Reading Notes: IoT Security

Panos Papadimitratos

December 14, 2015

Contents

1 Introduction (Slides 1-5)

The concept of IoT includes a great gamut of wireless devices (e.g., smartphones, PDAs and computers). It is important that the communications between these devices are secured. This implies services such as authentication, integrity, confidentiality and non-repudiation (see introductory lectures). To offer these services we rely on security mechanisms such as digital signatures, encryption/decryption algorithms (symmetric and asymmetric) and Message Authentication Codes (MACs). Nevertheless, when designing protocols for such emerging networks, various issues must be taken into consideration.

2 Public-key Approach (Slides 6-12)

In asymmetric cryptography, each node has an identity (e.g., A for Alice) and possesses a key-pair (i.e., a private key k and the corresponding public key K). This approach allows the establishment of secure communication channels between any pair of nodes. Nonetheless, as we saw, public keys must be associated with the identities of the nodes.

Slide 7 presents an example of a secure communication channel between Alice and Bob. Alice sends to Bob a message ($text$) a nonce n_A and her identity, A, signed with her private key k_A. When Bob uses the public key of Alice (K_A) to verify the signature. This way, he can be sure that the message originated by Alice. Please note that at this step of the protocol we are not interested in the confidentiality of the message. Bob replies back to Alice with a confidential message $sec - text$ encrypted with the public key of Alice (K_A). Moreover, he includes a digital signature, under k_b, over the whole message (including the identity of Alice and the nonce he received from Alice from the previous message). Since it is only Alice that knows her private key it is only her that can decrypt $sec - text$. This is just an example of a secure communication channel.

But how can Alice and Bob trust each-others public keys? This is when Trusted Third Parties such as Certification Authorities come to the game. As we saw during the introductory lectures, CAs certify the public keys of communicating entities by issuing certificates. A certificate is a digital structure that binds the identity of an entity with her public key. It also includes additional pieces of information such as the lifetime of the certificate. Certificates are signed by CAs so that nodes can verify them.

Now, another questions arises; *How can nodes obtain the certificates of devices?*. There are different approaches to achieve this and this is what slide 9 is about. One way would be to have users obtain the certificates of other users over wire-line networks (remember that this lecture focuses on wireless communications). Another approach would be to pre-install certificates on the wireless devices (similar to the way browsers work).

Slide 10 presents some examples of wireless networks that leverage certificates and certification authorities and *slide 11* presents the pieces of certificates include.

As we already know, the private keys must be kept secret and secure. Nonetheless, this is not always the case as devices can be compromised and thus, private keys might be stolen. If this happens, we need ways to invalidate

a certificate when the corresponding private key is stolen. The process of revoking a certificate is known as *Certificate Revocation*. CAs publish the certificates that must be revoked in a Certificate Revocation List. This list contains all the certificates that are no longer valid. Nevertheless, since IoT networks are expected to include a great number of devices these lists must be disseminated and updated in an efficient manner.

2.1 Public key cryptography - Practical aspects (Slide 13)

Although public key cryptography and CAs offer all the necessary means to establish secure communication links, their applicability to the wireless domain is not straight forward. One problem has to do that there exist multiple CAs for different administrative domains (for example KTH and Stockholm University belong to different domains and thus, they could have different CAs). A way to overcome this problem is by employing certificate chains as we saw during the introductory lectures. Moreover, public key cryptography is a costly operation and this is why we need to establish symmetric keys.

3 Symmetric key establishment (Slide 14)

There are different ways to establish symmetric keys. One is by using key agreement protocols like Diffie-Hellman (covered during the introductory lectures). Another approach is to have one node decide a session key and transport it to the other party by encrypting it with the other party's public key (so that we ensure its confidentiality)

4 Key Agreement

4.1 Key Agreement (Slide 15)

This slide revisits the Diffie-Hellman protocol.

4.2 Key Transport (Slide 16)

This slide presents three-pass key transport protocol. Alice generates a symmetric key K_{AB} and transfers it to Bob (encrypted with his public key). Moreover, the first message includes digital signatures and nonces that ensure that the message is fresh (i.e., it is not part of some prior communications). Similarly, Bob generates another symmetric key (K_{BA}) and sends it to A. By combining these keys, Alice and Bob can agree on a symmetric key that will be use to protect their communications (usually these keys are combined by some key generation function).

4.3 Key transport and establishment (Slide 17)

Other approaches of establishing symmetric keys is by leveraging Key Distribution Centers like in Kerberos (see Unauthorized Access slides).

5 Hash Chains (Slide 18-22)

As their name implies, hash chains are generated by using hash functions (see introductory lectures). To construct a hash chain a node generates a random number r. This number is hashed k times and thus, k chain elements are generated. The first element (H_0) is known as the *trust anchor* and it is signed by the node that generates the chain. The remaining $k-1$ elements can be used for authenticating messages. The sender splits time in time intervals. Each element of the hash chain is assigned to a time interval in **reverse order**. For examples messages sent during time interval $t = 1$ will be authenticated[1] with H_{k-1}, messages sent during the interval $t = 2$ will be authenticated with H_{k-2} and so on. After d time intervals the sender broadcasts elements of the chain. For example at $time = 5$ (so here we have $d = 5$) the sender can broadcast H_{k-1}. This way, the receiver can now authenticate (i.e., examine the Message Authentication Code) of all the messages it received at $t = 2$. At $t = 10$ the sender will publish H_{k-2} and so on. Finally, the sender publishes H_0. This kind of validation is known as *a posteriori message validation.*

For this protocol to work it is required that the sender and the receivers are synchronized.

6 What if no CA is available? (Slides 24-25)

Although many schemes rely on Certification Authorities, reaching a Trusted Third party is not always possible, for example in short-range communications without Internet access. Therefore, in such scenarios, the main challenge for establishing a session key is to protect the protocols against Man-in-the-Middle attacks, where the actual communication goes through an undetected attacker.

However, specific characteristics of the network or the mobile application can be exploited to enhance the security of the protocols. First, the devices are usually operated by human beings that can assist the security association with their knowledge or actions. Secondly, short-range communication (especially on line of sight) can decrease the probability of having other devices in the area, thus reducing the risk of an adversary.

7 Leveraging on the users (Slides 26-34)

Bluetooth is an example of a system that leverages the user during the security association. This wireless technology standard for exchanging data over short distances is a a packet-based protocol with a master-slave structure, where the master is usually the initiator of the connection but the devices can switch roles. Bluetooth uses the frequency-hopping spread spectrum method for transmitting radio signals by rapidly switching a carrier among many frequency channels, using a pseudorandom sequence known to both transmitter and receiver. Although this approach increases the stability in case of channels with high interference and decreases the probability of encountering an attacker, the protocol still suffers of security issues, especially regarding the mutual device authentication and the confidentiality of the communication.

[1] To authenticate a message we have to compute a Message Authentication Code

First, an *initialization key* is generated. This temporary key is a function of a random number IN_RAND, which is generated by the device that initiated the communication, e.g., A, and sent to the other device (B). It is also a function of a shared PIN; the length of the PIN can vary between 8 and 128 bits but usually 4 digits are used. Although the PIN should be entered in both devices, if one of them does not have an input interface, a fixed default PIN is used and, often, the value is 0000. Both devices now share an initialization key, which will be used to agree on a new, semi-permanent key (called the *link key*) that will be kept on the devices for future communication. Both parties first generate a random number $RAND_A$ and $RAND_B$. These random numbers are encrypted with the initialization key and sent to the other device.

Now they both can calculate LK_K_A and LK_K_B, using also the device addresses BD_ADDR_A and BD_ADDR_B, respectively. The combination key K_{link} is the XOR of LK_K_A and LK_K_B. After the generation of the link key, the old temporary initialization key is discarded and a mutual authentication is started with the exchanged link key.

The authentication scheme is based upon a challenge-response protocol. After B has sent its Bluetooth address, A generates a random number AU_RAND, that is the *challenge*, and sends this to B. Both devices now calculate a response $SRES = E_1(BD\_ADDR, K_{link}, AU\_RAND)$, where the function E_1 is a computationally secure authentication code. B then sends its response to A that checks it against the pre-calculated value. The authentication scheme also has the value ACO (Authenticated Ciphering Offset) as result, which will be used for the generation of the encryption key.

For agreeing upon the encryption key, A generates a random number EN_RAND and sends this to B. Both devices then generate the encryption key $K_{enc} = E_3(EN\_RAND, K_{link}, ACO)$. Finally, the encryption key K_{enc} is fed to the encryption scheme E_0 together with the Bluetooth address and the clock of the master. The result is the key stream K cipher. The master clock is used in order to make the key stream harder to guess.

Slide 31 highlights the weaknesses of the aforementioned PIN based key exchange protocol.

Slides 32-33 present a password based key establishment protocol. Both Alice and Bob share a password π and q that is a publicly known system parameter. Based on π and q they compute $g_\pi = (h(\pi))^2 mod\ q$. Then, they select two random numbers x for Alice and y for Bob. With these numbers, they compute and exchange W_A and W_B respectively (note that this resembles the Diffie-Hellman protocol). With W_A and W_B they both compute z. Later on, they prove to each-other that they know the same z which can then be used to derive a common session key (not shown in the slides).

8 Leveraging on the wireless link/network (Slide 35)

An other option to implement authentication is to leverage off-line channels. This way, users can use one channel (e.g., infrared or bluetooth) to establish security associations that can be used to authenticate users over another channel (e.g., wireless). Nevertheless, caution has to be taken to assure that adversaries

cannot interfere with the local channel (i.e., infrared or bluetooth) for example by launching a MITM attack.

9 Wireless Sensor Networks

9.1 Introduction (Slides 37-39)

Recent advances in Micro-electro-mechanical systems (MEMS) and wireless communications technologies made it possible to build small devices that can run autonomously and be deployed in a large-scale, low power, inexpensive manner that is acceptable to many commercial and government users. These devices can be used to form a new class of distributed networking, namely *Wireless Sensor Networks* (WSNs). Sensor networks' configurations range from very flat, with few command nodes as *base stations*, *sinks* or *cluster controllers*, to hierarchical nets consisting of multiple networks layered according to operational or technical requirements. The robustness and reliability of such networks have improved to the point that enabled their proliferation to a wide range of applications for a variety of tasks; from battlefield surveillance and reconnaissance to other risk-associated applications such as environments monitoring and industrial controls.

The sensors are embedded devices, that are networked via wireless media, integrated with a physical environment and are capable of acquiring signals, processing the signals, communicating and performing simple computational tasks. Common functionalities of WSNs are broadcasting and multicasting, routing, forwarding, and route maintenance. By correlating sensor output of multiple nodes, the WSN as a whole can provide such functionalities that an individual node cannot. There is no central computer that performs the coordination tasks; instead, the network itself is a computer and users interact with it directly, possible in *interactive* or *proactive* paradigms.

Embedded systems are not new, but sensor networks greatly extend their capabilities. They are self-configuring systems and can be deployed spatially and temporally in various environments depending on the application demands. Their position need not be engineered or pre-determined. Often such networks lead to low-cost and reliable implementations. They can operate over large time scales and quick response times are feasible for demanding sensing loops. The reliability and overall throughput of sensing is improved by using *concurrent* operations and *redundant* routing paths. Redundancy is a strong feature that allows the development of fault-tolerant systems that degrade gracefully under exceptional circumstances.

The strength of distributed sensor networks lies in their flexibility and universality. The wide range of targeted applications forces them to provide some attractive characteristics, as shown in Table 1. A unique feature is the *cooperative effort* of sensor nodes. Sensors can detect multiple input modalities, and combining such values provides new information that cannot be sensed directly. They operate in complementary, collaborative mode and data gathered by individual sensors are integrated to synthesize new information using data fusion techniques. Instead of sending the raw data to the nodes responsible for the fusion, sensors use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data.

Table 1: *Attributes of distributed sensor networks*

Inherent Attributes	**Description::**
Sensors	*Size*: small (e.g., mobile phone sensors), large (e.g., radars, satellites) *Composition*: homogeneous, heterogeneous *Spatial Coverage*: dense, sparse *Deployment*: fixed (e.g., factory networks), ad-hoc *Dynamics*: stationary, mobile (e.g., vehicles)
Sensing Components	*Extent*: distr. (monitoring), local. (tracking) *Nature*: cooperative (e.g., intrusion det.), competitive (e.g., military)
Operating Environment	Benign, Unknown or Chaotic
Communication	*Networking*: wireless through multiple hops *Bandwidth*: high, low
Processing Architecture	Centralized, Distributed, Hybrid

Furthermore, sensor nodes are densely deployed and have short communication range. Hence, *multi-hop communication* is used which consumes less power than the traditional single-hop communication. Transmission power levels are kept low and signal propagation effects experienced in long-distance wireless communications are overcome. Finally, WSNs have the ability to dynamically adapt to changing environments. Adaptation mechanisms can respond to changes in network topologies or can cause the network to shift between different modes of operation.

9.2 What can go Wrong: Sensor Networks Limitations (Slides 40-43)

Although wireless sensor networks have an inherent ad-hoc nature, they pose a number of new constraints and limitations compared to traditional computer networks. Therefore, existing security solutions cannot be directly employed; instead, they have to be adapted to the characteristics of this special type of networking.

- **Limited Resources.** The extreme resource limitations of sensor devices pose considerable challenges to resource-hungry security mechanisms. In order to implement effective approaches, a certain amount of data memory, code space, and energy is required. However, these resources are very limited in a tiny wireless sensor and the trend has been to increase the lifetime of such devices by decreasing their memory, CPU, and radio bandwidth.

- **Unreliable Communication.** Certainly, the very nature of the wireless communication medium, which is inherently insecure, poses another threat to sensor security. Unlike wired networks, where a device has to be physically connected to the medium, the wireless medium is open and accessible to anyone. Therefore, any transmission can easily be intercepted,

altered, or replayed by an adversary. The wireless medium also allows an attacker to easily intercept valid packets and inject malicious ones.

- **Ad-hoc Deployment and Immense Scale.** Node mobility, node failures, and environmental obstructions cause a high degree of dynamics in WSNs. This includes frequent topology changes and network partitions. One of the most attractive characteristics is their ability to be deployed in large areas, with thousands or millions of nodes, without any prior knowledge on their position. It is crucial, therefore, that security schemes can operate within this dynamic environment.
- **Unattended Operation.** Depending on the application, nodes may be left unattended for long periods of time. This exposes them to physical attacks. Motes face the possibility of destruction or (perhaps worse) capture and *compromise* by attackers. Node compromise occurs when an attacker gains control of a node in the network after deployment. Not only are these adversaries capable of physically damaging the device, rendering it non-functional, but they can also alter device characteristics/mechanisms to send out data readings of their choice. Once in control, the attacker can alter the node to listen to information in the network, input malicious data or perform a variety of attacks. She may also disassemble the node and extract information vital to the network's security such as routing tables, data, and cryptographic keys.

10 Attacking WSNs (Slides 44-54)

Attacks can be categorized as *outsider* (external) and *insider* (internal) depending on whether the attacker is a legitimate node of the network or not. In the first case, the intruder node is not an authorized participant of the sensor network and can be used to launch passive attacks. Usually authentication and encryption techniques prevent such attackers from gaining any special access to the network.

Table 2: *Functions and Effects of External Attacks*

Functions	**Effects**
Initiate attacks without authentication	Gather & Steal information
Monitor & Eavesdrop traffic	Compromise privacy/confidentiality
Jam communications	WSN's resource consumption
Trigger DoS Attacks	WSN functionality degradation

However, as the communication takes place over a wireless channel, a passive attacker can easily *eavesdrop* (Slide 44) on the network's radio frequency range in an attempt to steal private or sensitive information. The adversary can also *alter* or *spoof* (Slides 45-46) packets to infringe on the authenticity of communication or *inject* interfering wireless signals to jam it. Another form of outsider attack is to *disable* sensor nodes. An attacker can inject useless packets to drain the receiver's battery, or she can capture and physically destroy nodes.

In the case of an insider attack, intrusions are performed by *compromised* nodes in the WSN. In contrast to disabled nodes, compromised nodes activity

Table 3: *Functions and Effects of Internal Attacks*

Functions	**Effects**
Inject faulty data into the WSN	Accessing & revealing WSN codes/keys
Impersonation	Data alteration
Unauthorized access & modification of resources and data streams	Obstructing/cutting of nodes from their neighbors (*selective reporting*)
Create holes in security protocols	Patial/Total degradation/disruption
Overload the WSN	Denial of Service
Executing malicious exploits or use of legitimate cryptographic content	High threat to the functional efficiency of the whole network

seek to disrupt or paralyze the network. An adversary by physically capturing the node and reading its memory, can obtain its key material and forge network messages. As shown in Table 3, having access to legitimate keys can give the attacker the ability to launch several kinds of attacks, such as *false data injection* and *selective reporting*, without easily being detected. Overall, insider attacks constitute the main security challenge in sensor networks.

Furthermore, the most important and hard to identify security breaches target the *network* layer. Network layer is responsible for routing packets across multiple nodes. Wireless sensor nodes do not need to communicate directly with the nearest high-power control tower or base station, but only with their local peers. Thus, every node in a sensor network must assume routing responsibilities. WSNs are particularly vulnerable to routing attacks because every node is essentially a router. There are many sophisticated attacks that exploit specific characteristics of the routing protocols in order to affect the created topology and gain access to the routed information. These attacks have been classified and described analytically by Karlof and Wagner ("*Secure routing in wireless sensor networks: attacks and countermeasures*").

- *HELLO Flood Attack*. In many sensor network protocols, nodes need to broadcast HELLO packets for neighbor discovery purposes. In a HELLO flood attack, an attacker can send or replay such messages with high transmission power. In this way, she creates an illusion of being a neighbor to many nodes and can disrupt the construction of the underlying routing tree, facilitating further types of attacks.

- *Sinkhole Attack*. The sinkhole attack is a particularly severe attack that prevents the base station from obtaining complete and correct sensing data, thus forming a serious threat to higher-layer applications. Using this attack, an adversary can attract nearly all the traffic from a particular area. Typically, sinkhole attacks work by making a malicious node look especially attractive to surrounding nodes with respect to the underling routing algorithm (e.g., advertise lower distance to the sink as is the case in Slide 48).

- *Wormhole Attack*. The wormhole attack constitutes a threat against the routing control plane of a network which is particularly challenging to detect and prevent. In this kind of attack, an adversary can convince

distant nodes that are only one or two hops away via the wormhole. The wormhole is a low-latency link that is formed in such a way so that packets can travel from one end to the other faster than they would normally do via a multi-hop route. This would confuse the network routing mechanisms (Slides 49-50).

- *Sybil Attack*. The sybil attack targets fault tolerant schemes such as distributed storage, multi-hop routing, and topology maintenance. In this kind of attack, an adversary uses a malicious device to create a large number of pseudonymous entities in order to gain influence in the network traffic. We refer to these pseudo-nodes as Sybil nodes. The ID of these nodes can be the result of "fake" network additions or duplication of already existing legitimate identities.

Detecting the existence of a sinkhole or wormhole in the network is considered to be the most challenging task since adversaries can operate in a *stealthy* mode (Slide 52). By stealthiness we highlight the attacker's advantage in misleading the routing protocol while operating in a legitimate, undetectable manner. While she manages to fool benign nodes, her actions are masqueraded by the existing routing policies.

11 Securing WSNs (Slides 55-75)

Sensor networks are a special type of distributed networking. They share some commonalities with a typical computer network, but also pose unique requirements and constraints of their own. Therefore, we can think of their security goals as encompassing both the typical network requirements and the unique requirements suited solely to WSNs.

Their security profile must be enhanced with attributes including Confidentiality, Integrity, Data Freshness, Authentication, and Availability. All network models (including communication, routing, and security) allow provisions for implementing these properties in order to assure protection against the kind of attacks to which these types of networks are vulnerable to.

- *Data Confidentiality*. Confidentiality is the ability to conceal network traffic from a passive attacker so that any message communicated via the sensor network remains secret. In many applications (e.g., key distribution) nodes communicate highly sensitive data. The standard approach for keeping sensitive data secret is to encrypt them with a secret key that only intended receivers possess, hence achieving confidentiality. Since public-key cryptography is too expensive to be used in the resource constrained sensor networks, most of the proposed protocols use *symmetric* key encryption methods. Furthermore, while confidentiality guarantees the security of communications inside the network it does not prevent the misuse of information reaching the base station. Hence, it must also be coupled with the right control policies so that only authorized users can have access to confidential information.

- *Data Authentication & Integrity*. In a sensor network, an attacker can easily inject additional false messages, so the receiver needs to make sure

that the data used in any decision-making process are valid. Integrity and authentication is necessary to enable sensor nodes to detect modified, injected, or replayed packets. Data authentication is usually achieved through symmetric or asymmetric mechanisms where sending and receiving nodes share secret keys. However, authentication alone does not solve the problem of node takeovers as compromised nodes can still authenticate themselves to the network. Hence authentication mechanisms should be "collective" and aim at securing the entire network. Using intrusion detection techniques we may be able to locate the compromised nodes and start appropriate revoking procedures.

- *Data Availability*. Availability determines whether a node has the ability to use the resources and whether the network is available for the messages to communicate. A sensor network should be robust against various security attacks, and if an attack succeeds, its impact should be minimized.
- *Data Freshness*. Data freshness implies that the data is recent, and it ensures that an adversary has not replayed any old messages. To solve this problem a nonce, like *sequence numbers*, can be added into the packets for sorting the old ones out.
- *Non repudiation*. Entities cannot deny prior actions.
- *Privacy*. Protect the identity and/or the location of the nodes.

11.1 Key Management (Slides 57-64)

Several intrusion prevention techniques have been proposed for sensor networks over the last few years. Encryption and authentication algorithms have been extensively studied aiming to protect information from being revealed to an unauthorized party and guarantee its integral delivery to the base station. Other specific services, like localization, aggregation, cluster formation and time synchronization have also been secured under certain conditions. Some security protocols have been also designed with the goal of protecting a sensor network against specific attacks like selective forwarding, sinkhole or wormhole attacks.

Such schemes make use of various cryptographic primitives such as encryption, hash functions, MACs, digital signatures, etc. The security level of such encryption schemes depends mainly on the ability of the attacker to figure out the key (it has to be computationally hard to extract it) and systematically obtain plaintexts. Therefore, there is a need for efficient key management protocols. Since public key methods are rather expensive (even though several implementations are available now), the focus is on using symmetric key approaches. The questions that remain are *how many keys, shared by which nodes, why, how, for how long*?

The answers to these questions come from the key distribution, discovery, establishment and update protocols. One naive solution is to have a single network-wide private key shared by all nodes in the network (provides low security level). Also, encrypting data at their source sensor node, with a symmetric shared with the sink, is a straightforward confidentiality mechanism. However, they do not fully address the problem at hand since an adversary can actively exploit the poor physical protection of nodes to access the node memory contents and extract the symmetric key used for encryption.

Another dominant method of key distribution is the well-known *random key pre-distribution* in which each node can share private symmetric keys with all the other nodes in the network. Let m denote the number of distinct cryptographic keys that can be stored on a sensor node. The basic scheme works as follows. Before sensor nodes are deployed, an *initialization phase* is performed. In the initialization phase, the basic scheme picks a random pool (set) of keys S out of the total possible key space. For each node, m keys are randomly selected from the key pool S and stored into the node's memory. This set of m keys is called the node's *key ring*. The number of keys in the key pool, |S|, is chosen such that two random subsets of size m in S will share at least one key with some probability p.

After the sensor nodes are deployed, a *key-setup phase* is performed. The nodes first perform key-discovery to find out with which of their neighbors they share a key. Such key discovery can be performed by assigning a short identifier to each key prior to deployment, and having each node broadcast its set of identifiers. Alternatively, the set of keys in the key ring of a node could be related to its ID via a pseudorandom function. In this case, each node need only broadcast its ID to its neighbors. While these broadcast-based key discovery methods are straightforward to implement, they have the disadvantage that a casual eavesdropper can identify the key sets of all the nodes in a network and thus pick an optimal set of nodes to compromise in order to discover a large subset of the key pool S. A more secure, but slower, method of key discovery could utilize client puzzles. Each node could issue m client puzzles (one for each of the m keys) to each neighboring node. Any node that responds with the correct answer to the client puzzle is thus identified as knowing the associated key.

Nodes which discover that they contain a shared key in their key rings then verify that their neighbor actually holds the key through a challenge-response protocol. The shared key then becomes the key for that link.

After key-setup is complete, a connected graph of secure links is formed. In some applications, this infrastructure is sufficient for normal operation. For example, if the communications in the network consists of mostly node-to base or base-to-node communications, this infrastructure might be adequate assuming the base station can form secure links with all its neighboring nodes. However, if frequent node-to-node communications are performed, then the network needs each node to be able to form a secure link with all of its neighbors to improve long-term communication efficiency. In this case, nodes can set up path keys with nodes in their vicinity whom they did not happen to share keys with in their key rings. If the graph is connected, a path can be found from a source node to its neighbor. The source node can then generate a path key and send it securely via the path to the target node.

11.2 En route re-encryption & Key refreshing (Slides 65-75)

Consider a *parasitic* adversary (Slides 65-69) that seeks to exploit a WSNs' data rather that to disrupt, degrade, or prevent its operation. A parasitic adversary aims at obtaining measurements with the least expenditure of its own resources and disruption of the WSN it "attaches" itself to. Essentially,

the longer the parasitic relation of the adversary with a fully functioning WSN remains unnoticed, the more successful the parasitic adversary remains.

The parasitic adversary can physically access data stored at sensor nodes and retrieve their cryptographic keys (i.e., node compromise). It is *unobtrusive*, that is, cannot modify the implemented protocols stored in ROM. Furthermore, it can utilize its resources intelligently. Mobility gives the adversary the ability to be in the proximity of different nodes for periods of time during which it either compromises the nodes, or obtains snapshots of their measurement histories, or intercept messages sent from nodes within its receiving range.

The goal is to ensure the network confidentiality against such adversaries (in an energy efficient manner), that is, prevent any unauthorized access to data collected by a WSN. More specifically, the aimed property is to prevent measurement disclosure from a given fraction of sensor nodes over a period of time, meaningful with respect to the system and application. En route re-encryption schemes can achieve this; with the source node always encrypting the data (with the private key shared with the sink) and with relaying nodes *en route* to the sink flipping a coin to "decide" whether to perform re-encryption or not. To defend against the progressive compromise of an increasing number of nodes, $K_{i,BS}$ keys should be refreshed. The sink is typically unaware of which nodes are already compromised. Thus, it selects randomly a node S to refresh. This selection is, in general, made among the data source nodes of interest and all the intermediate nodes that connect those sources to the sink. In other words, the refreshing effort focuses on the same part of the network that is meaningful for the adversary to target. Throughout such *key refreshing* protocols, sensor nodes randomly elect a point in time to generate a new symmetric key which they encrypt it with the sink public key and (then) they send it as if it were a regular data measurement (Slide 72).

For a sensor measurement m, the source node S generates a nonce n for the communication with the sink (BS). It also calculates $H = MAC(K_{S,BS}, m, n, S)$. Then, it encrypts m, n and H with $K_{S,BS}$ and sends the corresponding ciphertext (along with its ID S) to the first relaying node in the path towards the sink. Upon receipt of such a packet p, the relaying node R generates a random number $X \epsilon [0, 1]$. If x is bigger than a protocol-specific parameter (defined in an initialization phase), R simply forwards p to the next relaying node in the path. Otherwise, it encrypts p with $K_{R,BS}$, appends its own identity and then forwards the new packet to the next relaying node. At the end, the sink upon reception of the final packet it will perform all the necessary decryptions (using the appropriate keys), it will check the validity of the sources' MAC and, if successful, will deliver m to the WSN user (Slide 71).