ABSTRACT

Title of Thesis: ON THE DISTRIBUTED REVOCATION OF NODES IN SENSOR NETWORKS

Degree Candidate: Gautam Muralidharan

Degree and Year: Master of Science, 2004

Thesis Directed By: Professor Virgil Gligor
Department of Electrical and Computer Engineering

Revocation in sensor networks is a challenging problem because asymmetric key cryptosystems are unsuitable for use in resource constrained sensor nodes. We present some properties of node revocation in distributed sensor networks (DSN) and explain their implementation challenges. We illustrate these challenges by analyzing prior work in centralized and distributed revocation schemes for DSNs. We present a distributed revocation scheme for DSNs based on voting, that provides revocation vote authenticity, improved resilience to node replication, and well-defined policies for revocation. We also present the correctness properties of our scheme and prove its robustness

in the context of the various problems identified in distributed revocation. Further, we explain why tracking the degree of connectivity of sensor nodes in a DSN is a complex problem and identify its role in solving the distributed revocation problem.

ON THE DISTRIBUTED REVOCATION OF NODES IN SENSOR NETWORKS

by

Gautam Muralidharan

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2004

Advisory Committee:

Professor Virgil D. Gligor, Chairman/Advisor
Professor Charles B. Silio
Professor Gang Qu

DEDICATION

To my parents, Muralidharan and Srimathi for making me possible,

and to Aneetha for her invaluable love and support.

ACKNOWLEDGEMENTS

I am grateful to my advisor Dr.Virgil Gligor for his advice, support and encouragement. I would also like to thank Dr.Charles Silio and Dr.Gang Qu for agreeing to serve in my committee and for providing feedback.

Also, I would like to thank my office-mates Radostina, Rakesh, Farshad, and Gelareh for providing a creative and refreshing work environment.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Distributed Sensor Networks (DSN) are ad-hoc networks that consist of a large number of low-power sensor nodes that communicate in short distances through wireless links [16]. Sensor networks are used for a wide range of applications like health monitoring, data acquisition in adverse environments and military operations. The desirable features of sensor networks have attracted many researchers to develop protocols and algorithms that can fulfill the requirements of these applications [11, 5, 13, 8, 1].

Sensor nodes may be deployed in hostile environments and thus, are susceptible to capture by an adversary. Node capture must be detected and the captured node must be revoked. This involves prompt termination of all forms of communication with a captured node. There are two forms of revocation namely - centralized and distributed revocation.

In the case of centralized revocation, a controller node or base station broadcasts a single revocation message containing a signed list of nodes to

revoked. Upon receiving the message, the other nodes in the DSN revoke the appropriate nodes.

Centralized revocation has the following advantages: (1) Non- circumventable (2) No need for node-to-node message authenticity (3) Revocation policy is uniformly enforced.

Centralized revocation has the following disadvantages: (1) Single point of failure (2) Slower than distributed revocation (3) Needs global revocation message

Although the single point of failure can be eliminated by revoker replication, the last two disadvantages cannot be easily removed. This requires revoking nodes in a distributed fashion. An example of a distributed revocation scheme was proposed by Chan, Perrig, and Song [12]. In their scheme, nodes that have a shared key with the node to be revoked, vote on the decision to revoke. Upon registering a threshold number of votes, the voting nodes revoke the concerned node. We illustrate the complexity of distributed revocation using this scheme.

1.1 Organization

This thesis is organized in five chapters. In the first chapter, we define what Mobile ad-hoc networks and DSNs are, and present some inherent problems

in DSNs that make it impossible to implement traditional public key cryptography solutions. The second chapter introduces revocation in DSNs and presents prior work on centralized and distributed revocation. In the third chapter our approach to distributed revocation in DSNs is explained. We then introduce a new revocation scheme based on threshold cryptography. In the fourth chapter, we evaluate the correctness of this new scheme in the context of its resilience to the identified problems with prior distributed revocation schemes. We define the notion of node degree tracking and explain why degree tracking could be a limiting factor in achieving distributed revocation. The final chapter concludes this work and presents possible future work.

1.2 Mobile Ad-Hoc Network (MANET)

Ad-hoc networks are a new wireless networking paradigm for mobile hosts. Unlike traditional mobile wireless networks, ad-hoc networks do not rely on any fixed infrastructure. Instead, hosts rely on each other to keep the network connected. Military tactical and other security-sensitive operations are still the main applications of ad-hoc networks, although the unique properties of ad hoc networks pave way for its commercial usage [26].

Ad-hoc networking refers to the spontaneous formation of a network of

nodes without the help of any infrastructure, usually through wireless communication channels [15]. In MANETs, in addition the nodes are also mobile. The infrastructure in a MANET is thus highly dynamic not only because of mobile nodes, but also because of the lack of guaranteed node connectivity. This lack of guaranteed connectivity is caused by the limited-range, potentially unreliable wireless communication that is characteristic of MANETs.

1.3 Distributed Sensor Networks (DSN)

DSNs are a special type of MANETs, with nodes that have limited computational and communication capabilities. These networks are dynamic in the sense that they allow the addition and deletion of nodes after deployment. Nodes are typically added to increase network size or to replace failing and unreliable nodes. In a DSN, nodes can communicate only with other nodes within wireless communication range. These nodes are said to form a neighborhood. In the Figure 1.1 we show an example a neighborhood.

DSNs are typically used for data acquisition in tasks such as health monitoring, military operation, etc. Sometimes confidentiality of communications between sensor nodes may be desired; e.g., when DSNs are used in military applications to collect information about tank positions or troop movements. Distributed sensor networks are, in such cases, a mission critical component

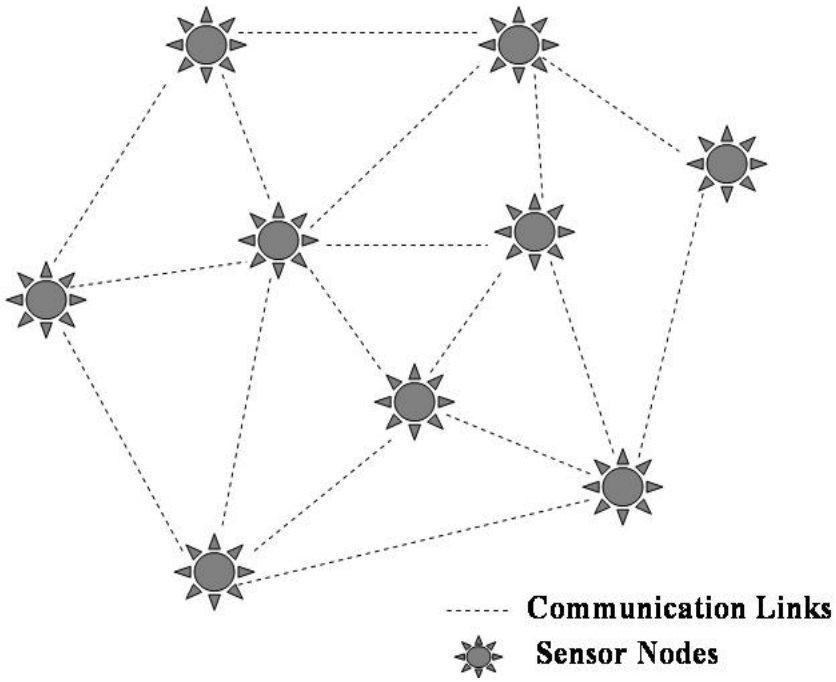

Figure 1.1: A neighborhood in a DSN

requiring commensurate communications security protection. Military equipment and personnel must be assured that received sensor information is correct. Deployed sensors must only accept legitimate queries, commands, and software updates. Sensor network communications must prevent disclosure and undetected modification of exchanged messages.

Confidentiality, integrity, and authentication services are critical to preventing an adversary from compromising the security of a distributed sensor network. Cryptography is an essential mechanism for protecting sensor communication and key management is likewise critical to establishing the keys

necessary to provide this protection [16]. However, providing key management can be a complex task due to the ad-hoc nature, intermittent connectivity, and resource limitations of the sensor network environment [9]. If a sensor network is deployed via random scattering (e.g. from an aerial vehicle), the sensor network protocols cannot know beforehand which nodes will be within communication range of each other after deployment. Even if the nodes are deployed by hand, the large number of nodes involved makes it costly to pre-determine the location of every individual node. Hence, a security protocol should not assume prior knowledge of which nodes will be neighbors in a network [12].

A number of key distribution and management schemes have been proposed [16, 12, 25, 6, 7, 22] to satisfy both operational and security requirements of sensor networks. If the sensor nodes are deployed in hostile regions, they are susceptible to be captured by an adversary. These captured nodes can be used by adversary for a number of malicious reasons like false data injection, launching DoS (denial of service) attacks to deplete network resources etc. Thus, upon capture detection, these compromised nodes have to be revoked.

Typically revocation involves termination of communication with the captured node. The limited computation and power resources of sensor nodes often makes it undesirable to use public-key algorithms, such as Diffie-Hellman

key agreement [24] or RSA signatures [20]. Currently, a sensor node may require in the order of tens of seconds up to minutes to perform these operations [9]. This exposes a vulnerability to denial of service (DoS) attacks by sensor battery exhaustion [12]. Such resource consuming attacks are especially significant in DSNs [9]. Hence, in the absence of conventional public key cryptography solutions, revocation in DSNs is a particularly challenging problem.

Chapter 2

Revocation in DSNs

In this chapter we present and analyze prior work on revocation in sensor networks. We specifically examine two schemes that propose methods for performing centralized [16] and distributed revocation [12]. Finally, we perform a detailed analysis of the distributed revocation scheme proposed in [12], and present inherent problems associated with any distributed revocation scheme for DSNs in general.

2.1 An Approach to Centralized Revocation

In this section we outline the scheme proposed Eschenauer and Gligor in [16]. In DSNs due to the absence of a centralized KDC (Key Distribution Center) [3], limited resources and computational capabilities, keys are generally predistributed. On deployment, the sensor nodes discover their neighbors; i.e., nodes that share a key with them, and establish secure communication links.

2.1.1 The Probabilistic Key Distribution Scheme

The probabilistic key distribution scheme relies on probabilistic key sharing between nodes of a random graph and uses a simple shared-key discovery protocol for key distribution, revocation and node rekeying. Prior to the deployment, a ring of keys are distributed to each sensor node. Each key ring consists of k keys randomly drawn without replacement from a random pool of P keys, which is generated offline. Along with the keys, key identifiers are also loaded into the sensors.

In the shared key discovery phase, every node discovers its neighbors in wireless communication range with which it shares keys. Each node broadcasts the key identifiers in its key ring unencrypted. Nodes that share a key can then perform a cryptographic handshake.

2.1.2 Revocation

Whenever a sensor node is captured, it is essential to be able to revoke the entire key ring of the node. For this, there is a controller node or a centralized node(which has a large communication range and may be mobile) broadcasts a single revocation message containing a signed list of k key identifiers to be revoked. To sign the list of key identifiers, the controller node generates a signature key, K_e and unicasts it to each node with K_{ci}, where K_{ci} is the

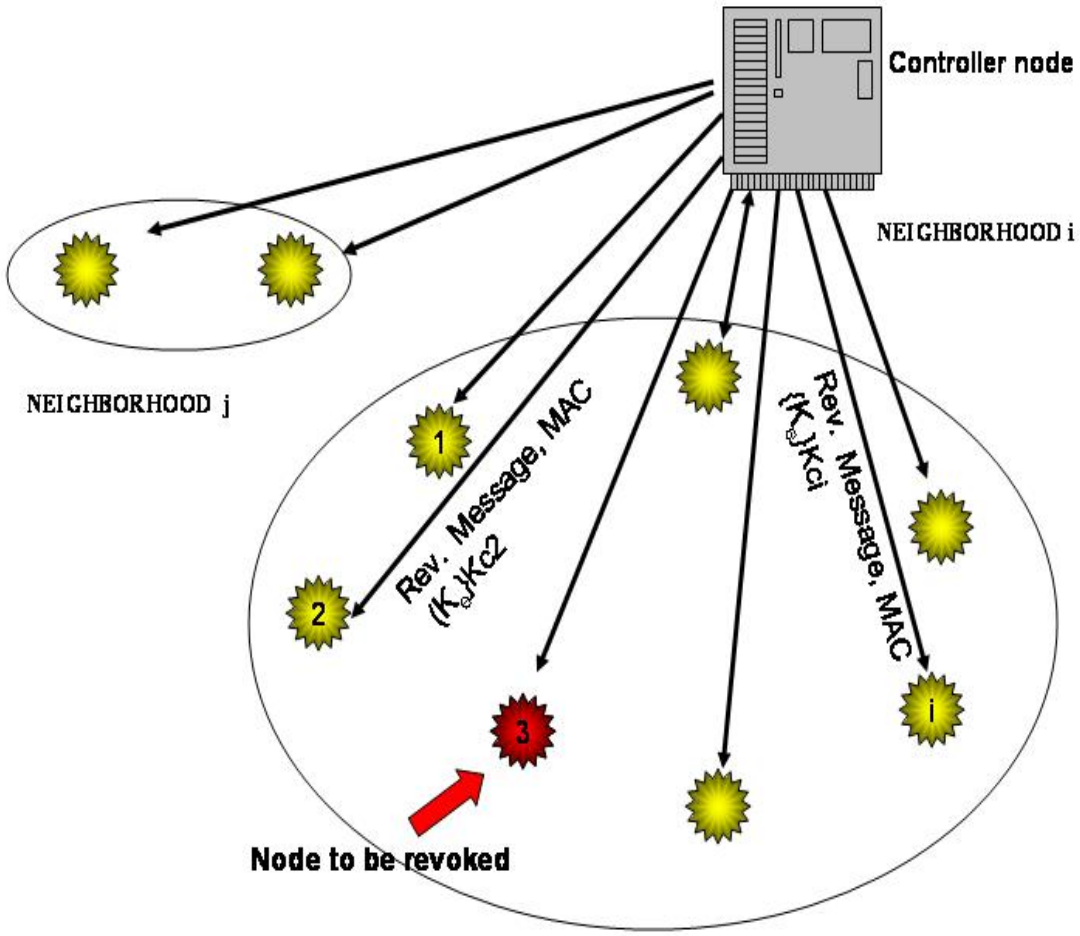

Figure 2.1: Controller node sends out Revocation message (Key Ids in Key Ring of Node 3) MAC-ed with K_e. MAC key K_e is send encrypted in K_{ci}.

key shared by the i^{th} controller with each sensor node during pre-distribution phase. For example, the controller node could sign the list of key identifiers by computing a MAC (Message Authentication Code) [21] with K_e and send K_e encrypted in K_{ci}. We illustrate this example in Figure 2.1 . After obtaining the signature key, each node verifies the signed list of key identifiers and deletes the appropriate keys from the key ring. Once the keys are removed

from the key ring, links with the compromised node disappear.

2.1.3 Summary

The scheme proposed by Eschenauer and Gligor in [16] is an example of centralized revocation. Centralized schemes have the following advantages:

- *Non-Circumventable:* As the revocation message is unicasted by a reliable controller node, and the message can be verified by all the sensor nodes in the DSN, the revocation of a compromised node is non-circumventable.

- *No need for node to node authenticity:* As revocation message is unicasted with a signature key distributed securely to all nodes by a centralized controller node, there is no need for node to node authenticity.

- *Revocation policy is uniformly enforced:* As there is controller node that acts as a centralized point of decision making revocation policy is uniformly enforced.

Centralized Revocation has the following disadvantages:

- *Single point of failure:* Due to the inherent nature of centralized schemes, compromise of a single controller node can completely defeat revocation of numerous compromised nodes in a network.

- *Slower than Distributed Revocation:* Due to the large number of unicasted messages, centralized revocation is slower than distributed revocation schemes.

- *Needs Global Revocation Message:* The design of the scheme enforces the necessity of a global revocation message that may add to the communication overhead of the network.

2.2 Distributed Revocation

In the key pre-distribution scheme presented thus far, while each node could verify that some of its neighbors had certain secret keys and thus were legitimate nodes, no node could authenticate the identity of the neighbor it was communicating with. The random-pairwise key scheme proposed by Chan, Perrig and Song [12] provides the property of node to node authentication for a network that consists of nodes that do not trust each other. Note that this property is useful for supporting numerous security functions. For example, if a node detects that one of its neighbors is captured, it may be necessary for the node to know the identity of the compromised neighbor. Also, with node to node authentication, nodes can keep track of the identities of their neighbors and can thus prevent an adversary from replicating in a network.

2.2.1 The Random-Pairwise Key Distribution Scheme

In a network with n nodes, a simple solution to key pre-distribution problem is that each node stores $n-1$ pairwise keys. This however leads to an undesirable increase in memory overhead.

The random-pairwise scheme is a modification of the above pairwise scheme based on the observation that not all $n-1$ keys need to be stored in a node to have a connected random graph with high probability. If p is the smallest probability that two nodes are connected, such that the entire graph is connected with a probability c , then to achieve this probability p in a network of n nodes each node needs to store only np keys instead of exhaustively storing all *n-1* keys. Reversing the calculation, if each node in the sensor network can store m keys then, the largest possible connected network is of size $\frac{m}{p}$

In the *pre-deployment initialization phase* a total of $n = \frac{m}{p}$ unique node identities are generated. The actual size of the network may be smaller than n. Unused node identities are used if additional nodes are added to the network in the future. Each node identity is matched up with m other randomly selected distinct node IDs and a pairwise key is generated for each pair of nodes. The key is stored in both nodes' key rings, along with the ID of the other node that also knows the key.

In the *post-deployment key-setup phase*, each node first broadcasts its node ID to its immediate neighbors (i.e., nodes within wireless communication range). By searching for each others' IDs in their key rings, the neighboring nodes can tell if they share a common pairwise key for communication. A cryptographic handshake is then performed between neighbor nodes who wish to mutually verify that they do indeed have knowledge of the key.

2.2.2 Revocation

To reduce the disadvantages associated with a controller dependent revocation protocol, Chan, Perrig, Song [12] describe a distributed node revocation scheme for the random pairwise scheme.

An outline of their scheme is as follows: Consider a node B, which, like every other node in the network, has m keys in its key ring. Since all the keys are issued to exactly two nodes and no two keys are issued to the same pair of nodes, we have exactly m nodes that share a pairwise key with node B. This set of m nodes are called the set of *voting members* of B. Each of these m voting members are assigned a random voting key k_i. Each voting member also knows the respective hashes of the voting keys of all the *m-1* other voting members; i.e., all $hash(k_j), j \neq i, 1 \leq j \leq m$. Hence, if m pairwise keys are stored on the node, the node stores $m - 1$ hash values of the corresponding

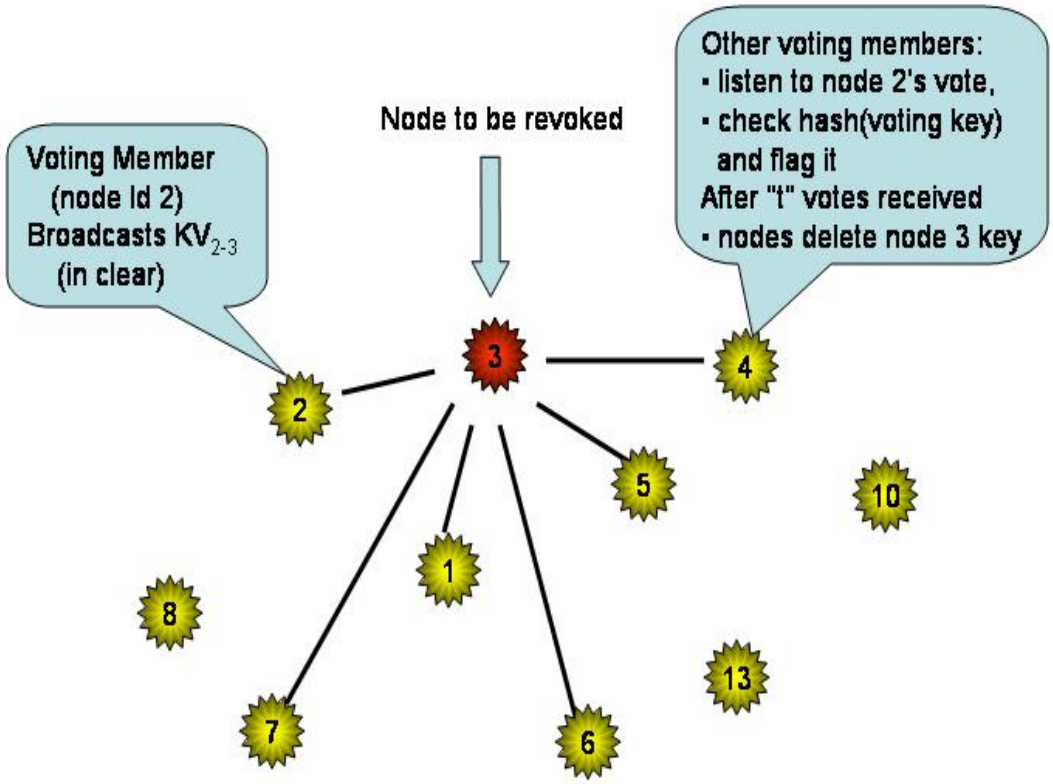

Figure 2.2: Voting scenario to effect revocation of node 3

voting keys, for each of these m neighbors. To cast a public vote against B, the *voting member* broadcasts k_i in clear(unencrypted). All voting members can verify the vote by computing $hash(k_i)$. If threshold t number of valid votes have been heard, a voting member deletes the shared key (with the node B) and thus revokes node B. Figure 2.2 illustrates the voting process for revocation. As a consequence, if a node cannot form at least $k*t$ connections

(where k is some small multiple, e.g. 2); i.e., if a node has less than $k*t$ neighbors at key setup phase then, that node is automatically revoked (for example; the node self destructs).

To prevent widespread release of revocation keys by compromised nodes, the scheme requires that only nodes that have established direct communication with some node B have the ability to revoke B. This is done by distributing the revocation keys to the voting members of B in a deactivated form, i.e. each voting member j stores its revocation key for B, k_{Bj} masked ($XORed$) with some secret S_{Bj}. This deactivated key will not hash to the correct verifying value and is thus useless for voting. Node B knows all the activation secrets $S_{Bj}; 1 \leq i \leq m$. During the key discovery and setup phase, if node j wishes to complete key setup with node B, it requires node B to transmit its activation secret S_{Bj} (and viceversa). Once node j has received S_{Bj} it unmasks k_{Bj} using S_{Bj}, and verifies that it was given the correct unmasking secret by performing vote verification on the unmasked k_{Bj} to see if it is a valid revocation key. Such a policy of need-to-know based, key activation ensures that the majority of revocation keys recovered through node capture are in an unusable masked state. In order to use these revocation keys to revoke some node B the adversary now has to physically communicate with B and complete key-setup for up to t new connections.

Degree Counting Mechanism

To limit the amount node replication possible on the network, the degree of any node can be limited [12]. This implies that we can limit the degree of nodes to d_{max}, without disrupting network connectivity. The operation of the degree counting scheme is exactly identical to the voting mechanism. Each node contains a voting key and some way to verify valid voting keys. Each time a given node A forms a connection with some node B, A broadcasts its voting key for B and vice-versa. Each node can thus track the degree of all m of the nodes which share pairwise keys with it, and refuse to form new connections if the degree becomes too large.

2.3 Problems of Distributed Revocation Schemes based on Voting

In this section we illustrate a few problems that are applicable to all distributed revocation schemes and not just specifically to the scheme described earlier.

1. Revocation Vote Authenticity

2. Node replication falls outside the scope of degree counting

3. Irrevocable nodes

4. Unspecified Revocation Policies

    - Unspecified start and end of revocation sessions
    - Unspecified Revocation Outcome
    - Unspecified Revocation Quotas

2.3.1 Revocation Vote Authenticity:

At the time of forming connections each node transmits the voting key (in clear) of the node with which it forms a connection. For example if node #2 and node #3 were to form connections, node #2 would transmit the voting key that could be eventually used for revoking node #3. Node #3 does the same in the context of node #2. The prospective neighbors of node #2 and node #3 (and any node in the neighborhood; i.e., node #5, node #7 and node #8)can thus hear this vote in clear.

Suppose as shown in Figure 2.3, a node #8 that is not a prospective neighbor of node #4, makes a copy of this voting key KV_{3-2} (node #3's voting key for revocation of node #2) at the time of connection establishment between node #2 and node #3. Now node #8 can broadcast a vote against node #3. The other voting members have a no way of distinguishing a legitimate vote

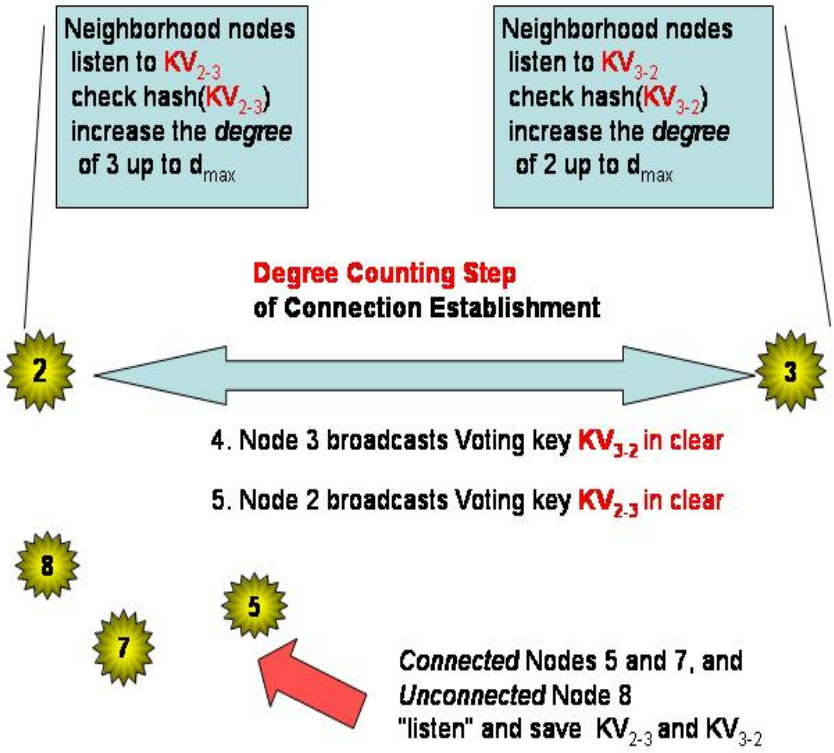

Figure 2.3: Lack of revocation vote authenticity: Any node in the neighborhood can hear another node's voting key.

(i.e., from a voting member) from that of vote forged by any node in the neighborhood, such as node #8. We illustrate this in Figure 2.4.

Hence an adversary can collect t votes against any target node of choice at the time of connection establishment and replay them later to revoke the target node without the participation of t valid voting members.

Problem Summary: A node receiving a vote cannot ascertain a vote's source with certainty.

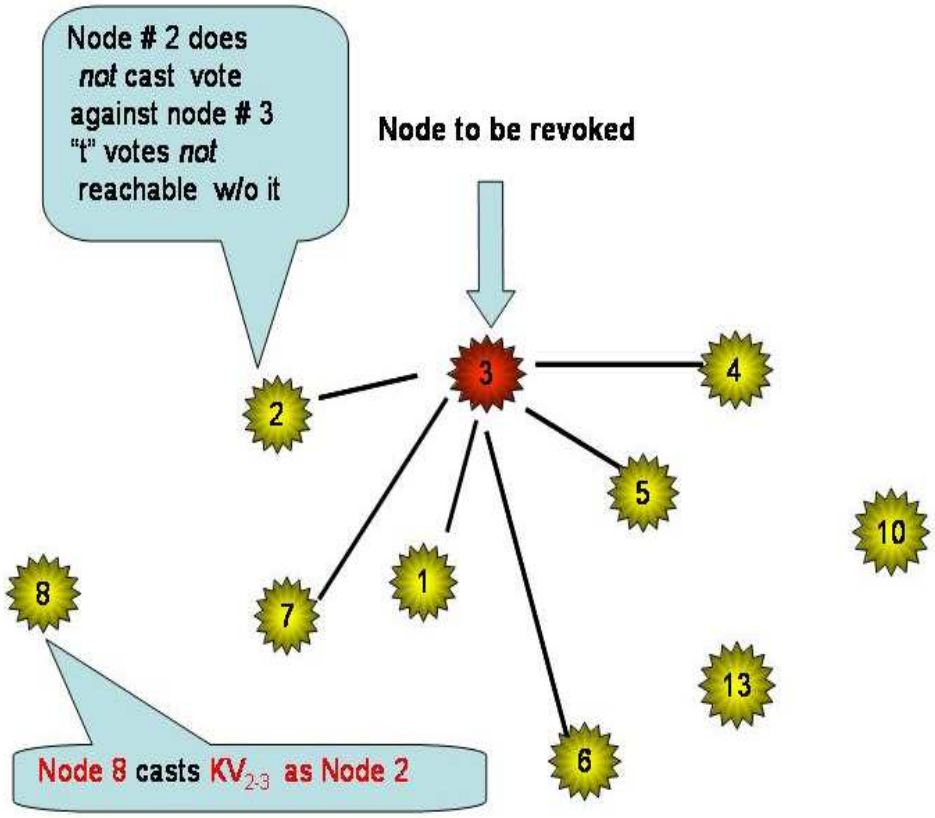

Figure 2.4: Lack of revocation vote authenticity: Any node in the neighborhood can cast a legitimate vote pretending to be some other node.

2.3.2 Node Replication Falls Outside the Scope of Node-Degree Counting in Neighborhoods

Nodes in a neighborhood can keep track of another node's degree (i.e., perform neighborhood degree counting) by counting the voting keys of a prospective neighbor at the time of connection establishment. If the degree of any node requesting a connection is equal to an upper limit d_{max}, the other nodes refuse to form connections with the node whose degree has reached the max-

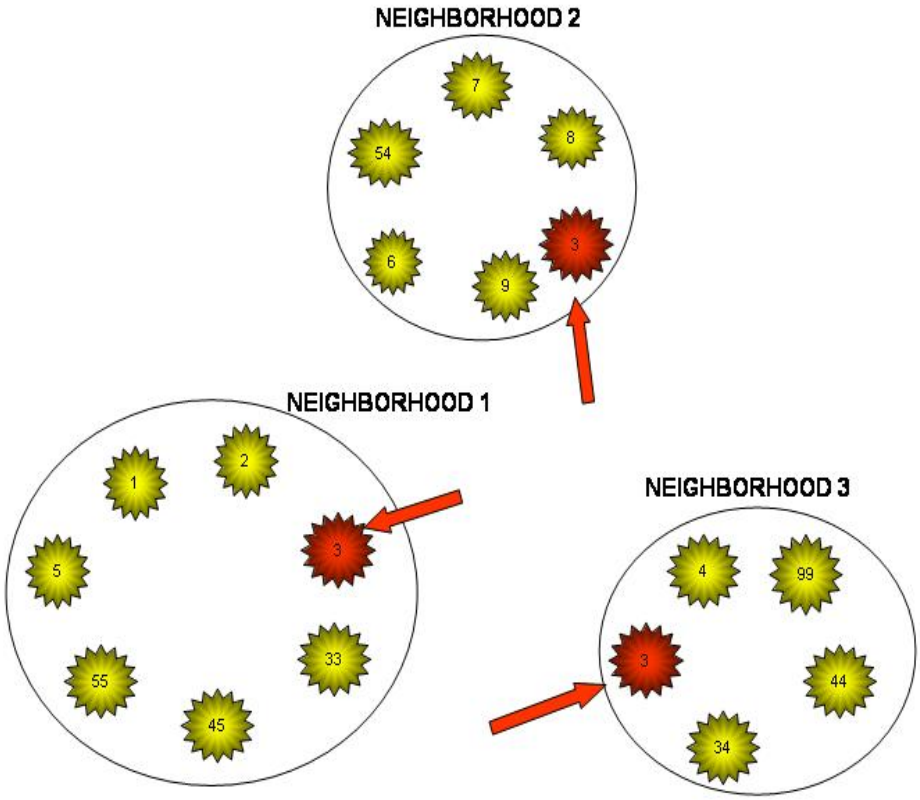

Figure 2.5: Adversary captures node 3 in neighborhood 1, copies its key ring, introduces new nodes with copy of node 3 key ring in the other neighborhoods. With a single node capture, an adversary circumvents the intent of degree counting mechanism

imum allowed value (d_{max}). This prevents nodes from replicating in a neighborhood. However, an adversary can still launch a replication attack despite neighborhood degree counting. Suppose an adversary captures a node # i, he copies the key-ring of node# i and introduces new sensor nodes with the key ring of node# i in other neighborhoods. Nodes in other neighborhoods cannot perform global degree tracking as most communications can be heard only within the neighborhood of a node. Hence, counting node degrees within the

neighborhood of a node is insufficient to detect node replication. Figure 2.5 shows the above scenario.

Problem Summary: With a single node capture an adversary can replicate in multiple neighborhoods without detection.

2.3.3 Irrevocable Nodes

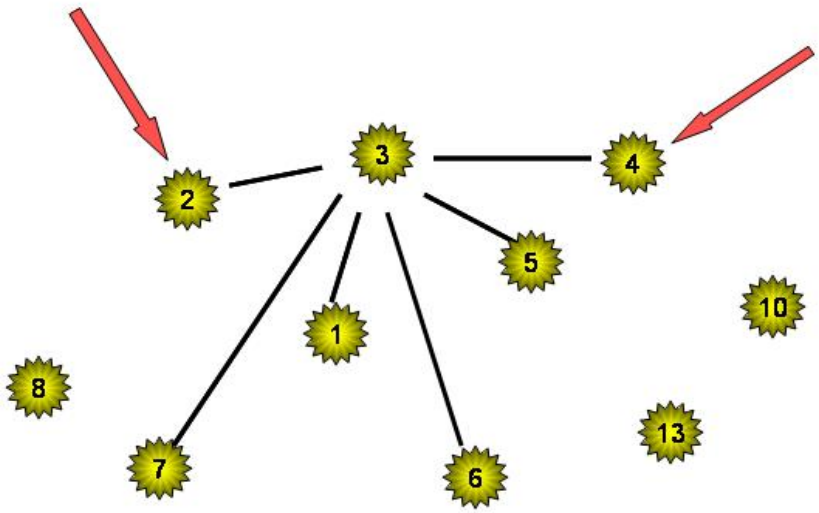

Figure 2.6: Node 4 colludes with node 2(degree of node 2 $=t$)to revoke node 3 => after revocation, node 2 becomes irrevocable ($< t$ neighbors)

To revoke a target node (using voting), at least t votes have to be heard

against the target node. Suppose, nodes #4 and #2 are voting members of a node #3. The degree of node #2 is t. Now nodes #4 and#2 can collude to revoke node#3. This would make node #2 irrevocable as the degree of node #2 is now *t-1* (after revoking node#3) and at least t votes are needed to revoke a sensor node.

Problem Summary: In any threshold based voting scheme there needs to be a way to prevent nodes in the network from falling below threshold.

2.3.4 Unspecified Revocation Policies

Unspecified start and end of revocation sessions

- Nodes participating in a revocation session do not know when to stop counting votes. This may imply an indefinite wait time for a vote.
- If multiple voting sessions are introduced, voting members have no authentic way of determining whether a voting session has been started off by a legitimate voting member or by an adversary who simply wants to collect voting keys to replay later.

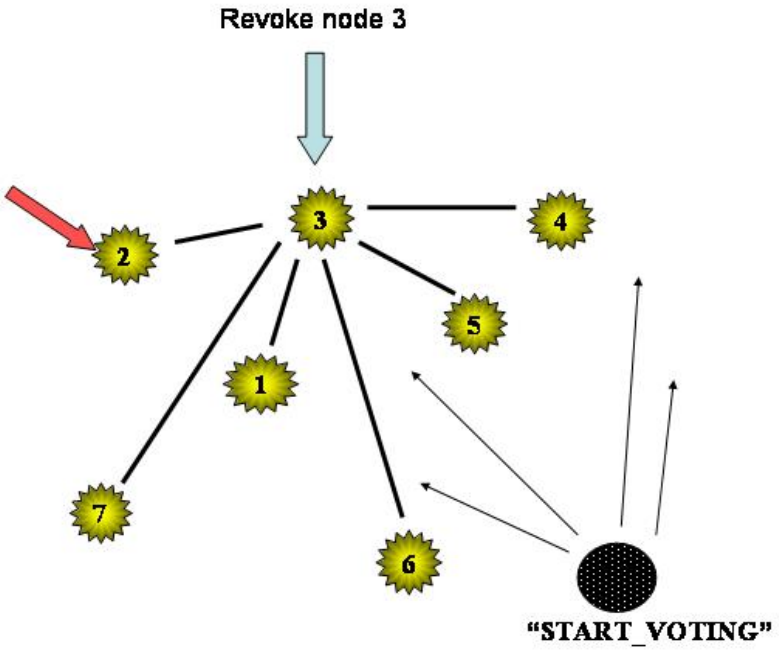

Figure 2.7: Unspecified start and end of revocation sessions: (1) Node 2 does not know when to stop counting votes. This leads to an infinite waiting time. (2) As nodes cannot verify the authenticity of a 'Start Voting' message.An adversary could start off a voting session to collect voting keys to replay later.

Unspecified Revocation Outcome

Suppose a revocation session against a target node fails, meaning that only g votes have been heard and $g < t$. In the future, t voting members may be prepared to vote against the same target node. Hence there needs to be in place a provision for multiple voting sessions to enable multiple voting attempts against a target node. This may not be possible with voting keys

that are already made public in the first voting session.

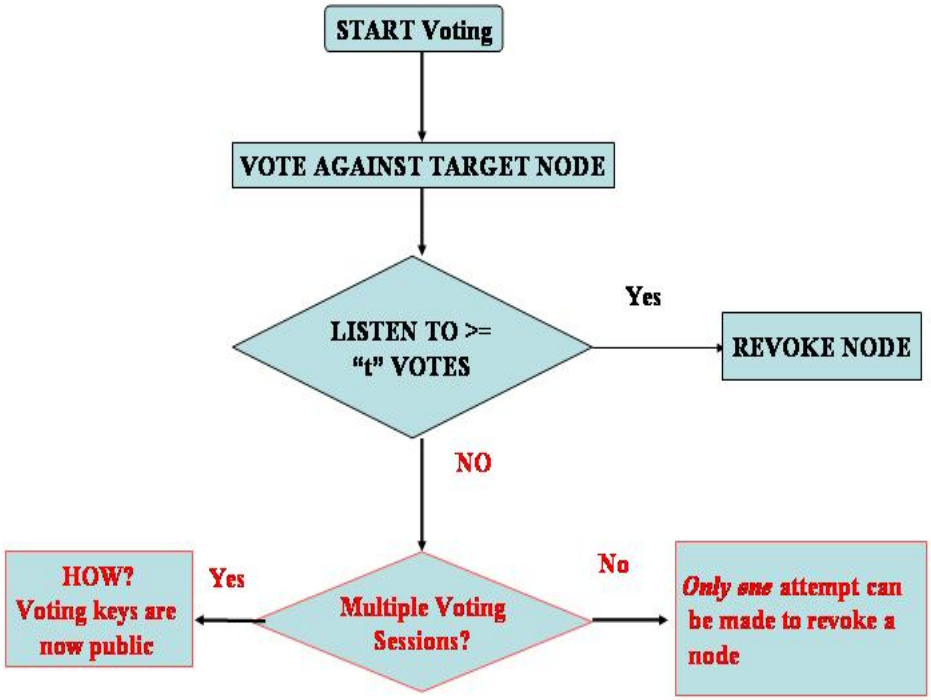

Figure 2.8: Flowchart illustrating the lack of a well specified revocation outcome

Unspecified Revocation Quotas

There has to be a ceiling limit on the number of nodes a particular node can revoke. This is required as a node# i could attempt to revoke its neighbors to such an extant that in the future the membership of the node# i falls below the threshold t meaning that the node# i is now irrevocable. If a node

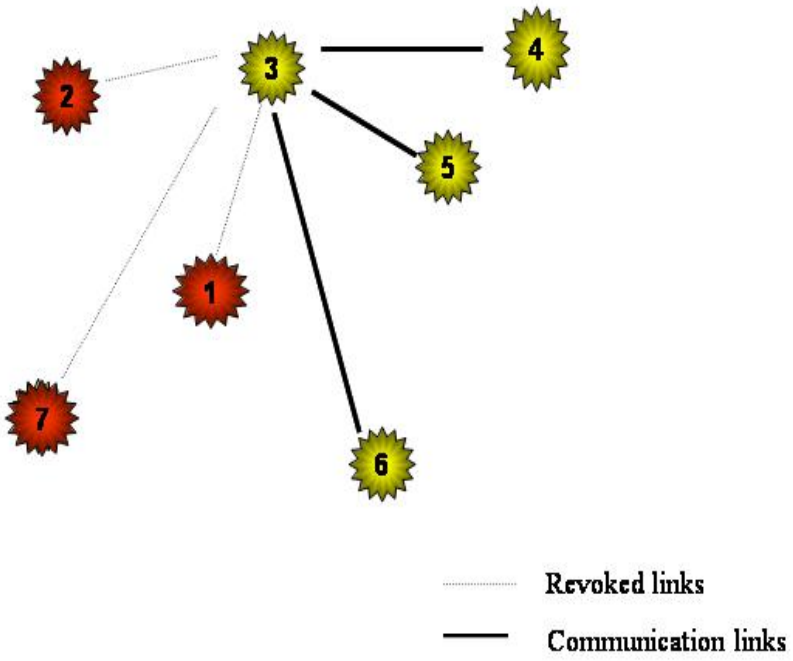

Figure 2.9: Example scenario: Suppose the threshold is $t = 4$. Suppose node 3 helps/manages to get node 2 revoked and over a period of time gets nodes 7,1 revoked too. As degree of node 3 ¡ t, node 3 is now irrevocable.

votes against a target node in its neighborhood and the target node is subsequently revoked, then the degree of the voting node decreases. Thus over a period of time the voting nodes could become irrevocable as their degree could fall below t. Hence, in any revocation session, it is in the best interest of participating nodes to revoke the target node even when the target node is harmless. This is an unintended side effect of threshold-based revocation.

Problem Summary: Revocation policy decisions need to specified. For example if multiple revocation sessions are supported, they must have a time bound. There also must be provisions for multiple voting sessions, and control over the number of revocations a node can attempt.

Chapter 3

A Distributed Revocation Scheme

In this section we propose a distributed revocation scheme for DSNs based on threshold cryptography. Threshold cryptography is generally used to divide data D into n pieces in such a way that D is easily reconstructable from any k pieces ($k \leq n$), but even complete knowledge of $k-1$ pieces reveals absolutely no information about D [2].

3.1 A Distributed Revocation Scheme

Our scheme relies on three cryptographic primitives: (1) hash functions, (2) random polynomials and (3) authenticated encryption.

3.1.1 Cryptographic Primitives Used

1.*Hash functions*

We use the one-way and collision-resistance properties of hash functions.

2.*Random Polynomials and their Threshold Keys*

We use a random number generator to produce coefficients for polynomials of degree t. Polynomial $q(x) = a_0 + a_1x + a_2x^2 + ... + a_{t-1}x^{t-1}$ is random if all its coefficients $a_0, a_1...., a_{t-1}$ are random uniformly distributed values in a certain range $[0, l-1]$ (e.g., $l = 2^{64}$). We define the threshold key of a random polynomial $q(x)$ to be $K_t = hash(a_0|a_1|....|a_{t-1})$, where hash is a hash function and $a_0, a_1, ..., a_{t-1}$ are the coefficients of $q(x)$. The properties of the hash function guarantee that: (1) if two random polynomials of the same degree are distinct (i.e., they differ in at least a coefficient), the probability that their threshold keys are identical is negligible; and (2) given a threshold key of a random polynomial, it is computationally unfeasible to find the coefficients of that polynomial. As their name implies, threshold keys have a threshold property; i.e., at least t distinct pairs $(q(x_i), x_i)$ of a random polynomial are required to compute its threshold key, and any set of t such pairs will yield the same threshold key. [Note: In our scheme, random polynomials are computed off-line. If a large number of random polynomials $j, 1 << j < 2^{l/2}$, of degree t are needed, they can be constructed efficiently

from a single random polynomial by setting $a_{ij} = f(a_i^{j-1})$, where $a_{i0} = a_i$ and f is an encryption function $E_k(0)$ with $k = a_i^{j-1}$.]

3. *Authenticated Encryption*

We use authenticated-encryption modes to detect (1) existential forgeries, and (2) false (or inauthentic) decryption keys (i.e., decryption keys not used by the corresponding encryption operation), during ciphertext decryption.

Existential Forgeries: Informally, authenticated encryption modes detect forged ciphertext (i.e., ciphertext not produced by encryption with a given key) at decryption with very high probability (i.e., probability negligibly close to one) and return an error.

Key Authenticity: Informally, given a ciphertext produced by an encryption mode and a random key, a key-authenticity test decides whether that ciphertext was produced with that key. Authenticated-encryption modes detect false (or inauthentic) encryption keys with very high probability at decryption and return an error. In this scheme, we test the authenticity of encrypted votes and of the session key with which votes are encrypted, using efficient AE modes. Such modes are widely known [23, 4, 18].

In general, we say that an encryption mode admits a key authenticity test, if there is a probabilistic polynomial time algorithm that that can distinguish between a ciphertext and the key it was encrypted with from a ciphertext and a random key. If the distinction is made with non-negligible (negligibly-

close-to-one) probability, the test is called the weak (strong) key authenticity, or WKA(SKA), test [17]. Another key-authenticity test is provided by the confusion freedom (CF) property of an encryption mode [10]. That is, if k and k' are random keys, then for any plaintext string x in the 3 domain of the encryption function Ek, Pr[Dk'(Ek(x)) =/= Null] is negligible, where Dk' is the decryption function using key k'. Or, decryption of a ciphertext with a key that differs from the random key used to generate that ciphertext will fail with probability negligibly close to one. Further, if AE is an authenticated encryption mode that is secure in an existential unforgeability sense [14], implications $AE => CF => SKA => WKA$ hold [17]. Hence, the use of AE modes is sufficient for key authenticity tests.

Notation:

For clarity, we list the symbols used in the section below:

n size of network,

m number of prospective neighbors of a node

t (threshold) number of votes required in a voting session to revoke a node

s maximum number of revocation sessions allowed against a node

SK session Key for encrypting revocation votes against a target in one of s sessions

K_t threshold key of a random polynomial of degree t (used to generate SK)

Mask session Mark received from the target node for session activation

H hash(Mask) used to verify the value of session Mask received from target node

Δt maximum duration of a revocation session

3.1.2 Off-line Node Initialization

First, we compute s random polynomials of degree t for each of the n nodes in the network, where s is the number of revocation sessions (attempts) against any node in the network. (For example, if the size of the sensor network is n = 10,000 and the number of revocation sessions is $s = 6$, this would require 60,000 polynomials of degree t.)

Second, we (1) generate a session key (SK) for each revocation session against a target node and mask it by xor-ing each it with a separate per-session Mask; (2) load the result of the xor in each of the target's potential neighbor nodes and each Mask in the target node; and (3) load the values of $H = hash(Mask)$ to each potential neighbor node of the target node. (The hash value, H enables each neighbor to verify the validity of each session's Mask received from a target node whenever a revocation session is activated;

viz., discussion in section3.1.3). Each SK is generated by encrypting a random 64-bit number in the threshold key $K_t = hash(a_0|a_1|....|a_{t-1})$ associated with a session's random polynomial, using an AE mode. (Note that a separate SK corresponds to each of the s random polynomials of a node.) SK is recovered upon session activation by xor-ing the masked session key with the session Mask.

Third, for each of the $m >> t$ neighbors of a target node and each of the s revocation sessions against that target, we compute a vote that is represented as the tuple $(q(x_i), x_i)$, where $q(x_i)$ is the value of a session's polynomial at $x = x_i$ for that session. Each vote is associated with the other revocation-session parameters (e.g., SK and hash(Mask)) and loaded into the corresponding node. Further, each of the $m >> t$ neighbors of a revocation target receives a value Δt, which represents the duration of a revocation session. (The value of Δt value is arrived at using network characteristics and computational power of sensor nodes.) In summary, the quantities are computed off-line and pre-loaded in nodes are:

1. Computed: s*n polynomials of degree t, keys K_t, keys SK and $hash(SK)$;

2. Pre-loaded in each node: s*m votes, $SK \oplus Mask$ values, hash(Mask) values, other Node's Masks, and a Δt value.

3.1.3 Activation of a Revocation Session

A session becomes active upon receipt of an authentic revocation vote by any of the local neighbors of a target node. An authentic vote is encrypted in the SK of that session. To ensure that only the actual (i.e., local, as opposed to all) neighbors of a target node can revoke that node, the $SK \oplus Mask$, instead of SKs are pre-loaded into neighbor nodes. Each local neighbor node obtains the session Mask from the target node, either as a result of connection establishment or later as a result of a direct request to the target node. A neighbor node receives a Mask encrypted in the pairwise shared key with the target node. The neighbor node decrypts the Mask and verifies its value by computing its hash and comparing the result against the stored value $H = hash(Mask)$ of that session, and then xors the Mask with $SK \oplus Mask$ to obtain the session key SK. Nodes exchange Masks after connection establishment for an initial revocation session against each other [CPS03]. If a revocation session against a target node fails, each neighbor requests the activation of a new session and corresponding Mask from the target node. Up to s sessions can be activated against a target node. If the target node refuses to activate the session and does not return the session Mask within a fixed interval of time, revocation of that node is automatic.

3.1.4 Revocation Commit and Target Revocation

The revocation decision is made only by the actual (local) neighbors of a target node, and requires that at least t neighbors' vote, and that a neighbor's vote cannot be forged/reused/replayed in the same or any other revocation session. To revoke any target node during a session, t votes $(q(x_i), x_i)$ encrypted in the SK for that session must be cast and heard by that target's neighbors. The coefficients $a_0, a_1 ... a_{t-1}$ are computed by every voting node from the first t votes heard. Once the coefficients are computed, the key K_t can be derived and the session's SK decrypted with K_t. Since SK decryption is performed using an authenticated encryption mode, if the SK decrypts correctly, then key K_t is correct (with probability negligibly close to one), which means that t participants must have voted and the votes are not replays (Revocation Commit). Then, each voting member deletes the key of the target node from its key ring (Target Revocation). Note that if fewer than t legitimate votes would be cast and some replayed to reach the threshold t, a key $K'_t = / = K_t$ would be produced and decryption of SK using an AE mode would fail with very high probability.

If t authentic votes haven't been heard within Δt, the revocation session against a target node would fail. This means the voting session has expired and all the participants delete the votes heard (for the failed session). A new

Mask from the target node is requested for the next revocation session.

3.1.5 Propagation and Global Verification of Local Revocation Commit

After a revocation decision is made in a neighborhood, each neighborhood node transmits the sessions K_t and Mask in clear to other nodes outside the neighborhood to reach all potential neighbors of a revoked node. This avoids replication attacks by adversary in new neighborhoods). A node receiving transmission of $Mask, K_t$ verifies the Mask by checking that $hash(Mask) = H$ and then unmasks $SK \oplus Mask$ to obtain the session key SK. If K_t decrypts SK correctly, the key corresponding to the revoked node is deleted and the message $Mask, K_t$ broadcast to other nodes.

The base station listens to the message $Mask, K_t$ broadcasted in any neighborhood in the network. The base station then deletes the keys shared by undeployed nodes with the target node. This way an adversary cannot have a revoked node establish connections with nodes that may be deployed in the future.

Chapter 4

Evaluation

In this section we define the few key notions and state the properties of our revocation scheme. We then proceed to argue the correctness of these properties using the implementation details of our revocation scheme, presented in chapter 3. This way, we evaluate the new scheme and its robustness in the context of the various problems relating to distributed revocation (discussed in chapter 2). We then list the problems that have not been addressed by the new scheme. That is, we discuss why tracking the degree of nodes in a DSN is a difficult problem and its role in solving the distributed revocation problem.

4.1 Definitions

Definition 1: (Target) A node to be revoked is called a target.

Definition 2: (Participant) A participant in the revocation protocol is any node that has a shared key with a target node.

Definition 3: (Vote Authenticity) Given a set of participants that vote, we say that a vote is authentic if the vote (a) is cast only by a participant, (b) is not forged, and (c) is not a replay of another vote.

Definition 4: (Revocation Commit) Given a threshold t and a set of m participants, $m >> t$, we say that revocation is committed when any set of t authentic votes is heard (recorded) by any of the participants.

Definition 5: (Target Revocation) A target is revoked when all m participants (a) verify revocation commitment, and (b) delete their keys shared with that target.

Definition 6: (Non-blocking Revocation) A revocation protocol is non-blocking if a non-participant, target or non-local participant node cannot block revocation.

Note 1: If a revocation policy is implemented, then we can provide a revocation outcome in bounded time; i.e., we bound a revocation session by a

start time and a stop time within which participants have to decide whether to commit or not. In the presence of revocation sessions, there is a provision for failure outcome in a revocation session. In a revocation session, the outcome could either be a success (target node revoked) or a failure (target node not revoked).

Definition 7: (Revocation Session) A revocation session is defined by the following parameters:

(a) a session VOTE-START time that marks the receipt of the first authentic vote by a participant;

(b) a session VOTE-STOP time that is defined by VOTE-START $+\Delta t$, where Δt is the maximum duration of a session;

(c) a set of votes that are unique to that session;

(d) an outcome (success/failure) for that session:

- a Failure outcome, which implies the condition: (Participant Clock = Stop-Vote and number of authentic votes received $< t$) or (Revocation Commit Message is not heard (received)).

- a Success outcome is target revocation

Note 2: In the presence of a revocation policy and provisions for failure outcome, it is implied that we allow session retries. This, in turn implies we

must have the notion of a session number (needed to identify a session) and a limit to the number of new revocation sessions allowed against the target node.

Note 3: The notion of local participants along with a revocation policy, means that the revocation outcome of a session is local.

Definition 8: (Locality of Revocation Sessions) The outcome of a revocation session against a target is conducted by $m' < m$ local participants and the outcome of that session is transmitted to all (non-local) participants.

4.2 Properties of Revocation :

Property 1: *If a target node is revoked, then at least t local participants voted to revoke and all t votes must have been authentic.*

Proof Sketch: By definitions 4 and 5, if a target is revoked, $m >> t$ participants verified revocation commitment and it means that t authentic votes have been heard (recorded) by any of the participants. However, by definition 3, vote authenticity requires that only local participants can cast votes, the votes can be neither forged nor replayed.

- *At least t local participants must vote to revoke:* There must exist at least t local participants; i.e., at the time of deployment, if a node has

fewer than t neighbors, it self-destructs. Then, by definition 5, target revocation implies revocation commit and by definition 4, at least t local participants must have voted to revoke.

- *Only local participants can cast votes:* Only the local participants of a node can recover a SK for a given session since the Masks received from the target node are encrypted in the pairwise shared keys. Hence only local participants can cast votes.

- *Votes cannot be forged:* Votes are encrypted with the SK for that session. This means that only votes cast by local participants decrypt correctly in an AE mode using SK. A local participant cannot forge another local participant's vote (i.e., the vote of a local participant who decided not to vote for revocation), since it would not be able to compute a valid vote other than its own. That is, a local participant could not compute the coefficients of a session's random polynomial from fewer than $t(q(xi), xi)$ values.

- Votes cannot be replayed:

  - a. Votes cannot be replayed within a voting session: If any of the t voted are duplicates, the threshold key K'_t obtained would correspond to a polynomial of a lower degree $t' < t$. Hence, $K'_t =$

$/ = K'_t$ and decryption of SK in an AE mode with K'_t would fail.

- b. Votes cannot be replayed across voting sessions: We use different SKs for different sessions. Hence, decryption of a $SK' = / = SK$ of the current session in an AE mode with K_t would fail.

Property 2: *Target revocation is a unitary (i.e., all or nothing) action.*

Proof Sketch: By definition 2, participants are nodes that have a shared key with the target node. Some of the participants may either be currently deployed in the sensor network or may be deployed by the base station in the future. By definition 4, revocation commit implies that any participant has heard t authentic votes. By definition 5, target revocation implies that all m (deployed and undeployed) participants have verified revocation commitment and deleted shared keys with the target node.

- If a node has been revoked in a neighborhood, then at least one of the local participants has used t authentic votes to arrive at K_t and has unmasked SK. This participant broadcasts the message $Mask, K_t$ in clear. All local participants delete their shared key with target node.

- Nodes that can hear the broadcast $Mask, K_t$, re-broadcast this message further. This message reaches other neighborhoods (other than the neighborhood in which the target node was revoked).

Target revocation by participants deployed in the network:

- Participants of the target node that may be present in other neighborhoods can verify Mask against the pre-stored $hash(Mask)$, unmask the SK (with Mask in message) and can try decrypting the SK with the key K_t in the transmission.

- If K_t decrypts the SK (i.e., the SK is valid), then participants in other neighborhoods delete the shared key with the target node. Now, the target node cannot form connections with any participant in the network.

Target revocation by undeployed participants:

- The base station can also listen to the broadcast $Mask, K_t$ from any neighborhood. The base station can verify Mask against the pre-stored hash(Mask), unmask the SK (with Mask in message) and can try decrypting the SK with the key K_t in the transmission.

- If K_t decrypts the SK (i.e., the SK is valid), then the base station deletes the keys shared by undeployed nodes with the revoked node. Now, the target node cannot form connections with any participant that may be deployed into the network in the future.

Property 3: *The target revocation protocol is non-blocking.*

Proof Sketch: Definition 8 implies that only a local participant can influence the revocation decision. Only the target, non-participant and non-local participant nodes block revocation against a target node and by design, there are intuitively only two ways these nodes could possibly block the protocol; i.e., within a revocation session by forcing local participants to compute the wrong Kt within revocation session or by exhausting the number of revocation sessions against a target node.

Claim 3.1:Target, non-participant, and non-local participant nodes cannot block target revocation by forcing local participants to compute wrong K_t within a revocation session.

- Only local participants can unmask SK from Mask and $SK \oplus Mask$. An authentic vote is encrypted in SK in the AE mode. Target node knows only the Mask and does not have the $SK \oplus Mask$. Hence, the target node cannot cast an authentic vote without the knowledge of SK.

- Non-participant nodes do not have $SK \oplus Mask$ (pre-distributed to all participants) and Mask (distributed only to local participants by the target node using shared keys) and thus, do not have SK. Hence,

non-participant nodes cannot cast an authentic vote without obtaining SK.

- Participants outside a neighborhood cannot obtain the SK by unmasking their $SK \oplus Mask$ without the Mask, which is available only within a local neighborhood (Mask distributed only to local participants by the target node using shared keys). Hence, non-local participant nodes cannot cast an authentic vote without unmasking SK.

Any target, non-participant and non-local participant node cannot cast authentic votes and hence, these nodes cannot cast votes that do not compute the correct K_t (for a given revocation session), and consequently, block the revocation of the target node.

Claim 3.2: Target, non-participant, and non-local participant nodes cannot block target revocation by exhausting sessions.

By *Claim 3.1*, any target, non-participant and non-local participant node cannot block the target revocation protocol by forcing local participants to compute wrong K_t. Thus, these nodes cannot exhaust the maximum number of sessions allowed against a target node, by forcing local participants to repeatedly compute the wrong K_t for each separate session.

Note: The target revocation is blocking if a malicious participant leaks the key SK to other nodes (both participants and non-participants). In that

case, any node can cast invalid votes that compute the incorrect k_t, and consequently block revocation of the target node. However, Property 3 does not claim to associate the non-blocking property with participants. The above attack is feasible only when a participant decides to block target revocation and thus, is beyond the scope of Property 3. However, this problem can be solved by loading the hash values of legitimate votes into the participants before deployment. This way, a participant can differentiate between a valid vote that computes the correct k_t and an invalid vote. Chan, Perrig and Song proposed a similar scheme [12] using Merkle hash trees[19] to address this problem.

Property 4: *A revocation decision is made in bounded time.*

Proof Sketch: By definition 7, revocation sessions are finite and also, failure outcome in a revocation session implies that there are multiple revocation sessions. Also, there are finite number of attempts to revoke a target node within which the participants must arrive at a revocation decision.

- If Revocation session fails then, each participant requests a new Mask from the target node to unmask a new SK for the next revocation session.

- If the number of revocation sessions is greater than session-retry Limit

then the number of attempts to revoke the target node have been exhausted and a new mask (to unmask a new SK) is not available to the participants for revoking the target node.

Property 5: *If revocation commit is verified by all participants, replication of a revoked node outside its neighborhood is not possible.*

Proof Sketch: By definition 8 and 5 any participant in the network (other than in an actual neighborhood) can verify a local revocation commit and can delete shared-keys with the revoked node.

- If a node has been revoked in a neighborhood, at least one of the local participants has arrived at K_t using t authentic votes and has unmasked SK. This participant broadcasts the message $Mask, K_t$ in clear.
- Nodes that can hear this message, re-broadcast this message further. This message reaches other neighborhoods (other than the neighborhood in which the target node was revoked).
- Non-local participants of the target node that may be present in other neighborhoods can verify Mask against the pre-stored hash(Mask), unmask the SK (with Mask in message) and can try decrypting the SK with the key K_t in the transmission.

- If K_t decrypts the SK (i.e., the SK is valid), then participants in other neighborhoods delete the shared key with the target node. Now, an adversary cannot introduce a revoked node (its key ring) in new neighborhoods.

4.3 Problems not Addressed by the Distributed Revocation Scheme

The following three problems are not addressed by the new scheme:

1. Node-degree tracking
2. Irrevocable nodes
3. Enforcing quotas for the number of allowed revocations

If a solution to problem 1 is found, then solutions to problems 2 and 3 are immediate. To solve problem 2 (irrevocable nodes) it is necessary to establish the degree of a node in an authentic manner, since a node would refuse to cast a vote against a target node if there is a voting member with degree = t in the neighborhood of the target node. Hence, solving problem 2 implies solving problem 1. This would consequently limit the number of revocations any node can effect; i.e., problem 3 (enforcing quotas) would be solved. Thus

solving problem 3 implies solving problem 2 implies solving problem 1. Degree tracking could either be done in a centralized or a distributed fashion. In either case, the verification of the degree of any node in the neighborhood would have to be done at any point of time, not just at deployment.

If the neighborhood would be fully connected, degree tracking could be implemented in a distributed manner. A fully connected neighborhood would mean any node could communicate in an authentic manner with any other node. Hence, there is a trustworthy connection between any two nodes in the neighborhood. This means there a need for an elaborate key distribution scheme would not arise, and revocation would only involve authenticated exchange of messages between nodes. However, establishing fully connected neighborhoods is a challenging problem in itself considering the memory overhead needed in storing that many keys/node to establish a fully connected neighborhood.

Another solution could be to use a centralized lookup of the degree of a node of interest. In that case, the centralized node that counts the degree would need to update the degrees of nodes periodically. This would mean that degree tracking might becomes a challenge in itself whenever revocation is distributed. This is the case because the centralized node in-charge of degree tracking might not be aware of which nodes would be revoked. If this centralized node would have to be informed of revocations in an authen-

tic manner, the line between centralized and distributed revocation would be blurred. Also, in such a scenario centralized revocation would become faster than distributed revocation as now, a centralized mechanism for degree lookup would have to be used in distributed revocation.

Chapter 5

Conclusion

5.1 Conclusions and future work:

The notion of revocation in sensor networks differs significantly from revoking entities in other common networks such as the internet. As public key cryptography solutions are not viable in DSNs and due to their mobile *ad hoc* nature, revocation of sensor nodes in DSNs is a particularly challenging problem. Revocation in sensor networks can either be centralized or distributed.

We discussed a centralized revocation scheme proposed by Eschenauer and Gligor [16], and identified problems with centralized revocation. We also discussed a distributed revocation scheme proposed by Chan, Perrig, and Song [12]. We illustrated some of the complexity of distributed revocation using this scheme and concluded that much of the complexity illustrated holds for all distributed revocation schemes, and not just for this scheme.

We then presented a distributed revocation scheme for DSNs using threshold cryptography. We finally evaluated the scheme by stating properties and arguing their correctness. In future work, we propose to extend the algorithm to address the problem of node degree tracking. As observed in chapter 4, solving the node degree tracking problem would automatically solve the other problems identified with distributed revocation schemes.

BIBLIOGRAPHY

[1] A.Perrig, R.Szewczyk, V.Wen, D.E.Culler, and J.D.Tygar. SPINS: security protocols for sensor netowrks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[2] A.Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[3] B.Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, 1994.

[4] C.Jutla. Encryption modes with almost free message integrity. *Lecture Notes in Computer Science*, 2045, 2001.

[5] D.Gay, P.Levis, R.Behren, M.Welsh, E.Brewer, and D.Culler. The nesc language: A holistic approach to networked embedded systems.

[6] D.Liu and P.Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003.

[7] D.Liu and P.Ning. Location-based pairwise key establishments for static sensor networks. In *ACM Workshop on Security in Ad Hoc and Sensor Networks*, 2003.

[8] D.Niculescu and B.Nath. Ad hoc positioning system (aps) using aoa.

[9] D.W.Carman, P.S.Kruus, and B.J.Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, September 2000.

[10] R.Housley D.Whiting and N.Ferguson. Counter with cbc-mac (ccm). Technical report, Submission to NIST, June 2000.

[11] F.Akyildiz, W.Su, Y.Sankarasubramaniam, and E.Cyirci. Wireless sensor networks: A survey. *Computer Networks*, 38:393–422, 2002.

[12] H.Chan, A.Perrig, and D.Song. Random key predistribution schemes for sensor networks. In *IEEE symposium on Research in Security and Privacy*, 2003.

[13] J.Hill, R.Szewczyk, A.Woo, S.Hollar, D.E.Culler, and K.Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[14] J.Katz and M.Yung. Unforgeable encryption and adaptively secure modes of operation. *Lecture Notes in Computer Science*, 2000.

[15] L.Eschenauer. On trust establishment in mobile ad-hoc networks. Master's thesis, Univeristy of Maryland, College Park, 2002.

[16] L.Eschenauer and V.Gligor. A key-mangement scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communication Security*, 2002.

[17] O.Horvitz and V.Gligor. Weak key authenticity and the computational completeness of formal encryption. *CRYPTO 2003*, pages 530–547, 2003.

[18] P.Rogaway, M.Bellare, J.Black, and T.Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security*, pages 196–205, 2001.

[19] R.Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 122–134, 1980.

[20] R.Rivest, A.Shamir, and L.Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1999.

[21] S.M.Matyas R.R.Jueneman and C.H.Meyer. Message authentication codes. *IEEE Communications Magazine*, 23(9):29–40, 1985.

[22] S.Zhu, S.Setia, and S.Jajodia. Location-based pairwise key establishments for static sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003.

[23] V.Gligor and P.Donescu. Fast encryption and authentication: Xcbc encryption and xecb authentication modes, 2000.

[24] W.Diffie and M.Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, (6):644–654, 1976.

[25] Y.S.Han W.Du, J.Deng and P.K.Varshney. A pairwise key predistribution scheme for wireless sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003.

[26] L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6):24–30, 1999.