantly impact the level of security offered by the key management scheme. This is because suppose if a node is captured it will reveal all the keys to an attacker and if that node possess all network administrative keys, then it will totally jeopardize the security of the entire network. Hence when a node is captured less number of keys should be known to the attacker. But in some key management schemes there will be overlapping keys among nodes to establish network connectivity and hence reducing the number of keys will hamper network connectivity. Random key assignment is the simplest and cheapest solution [1]. However, it limits control over the network security risk in case of multiple node captures. Deployment information such as the location or attack probability has been used to reduce the attacker's probability to uncover more keys by capturing related nodes [3, 4].

C. Key Generation

The administrative keys are generated once or multiple times over lifespan of the network. The generation of communication keys is the responsibility of the communicating parties such as sensor nodes, gateways, or base stations. In all the cases, key generating nodes must be trusted by all key-receiving nodes. In static key pre-distribution schemes, administrative keys are generated by a key server and loaded into nodes prior to deployment. In other schemes, new keys are generated periodically over the lifespan of the network.

D. Key Distribution and Redistribution

The key distribution is the process of delivering the keys to the designated nodes after the keys are generated and assigned to the nodes. If administrative keys are delivered to their destinations after network deployment due to rekeying, other previously distributed administrative keys may be used to encrypt the new keys. The distribution of communication keys usually takes place after the network has been deployed. Communication keys are used for a short period of time and should be regularly updated.

III. TRADITIONAL KEY MANAGEMENT SCHEME

The key management scheme has two major categories. They are namely traditional approach and dynamic approach. In the traditional approach, administrative key remains the same. It is also known as static key management scheme.

A very important issue in sensor networks is how to securely manage the keys between the sender and the receiver. There are two types of keys are used in cryptographic systems. They are symmetric key and asymmetric key. We will discuss in detail about the symmetric and asymmetric key management schemes used in the Sensor Networks.

A. Symmetric Key Management

In a symmetric key system, the sender and receiver share a common key that is kept secret from others. The sender encrypts a plaintext M with the key K by an encryption algorithm E to get a ciphertext is

$$C = E(M, K) \tag{1}$$

After receiving the ciphertext C, the receiver inputs C and the key K into a decryption algorithm D to get the original plaintext is

$$M = D(C, K) \tag{2}$$

Symmetric key algorithms are those which require simple hash, rotation, or scrambling operations. These operations can be easily and efficiently implemented in hardware or software. Examples of symmetric key algorithms are Data Encryption Standard (DES) and Rivest Cipher 5 (RC5). Asymmetric key algorithms require exponential operations over a field modulo a large prime number are more complex than symmetric key operations. Examples of asymmetric key algorithms are Diffie-Hellman or Rivest Shamir Adleman (RSA). Therefore, the symmetric key technology is more suitable on resource constrained low-end devices than the asymmetric key technology.

The main problem in using the symmetric key technology is to how to establish the symmetric key between two nodes. We present here three major approaches to the main problem.

1) Global Approach

One simple approach is to distribute the key globally (global key) to all sensor nodes. This approach makes key secure from external attackers but not from internal attackers because the key can be exposed if any one of the node is compromised.

2) Centralized Approach

Due to the existence of base stations centralized key distribution can be used. Each sensor nodes share a unique with the base station which acts as key distribution center (KDC). If two nodes communicate securely they acquire a shared key from the base station which unicast to each of them. This centralized approach though requires lot of communication overhead. This is so because if two neighboring nodes are at a distant place from the key server they require lot of overhead to do handshake

with key server. In addition, the key server could become a potential point of failure in that the entire network is disabled if the server is corrupted by an attacker.

3) Pre-distribution Approach

A recent approach to key establishment in sensor networks is distributed approach also known as pre-distribution. Here each sensor node is preloaded with the key material in order to establish shared key with other nodes after being deployed in the network terrain.

There are two main features in this pre-distribution approach. One is how to establish the shared key with preloaded key material and the other is how to distribute the key material.

a) Key Agreement Model:

To enable a pair of sensor nodes to share a key, the simplest method is to preload them with a unique shared key before deployment to the sensor networks and later the pair of nodes can use the shared key. To make sure every node has a unique shared key, each node must have N-1 keys, and hence the overall number of keys in a wireless sensor network is

$$N = N(N-1)/2 \quad (3)$$

As the size of the wireless sensor network increases the number of key becomes very large. Hence scalable methods are used in reduce the number of keys in the wireless sensor network.

Blom [7] proposed a key agreement model which is based on maximum separable linear codes which can be used in the sensor networks security design. In this scheme the central authority first constructs (t+1) x N public matrix P over finite field

$$GF(q) = \{1,2,\ldots,q-1\} \quad (4)$$

where q is a prime number. The Blom's scheme has a t-secure property in the sense that in a network with N nodes, the collusion of less than $t + 1$ nodes cannot reveal any key shared by other pairs of nodes. The memory cost per node in Blom's scheme is $t + 1$. Therefore, Blom's scheme trades the security for the memory cost. To guarantee perfect security in a IoT with N nodes, the $(N - 2)$ secure Blom's scheme should be used, which means the memory cost per node is $N - 1$.

b) Symmetric Key Distribution

The key agreement model which we discussed earlier states that in a network of N nodes every pair of nodes in the network has a unique shared key but only a cost of N-1.This is impractical for the sensor networks because of the memory constraint of the sensor nodes and if the network size become large. There are various methods to distribute symmetric key materials. Hence new approaches are used. They are random key material distribution, deterministic key material distribution, and location-based key material distribution.

(1) Random Key Material Distribution:

This new recent approach used is a partial pre-distribution approach which relaxes security requirements.

A typical scheme is the Random Key Pre-Distribution (RKP). Here each node is preloaded with subset of keys called a key ring which are randomly selected from global pool of keys in such a way that any pair of neighboring nodes have at least one shared key with certain probability. After deployment, two neighboring nodes can have a shared key directly or indirectly negotiate a key through a secure path, along which every pair of neighboring nodes has a direct shared key.

The foundation of RKP is the random graph theory. A random graph $G(n, p)$ is a graph of n nodes for which the probability that a link exists between two nodes is p. The graph does not have an edge if p=0 and is fully connected if p=1. If probability p increases then there will be transition from non-connected to full connected graph. PKP exploits this property by having p greater such that all nodes in the network are fully connected. The size of the global key pool and the size of the key subset for an individual node can be tuned to achieve such a property.

The main problem with RKP is the node compromise. In RKP the keys are selected randomly from the key ring and hence there will be reuse of each key by multiple nodes. Suppose if a node is compromised by the attacker the entire key ring may be exposed out of which some keys may be used by the non-compromised nodes. This leads to the failures of the links among those non-compromised nodes.

One scheme to reduce node compromise is q-composite RKP. This scheme follows RPK except that each node is required to share at least q keys with certain probability. Q-Composite RKP can improve network resilience but only if the compromised nodes are less. Suppose if the compromised nodes are more it won't be that effective.

Spatial diversity technique [8] is another method used to improve the network resilience to node capture. A global key is shared by all the anchor nodes and normal nodes, and each normal node is preloaded with a key ring following RKP. Anchor nodes are those powerful nodes in the network which are tamper proof. Each anchor node uses the global key to broadcast several rounds of random nonces at different power levels in its neighborhood. Each sensor node uses the received nonces to rebuild its key ring. Later, all the nodes can follow RKP to establish shared keys with their neighbors. Finally, each node deletes its original key ring and the global key. The spatial diversity introduced by the anchor nodes results in derived key rings which are different from key rings for the nodes which are far away from it. Thus even if a node is compromised the effect will only be in that local area. However this scheme assumes that during initialization phase there won't be any compromise of new nodes. This may not be true for some cases and moreover, the introduction of anchor nodes increases the cost of deploying a IoT.

Another problem of RKP is the lack of authentication because of the reuse of the same key by multiple nodes. To solve the problem, node identity information is use to derive key rings for

the sensor nodes. RKP requires the storage of a key ring by each node to make the network almost connected. In some cases where sensor nodes do not have enough memory resources, this becomes a problem. Hwang and Kim [9] revisited RKP and its variations and proposed to reduce the amount of key material that each node keeps, which means the probability p of direct key-sharing is smaller than that required in RKP. The smaller p cannot guarantee that the network is almost connected but only assures that the network consists of multiple isolated clusters, of which there is one cluster that is the largest and connects most nodes. In this way, less memory cost still achieves certain network connectivity, but the trade-off is that some of the small clusters of nodes are isolated because they do not share keys with the largest one.

(2) Deterministic Key Material Distribution:

The approach used in RKP is probabilistic in nature and hence we cannot guarantee that two neighboring nodes will establish shared keys between them for sure. If a condition happens that some sensor nodes cannot establish a shared key, then they will be isolated. Hence two deterministic approaches were developed.

One approach is to use a complete graph or strongly regular graph to replace the random graph to perform key pre-distribution. The strongly regular graph(n, r, λ, μ), has n nodes, each of which having a degree of r and any pair of which has λ common neighbors when they are adjacent and μ common neighbors when they are non-adjacent. Each link can be assigned with a unique key that is preloaded into the two nodes. In addition to the regular graph, the block design in the set theory can be used in key pre-distribution, in which all the nodes form a complete graph at the network layer. The tool is the balanced incomplete block design (BIBD). A (v, r, λ) BIBD is an arrangement into many blocks of v objects, such that each block contains r distinct objects and every pair of objects occurs in exactly λ blocks. This BIBD design scheme enables authentication.

The second approach is to use a multi-dimensional grid to replace the random graph. Particularly, each sensor node is assigned an ID (n_1, n_2,…,n_k) such that all of the nodes form a k-dimensional grid. Each node is preloaded with some key material such that it can establish shared keys with other nodes along the same dimension.

The scalable key agreement model developed by Zhou and Fang [46] states that two nodes with more than one component mismatch in their IDs can find a path to negotiate an indirect key because all the nodes are organized in the grid.

(3) Location-Based Key Material Distribution:

In the random and deterministic key material distribution, key materials are uniformly distributed throughout the network. Since they are uniformly distributed the probability is very small for two nodes share a direct key at one hop that is the local secure connectivity. Hence a lot of communication overhead is required to establish indirect keys over multiple hops. To improve the local secure connectivity, many researchers propose to involve location information in key establishment.

One approach for location-based key pre-distribution (LBKP) scheme [10] is cell based. In this approach the entire sensor network is divided into square cells. Each cell is associated with a unique t-degree bivariate polynomial. Each sensor node is pre-loaded with shares of the polynomials of its home cell and four other cells adjoining to its home cell horizontally and vertically. After deployment, any two neighboring nodes can establish a pairwise key according to the polynomial approach if they have shares of same polynomial.

In addition to the square cells used in previous schemes, hexagon and triangle grid models also are investigated to improve spatial diversity. Zhou, Zhang, and Fang [11] used a cell-pair-based method, instead of cell based method in which each pair of neighboring cells is associated with a unique t-degree bivariate polynomial or a matrix. It has been shown in that distributing to each pair of neighboring cells instead of each individual cell can reduce the memory cost while increasing the resilience to node. This scheme improves the security and reduces the memory cost significantly while still maintaining highly secure connectivity.

Another approach is not cell-based. In contrast, it estimates the nodes that are supposed to be deployed close to each other and then preloads them with related key materials. In this way, these neighboring nodes may have shared keys between them after deployment. Liu, Ning, and Du [17] established a group-based key pre-distribution framework that may incorporate previous schemes. They divided the entire network into many *deployment groups*. In each group, a specific keying material distribution scheme can be applied to provide the in-group connectivity. They also picked one node from each group, and all of those picked nodes form a cross group. There is also a specific keying material distribution scheme for each *cross group*. Therefore, two nodes from different deployment groups can establish a shared key through a path in a cross group. Similar approach is also taken in [12]. The difference is that all the nodes in one group are preloaded with pairwise keys with each other, and each node can be preloaded with a pairwise key shared with a node in another group. Those preloaded keys can build up a secure path between any two nodes. In [13], a key-position map is used to map a location with a key. If a node is expected to reside in an area according to some probabilistic distribution, it is preloaded with keys corresponding to randomly selected locations around the expected resident point. Therefore, if two nodes are expected to be close to each other, they are more likely to share a common key.

B. Asymmetric Key Management

In an asymmetric key system, each user has a pair of keys {*Ks*, *Kp*}. The user keeps secret his/her *private key*, *Ks*, while publishing his/her public key, *Kp*. When a sender wants to send a plaintext *M* to a receiver, the sender uses the receiver's public key *Kp*, to encrypt *M* to get a ciphertext

$$C = E(M, K_p) \tag{5}$$

Only the receiver can use his/her private key to decrypt the ciphertext and get

$$M = D(C, K_s) \quad (6)$$

only the receiver knows his/her own private key, *Ks*. Because a public key is used here, usually asymmetric key systems are called public key systems.

Asymmetric key is computationally expensive but it is easier to manage and more resilient to node compromise than symmetric key technology. Each node can keep secret its private key and only publish its public key. Hence compromised nodes cannot provide clues to the private keys of non-compromised nodes.

1) Efficient Algorithm Solutions

The most challenging problem here is how to perform asymmetric key algorithms in an efficient way. One approach is to use specific parameters that can speed asymmetric key algorithms without compromising security. Tiny public key (TinyPK) uses RSA-based certificates to authenticate external parties before they can access the network, where the RSA [10] public key is chosen as $e = 3$, such that the signature verification at the sensor side is simplified. Moreover, the Diffie-Hellman algorithm is used in TinyPK to exchange keys between sensor nodes, where the base of exponentiation is chosen as two, such that the exponential operation is simplified.

There are researches going on to find more efficient new asymmetric algorithms than the traditional old algorithms. A more recent and efficient technology is Elliptical Curve Cryptography. ECC achieves same security level with smaller key size as compared to old traditional algorithm. The RSA uses modular exponentiation in the integer rings. The security stems from the difficulty of factorizing large integers. Currently there exist only sub-exponential algorithms to solve the integer factorization problem of finding p and q given a positive integer $n = pq$, where p and q are large pairwise distinct primes. Whereas ECC operates on groups of points over elliptic curves and derives its security from the difficulty of the elliptic curve discrete logarithm problem (ECDLP) of finding an element $x \in GF(q)$,such that $xG = Q$, given a generator G of a finite cyclic point group G over an elliptic curve $E(GF(q))$ and a point Q in the group. The algorithms best known for solving ECDLP are exponential. Hence, ECC can achieve the same level of security as RSA with smaller key sizes. A 163-bit ECC can provide comparable security to the conventional 1024-bit RSA [10].

Computational efficiency is a critical issue if applying asymmetric key technology on a sensor platform. Gura *et al*. [14] evaluated the assembly language implementations of ECC and RSA on the Atmel ATmega128 processor, which is popular for sensor platform such as Crossbow MICA Motes. In their implementations, a 160-bit point multiplication of ECC requires only 0:81 s, whereas 1024-bit RSA public key operation and private key operation require 0.43 s and 10.99 s, respectively.

With same security level as provided by traditional methods such as RSA, ECC achieves with smaller key size plus it offers faster computational efficiency, memory, energy, storage and bandwidth. Hence ECC is better suited for resource constrained devices. In addition to RSA for authentication and Diffie-Hellman for key establishment [15], ECC also is attracting interest for the security design of IoTs due to its efficiency, storage and memory and bandwidth savings.

2) Authenticate Public keys

The second critical issue of applying asymmetric key technology is the authenticity of public keys. A public key should be owned by the node that claims to have it. Otherwise, attackers can easily impersonate any node by claiming its public key and launch the *man-in-the-middle* attack. For example, a malicious node *C* may impersonate node *B* to node *A* and also impersonate *A* to *B*, if *A* and *B* cannot verify the public key of each other. In this way, node *C* can act as an invisible router and learn all the messages between *A* and *B*. The conventional solution to public key authentication is to use a certificate signed by a trustful certificate authority (CA). Therefore, node *B* can send its public key with corresponding certificate to node *A* such that A can verify the correctness of the certificate with the well-known public key of the CA. Node *B* can verify the authenticity of *A*'s public key by following the same procedure.

Though technical advances make the use of asymmetric key technology viable in IoTs, asymmetric key algorithms are more expensive than symmetric key algorithms. The authentication of public keys still may incur high energy consumption because authentication is likely to be performed many times.

Identity-based cryptography removes the necessity of public key certificates and thus saves cost on certificate authentication. Based on this technique, Zhang *et al*., proposed the notion of *location-based* keys by binding private keys of individual nodes to their IDs and their locations. In this solution, the pairwise key of two neighboring nodes is the by-product of their mutual authentication process. In addition, any two nodes that are a multihop apart can establish a shared key on demand, based on their location-based keys. Their solution has perfect resilience to node compromise in that no matter how many nodes are compromised, the location-based keys of non-compromised nodes, as well as their pairwise keys, always remain secure.

Now let us look at Diffie-Hellman algorithm for key management which uses asymmetric keys. It is used to establish shared keys between sensor nodes. Now let us look in detail about Diffe-Hellman algorithm which uses asymmetric keys. It is used to securely exchange keys to encrypt data.

3) DIFFIE-HELLMAN ALGORITHM

The Diffie-Hellman (DH) algorithm is a mathematical algorithm that allows two computers to generate an identical shared secret on both systems, even though those systems may never have communicated with each other before. That shared secret can then be used to securely exchange a cryptographic encryption key. That key then encrypts traffic between the two systems.

The DH algorithm, introduced by Whitfield Diffie and Martin Hellman in 1976, was the first system to utilize public-key" or "asymmetric" cryptographic keys. These systems overcome the difficulties of "private-key" or "symmetric" key systems because asymmetric key management is much easier. In a symmetric key system, both sides of the communication must have identical keys. Securely exchanging those keys has always been an enormous issue. Asymmetric key

systems alleviate that issue because they use two keys: one called the "private key" that the user keeps secret, and one called the "public key" that can be shared with the world and, therefore, easily distributed.

Unfortunately, the advantages of asymmetric key systems are overshadowed by speed—they are extremely slow for any sort of bulk encryption. Today, it is typical practice to use a symmetric system to encrypt the data and an asymmetric system to encrypt the symmetric keys for distribution. That is precisely what Diffie-Hellman is capable of doing and does do when used for key exchange as described here.

a) Process of DH

Diffie-Hellman is not an encryption mechanism as we normally think of them, in that we do not typically use it to encrypt data. Instead, it is a method to securely exchange the keys that encrypt data. DH accomplishes this secure exchange by creating a "shared secret" sometimes called a "Key Encryption Key" or KEK between two devices. The shared secret then encrypts the symmetric key for secure transmission. The symmetric key is sometimes called a Traffic Encryption Key (TEK) or Data Encryption Key (DEK). Therefore, the KEK provides for secure delivery of the TEK, while the TEK provides for secure delivery of the data itself.

The process begins when each side of the communication generates a private key depicted by the letter A (shown in fig below). Each side then generates a public key letter B, which is a derivative of the private key. The two systems then exchange their public keys. Each side of the communication now has its own private key and the other system's public key. Noting that the public key is a derivative of the private key is important because the two keys are mathematically linked. However, in order to trust this system, we must accept that we cannot discern the private key from the public key. Because the public key is, indeed, public and ends up on other systems, the ability to figure out the private key from it would render the system useless. This is one area requiring trust in the mathematical experts.

There is a box labeled "Optional: CA Certifies Public Key." It is not common, but the ability does exist with the Diffie-Hellman protocol to have a Certificate Authority certify that the public key is, indeed, coming from the source you think it is. The purpose of this certification is to prevent Man-In-the-Middle (MIM) attacks. The attack consists of someone intercepting both public keys and forwarding bogus public keys of their own. The "man in the middle" potentially intercepts encrypted traffic, decrypts it, copies or modifies it, re-encrypts it with the bogus key, and forwards it on to its destination. If successful, the parties on each end would have no idea that there is an unauthorized intermediary. It is an extremely difficult attack to pull off outside the laboratory, but it is certainly possible. Properly implemented Certificate Authority systems have the potential to disable the attack.

Once the key exchange is complete, the process continues. The DH protocol generates "shared secrets"—identical cryptographic keys shared by each side of the communication. By running the mathematical operation against your own private key and the other side's public key, you generate a value. When the distant end runs the same operation against your public key and their own private key, they also generate a value. The important point is that the two values generated are identical. They are the "Shared Secret" that can encrypt information between systems.

At this point, the DH operation could be considered complete. The shared secret is a cryptographic key that could encrypt traffic. That is very rare, however, because the shared secret is, by its mathematical nature, an asymmetric key. As with all asymmetric key systems, it is inherently slow. If the two sides are passing very little traffic, the shared secret may encrypt actual data. Any attempt at bulk traffic encryption requires a symmetric key system such as DES, Triple DES, or Advanced Encryption Standard (AES), etc. In most real applications of the Diffie-Hellman protocol (SSL, TLS, SSH, and IPSec, in particular), the shared secret encrypts a symmetric key for one of the symmetric algorithms, transmits it securely, and the distant end decrypts it with the shared secret. Since the symmetric key is a relatively short value (256bits, for example) as compared to bulk data, the shared secret can encrypt and decrypt it very quickly. Speed is less of an issue with short values.

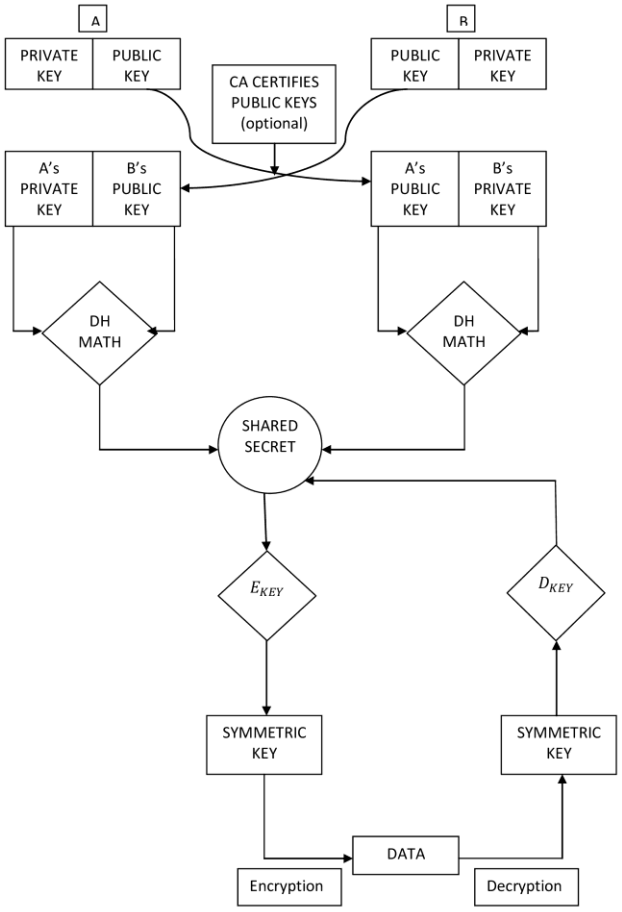

Fig. 1 Diffie-Hellman Key Exchange

Once secure exchange of the symmetric key is complete data encryption and secure communication can occur. Note that changing the symmetric key for increased security is simple at this point. The longer a symmetric key is in use, the easier it is to

perform a successful cryptanalytic attack against it. Therefore, changing keys frequently is important. Both sides of the communication still have the shared secret, and it can be used to encrypt future keys at any time and any frequency desired.

IV. DYNAMIC KEY MANAGEMENT SCHEME

A new emerging category of key management scheme called the Dynamic key management scheme. It changes administrative key periodically on demand or on detection of node capture.

The main advantage of dynamic keying is enhanced network survivability, since any nodes major advantage of dynamic keying is enhanced network survivability, since any captured keys are replaced in a timely manner and the process is known as rekeying. Another advantage is support for network expansion, since upon adding new nodes the probability of node capture does not increase as the keys are rekeyed periodically unlike static keying which use fixed pool of keys. The major challenge in dynamic keying is to design a secure yet efficient rekeying mechanism.

A. *Exclusion-Based Systems*

The one solution to efficient rekeying problem is using exclusion-based systems (EBSs); a combinatorial formulation of the group key management problem. In EBS-based schemes, each node is assigned 'k' keys out of a pool of keys size

$$P = k + m \tag{7}$$

Rekeying occurs periodically or any of the nodes are captured or suspected of being captured. Then replacement keys are generated and they will be encrypted m keys, which are unknown to the captured nodes and finally distributed to the other nodes which know the m keys. The main drawback of EBS based scheme is that even when a small number of nodes collude they may collectively reveal the entire network keys. This is true when the value of *m* is selected to be relatively small to make rekeying feasible in terms of number of messages. The application of EBS was first proposed for key management in sensor networks in [5]. In this scheme nodes were assumed to be anonymous. The sensor network establishes a virtual infrastructure around the base station. The virtual infrastructure provides nodes with post-deployment location information, which is used to select the combination of keys assigned to nodes within each cell. All ID-fewer nodes located in the same coordinate cell are assigned the same EBS key combination and are considered collectively as an EBS group member. Rekeying takes place on the cell level by sending *m* messages of *k* keys each to evict nodes in a specific cell. Although very efficient this scheme does not address collusion and assumes coarse keying granularity due to the assumption of ID-less nodes. A base station is central to the key management scheme with all sensor nodes performing the same keying functions; consequently, this scheme is heterogeneous.

B. *SHELL*

Younis *et al*. proposed SHELL [6] an EBS-based scheme which performs key assignment based on the location to minimize the number of keys revealed by capturing collocated nodes. SHELL uses the EBS framework to perform rekeying within each cluster. Cluster gateways keep track of the key assignment but not the actual keys. These *k* keys assigned to each node are stored by gateways of other clusters called *key generation gateways*. Keys are distributed to nodes by the key generation gateways through their own cluster gateway using an extra cycle of encryption/decryption. SHELL is collusion-resistant. SHELL uses post-deployment location information
in key assignment; collocated nodes share more keys than nodes that are not collocated. SHELL is more energy efficient and relies on the centralized key server to perform rekeying.

TABLE I
COMPARISON OF STATIC AND DYNAMIC KEYING

Features	Static Keying	Dynamic Keying
Network life	Assumed short-lived [18]	Assume long-lived [18]
Key pool	Very large pool; static administrative keys [20]	Small large pool; dynamic administrative keys [20]
Key assignment	Once pre-deployment [19]	Multiple times post deployment [19]
Key generation	Once pre-deployment [18]	Multiple times post deployment [18]
Key distribution	All keys are pre-distributed prior to node deployment [19]	Subsets of keys are re-distributed to some nodes as needed [19]
Handling node capture	Revealed keys are lost [18]	Revealed keys are altered [18]
Re-keying cost	May be practically infeasible with respect to number of messages [18]	Requires few messages [18]
Communication cost	Not applicable for administrative keys [20]	Re-keying overhead [20]
Storage cost	More keys per node [18]	Fewer keys per node [18]
Handling node addition	New node preloaded with keys from static pool; may decrease network resilience to node capture [18]	New node receives new set of keys; other nodes may be rekeyed; less impact on network resilience to node capture [18]
Network resilience	High as long as	High, largely

	number of captured node is small. Once threshold is exceeded, resilience falls sharply [18]	independent of number of nodes captured as long as rekeying is performed in a timely manner [18]
Network connectivity	Less connected due to large key pool; Connectivity improves with increasing number of keys per node [20]	More connected due to small size key pool [20]

V. COMPARISON OF STATIC AND DYNAMIC KEY MANAGEMENT SCHEMES

The static and dynamic schemes share the idea of selecting a random subset of keys for each node out of a pool of keys. Static schemes tend to rely on using a larger key pool to enhance network resilience to attacks, whereas dynamic schemes use a limited pool of keys as well as a limited number of keys per node to achieve better network connectivity. However in the dynamic schemes resilience to attacks is primarily achieved by rekeying.

The Table I summarize primary features of static and dynamic key management schemes and provide a qualitative comparison of both classes of schemes.

VI. SECURITY ISSUES

The two major security issues are authentication, and secure routing.

A. AUTHENTICATION

Authentication is very important because the wireless medium is open to unauthorized attackers who can modify packets. Therefore, every packet must be authenticated before the receiver submits it to higher layer applications.

According to the communication pattern, authentication can be performed in one-hop unicast, multihop unicast, and broadcast.

B. SECURE ROUTING

The goal of networking is to provide an infrastructure for delivering data from a source node to a destination node. Routing protocols are very important because they are responsible to find the path from source to destination and take charge of the delivery of data. In this way, the nodes in a network can collaborate with each other to fulfill various applications deployed in advance. If a routing protocol fails due to malicious attacks then the high level application also fails and hence network becomes useless. Hence secure routing is very critical to guarantee the network functionality in the face of malicious attacks.

VII. APPLICATIONS

The Table II summarizes the applications of various key management schemes along with the algorithms and approaches used.

TABLE II
APPLICATIONS OF KEY MANAGEMENT SCHEMES

	Applications	**Algorithm/Approach**
Symmetric Key Management	Real time video transmission (MPEG)	AES
Symmetric Key Management	Electronic financial transactions (E.g.: ATM), Access Control (E.g.: PIN)	DES
Symmetric Key Management	Online back-up service (E.g.: carbonite), Encrypting text editor (E.g.: coolfish)	Blowfish
Asymmetric Key Management	Unlimited Energy Static Sensor Networks (UESSN)	RSA
Dynamic Key Management	Clustered Sensor Networks	SHELL
Dynamic Key Management	Multimedia Applications	EBS

VIII. FUTURE RESEARCH

The key management schemes for IoT are good for certain applications. However, a new key management scheme protocol needs to be developed with superior extensibility and resilience properties to satisfy needs of adding new nodes to IoT after its deployment. In dynamic key management scheme, a new scenario needs to be proposed to reduce the communication overhead and establish new connection between nodes easily. Most key management schemes work based on node redundancy. While, there is a need to design new solutions by not assuming node redundancy. But this is an open issue where a

definite solution is yet to be found. Also, key distribution is an active research area in IoT.

IX. CONCLUSION

Security has become a major concern for IoT protocol designers because of the wide security-critical applications of IoTs. Most security protocols are based on cryptographic operations which involve encryption keys. A secure management of these keys is one of the most critical elements in the IoT when integrating cryptographic functions into a system, since any security concept will be ineffective if the key management is weak**.** Key management in sensor networks raises interesting research issues. In this survey we presented different key management schemes and their applications. However, there are still many open issues and hence key management in IoT can be researched further in future.

REFERENCES

[1] Eltoweissy, M. Moharrum, M.Mukkamala, R Virginia Tech, Blacksburg, VA, USA "Dynamic Key Management in Sensor Networks" April 2006.

[2] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Sec. and Privacy Symp.*, 2003, pp. 197–213.

[3] W. Du *et al.*, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *Proc. IEEE INFOCOM '04*, Mar. 2004.

[4] D. Liu and P. Ning, "Improving Key Pre-Distribution with Deployment Knowledge in Static Sensor Networks," *ACM Trans. Sensor Networks*, 2005, pp 204–39.

[5] M. Eltoweissy *et al.*, "Group Key Management Scheme for Large-Scale Wireless Sensor Network," *J. Ad Hoc Networks*, Sept. 2005, pp. 796–802.

[6] M. Younis, K. Ghumman, and M. Eltoweissy, "Location aware Combinatorial Key Management Scheme for Clustered Sensor Networks," to appear, *IEEE Trans. Parallel and Distrib. Sys.*, 2006.

[7] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," Advances in Cryptology: Proc. EUROCRYPT'84, Lecture Notes in Computer Science, Springer-Verlag, vol. 209, 1985, pp. 335–38.

[8] F. Anjum, "Location Dependent Key Management Using Random Key-Predistribution in Sensor Networks," *Proc. 5th ACM Wksp. Wireless Security (WiSe'06)*, Los Angeles, Sept. 2006.

[9] D. Liu, P. Ning, and R. Li, "Establishing Pairwise Keys in Distributed Sensor Networks," *ACM Trans. Info. and System Security*, vol. 8, no. 1, Feb. 2005

[10] D. Liu and P. Ning, "Location-Based Pairwise Key Establishments for Relatively Static Sensor Networks," *Proc. 2003 ACM Wksp. Security of Ad Hoc and Sensor Networks(SASN'03)*, Fairfax, VA, Oct. 2003.

[11] Z. Yu and Y. Guan, "A Robust Group-Based Key Management Scheme for Wireless Sensor Networks," *Proc. 2005 IEEE WirelessCommun.and Networking Conf. (WCNC'05)*, New Orleans, LA, Mar. 2005.

[12] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell Sys. Tech. J.*, vol. 28, Oct. 1949, pp. 656–715.

[13] FIPS PUB 46-2, "Data Encryption Standard (DES)," Dec. 1993.

[14] N. Gura *et al.*, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," *Proc. 6th Int'l. Wksp. Cryptographic Hardware and Embedded Systems (CHES'04)*, Lecture Notes in Computer Science, Springer-Verlag, vol. 3156/2004, 2004.

[15] R. Watro *et al.*, "TinyPK: Securing Sensor Networks with Public Key Technology," *Proc. 2nd ACM Wksp. Security of Ad Hoc and Sensor Networks (SASN'04)*, Washington, DC, Oct. 2004.

[16] Y. Zhou and Y. Fang, "A Scalable Key Agreement Scheme for Large Scale Networks," *Proc. 2006 IEEE Int'l. Conf. Networking,Sensing and Control (ICNSC'06)*, Ft. Lauderdale, Florida, Apr. 2006.

[17] D. Liu, P. Ning, and W. Du, "Group-Based Key Pre-Distribution in Wireless Sensor Networks," *Proc. 4th ACM Wksp. WirelessSecurity (WiSe'05)*, Cologne, Germany, Sept. 2005.

[18] Kwang-Jin Paek, Ui-Sung Song, Hye-Young Kim, Jongwan Kim, "Energy Effiecient Key Management (EEKM) protocol for Large-scale distributed sensor networks," Journal of information science and engineering 24, 1837 – 1858(2008).

[19] Yang Xiao, "Security in sensor Networks" 2007.

[20] Priyanka manhas, Parminder Kaur, "Secure network coding approach with distributed reprogramming protocol for cluster based ad-hoc networks in dynamic key management of wireless sensor networks," International Journal of advanced research in Computer and communication engineering, December 2013.