2009

A privacy preserving framework for cyber-physical systems and its integration in real world applications

Haodong Wang
College of William & Mary - Arts & Sciences

A Privacy Preserving Framework for Cyber-Physical Systems and its Integration in Real World Applications

Haodong Wang

Shanghai, China

Master of Science, The Pennsylvania State University, 2000

Bachelor of Engineering, Tsinghua University, 1994

A Dissertation presented to the Graduate
of the College of William and Mary in Candidacy for the Degree of
Doctor of Philosophy

Department of Computer Science

The College of William and Mary
August, 2009

APPROVAL PAGE

This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Haodong Wang

Approved by the Committee, July 2009

Committee Chair
Dr. Qun Li
Computer Science Department
The College of William and Mary

Dr. Phil Kearns
Computer Science Department
The College of William and Mary

Dr. Weizhen Mao
Computer Science Department
The College of William and Mary

Dr. Haining Wang
Computer Science Department
The College of William and Mary

Dr. Gexin Yu
Mathematics Department
The College of William and Mary

ABSTRACT PAGE

A cyber-physical system (CPS) comprises of a network of processing and communication capable sensors and actuators that are pervasively embedded in the physical world. These intelligent computing elements achieve the tight combination and coordination between the logic processing and physical resources. It is envisioned that CPS will have great economic and societal impact, and alter the qualify of life like what Internet has done. This dissertation focuses on the privacy issues in current and future CPS applications. As thousands of the intelligent devices are deeply embedded in human societies, the system operations may potentially disclose the sensitive information if no privacy preserving mechanism is designed. This dissertation identifies data privacy and location privacy as the representatives to investigate the privacy problems in CPS. The data content privacy infringement occurs if the adversary can determine or partially determine the meaning of the transmitted data or the data stored in the storage. The location privacy, on the other hand, is the secrecy that a certain sensed object is associated to a specific location, the disclosure of which may endanger the sensed object. The location privacy may be compromised by the adversary through hop-by-hop traceback along the reverse direction of the message routing path. This dissertation proposes a public key based access control scheme to protect the data content privacy. Recent advances in efficient public key schemes, such as ECC, have already shown the feasibility to use public key schemes on low power devices including sensor motes. In this dissertation, an efficient public key security primitives, WM-ECC, has been implemented for TelosB and MICAz, the two major hardware platform in current sensor networks. WM-ECC achieves the best performance among the academic implementations. Based on WM-ECC, this dissertation has designed various security schemes, including pairwise key establishment, user access control and false data filtering mechanism, to protect the data content privacy. The experiments presented in this dissertation have shown that the proposed schemes are practical for real world applications. To protect the location privacy, this dissertation has considered two adversary models. For the first model in which an adversary has limited radio detection capability, the privacy-aware routing schemes are designed to slow down the adversary's traceback progress. Through theoretical analysis, this dissertation shows how to maximize the adversary's traceback time given a power consumption budget for message routing. Based on the theoretical results, this dissertation also proposes a simple and practical weighted random stride (WRS) routing scheme. The second model assumes a more powerful adversary that is able to monitor all radio communications in the network. This dissertation proposes a random schedule scheme in which each node transmits at a certain time slot in a period so that the adversary would not be able to profile the difference in communication patterns among all the nodes. Finally, this dissertation integrates the proposed privacy preserving framework into Snoogle, a sensor nodes based search engine for the physical world. Snoogle allows people to search for the physical objects in their vicinity. The previously proposed privacy preserving schemes are applied in the application to achieve the flexible and resilient privacy preserving capabilities. In addition to security and privacy, Snoogle also incorporates a number of energy saving and communication compression techniques that are carefully designed for systems composed of low-cost, low-power embedded devices. The evaluation study comprises of the real world experiments on a prototype Snoogle system and the scalability simulations.

Table of Contents

To my wife, Yawen, and my sons, Branden and Caden

ACKNOWLEDGMENTS

First, this dissertation would not have been possible without the expert guidance of my esteemed advisor, Dr. Qun Li. Not only was he readily available for advising me, but he always carefully read my paper drafts, sentence by sentence, and generously helped me to prepare for all sorts of oral presentations. His comments are always extremely thoughtful, inspiring and helpful. His constructive urgings pushed many of my projects to the frontier that I would never reach by myself. His dedicate and passion for the high quality research has always been the criteria I want to pursue.

My thanks go out to the members of my dissertation committee, Dr. Phil Kearns, Dr. Weizhen Mao, Dr. Haining Wang and Dr. Gexin Yu. They have generously given their time and expertise to better my work. I thank them for their contribution and their great support.

I must acknowledge as well the many colleagues and friends who supported my research and projects over the years. Especially, I am grateful to my colleagues Bo Sheng and Chiu C. Tan for many collaborative projects during my PhD study. I thank Fengyuan Xu for his great assistants in the road tests for the vehicular wireless network project. My thanks also go to Hao Han, Lei Xie, Yifan Zhang, and Wei Wei for many engaging discussions. My appreciation must go also to Mengjun Xie for many discussions in various research topics.

I would like to extend my thanks to all the staff in the Computer Science Department. In particular, I would like to appreciate Vanessa Godwin and Jacqulyn Johnson. Their kind assistance on numerous administration issues and reimbursement saved me great amount of time and enabled me to completely focus on my research.

I am especially grateful to my high school, Shanghai High School (SHS). I have the great fortune of benefiting from SHS's rigorous curriculum, elite teachers and gorgeous campus. Not only did SHS give me a solid foundation that allows me to pursue higher education, but it helped

me to have the correct outlook on the world and on life.

Finally, I would like to dedicate this dissertation to my wife, Yawen, and our two little boys, Branden and Caden. It is truly a bad idea to marry a PhD graduate student, who has to spend almost all the time sitting in the lab or the library. I am blessed to have Yawen who has unwavering faith in me, always stand my side, support and encourage me throughout the years. Branden and Caden also cheered me during the final years in my PhD journey – teasing me as I struggled with paper and thesis writing.

List of Tables

List of Figures

A Privacy Preserving Framework for Cyber-Physical Systems and its Integration in Real World Applications

Chapter 1

Introduction

A cyber-physical system (CPS) comprises of a network of processing and communication capable sensors and actuators that are pervasively embedded in the physical world. These intelligent computing elements achieve the tight combination and coordination between the logic processing and physical resources. CPS creates a new realm that computing can be closely interacted with the physical world where it occurs. It is expected [57] that this tight link between the logical processing elements and the physical world will dramatical improve the system performance in terms of adaptability, autonomy, efficiency, functionality, reliability, safety and usability. The real world examples of the CPS system include the Distributed Robotic Garden [23] and CarTel [47] project by researchers in MIT. The potential applications of CPS are enormous and can be used in aerospace, automotive, chemical processes, civil infrastructure, energy, health-care, manufacturing, transportation, entertainment, and consumer appliances. It is envisioned that CPS will have great economic and societal impact, and alter the qualify of life like what Internet has done.

Unlike the conventional resource-rich computing systems, each individual computing element in CPS is fairly weak for complicated tasks. Unlike the traditional embedded systems, on the

other hand, CPS works in a form of tightly coordinated device network rather than a stand-alone system. The new computing system concept of CPS does present the technical challenges across computer science. In this dissertation, we target on the privacy preserving problems in this new realm. It is our strong belief that security and privacy is an indispensable component as thousands of, even millions of, intelligent small devices become deeply integrated in human societies. For example, sensors today can be found on diverse objects such as buildings, trees, cars [47], and even clothing [35]. As these tiny devices become more ubiquitous in our environment, the privacy issues have become the main concern for the real world applications of CPS. To facilitate the study, this dissertation selects the wireless sensor networks as the platform to study the privacy issues because sensor networks can be considered as a pre-cursor generation of CPS, and sensor devices have already become relatively affordable commodities. From now on, CPS and sensor networks are used interchangeably in the rest of the dissertation.

1.1 Privacy Issues

Before going into the problem, let us first look at the definition of privacy. Privacy can be defined as *"the interest that individuals have in sustaining a 'personal space', free from interference by other people and organizations"* [20]. In the sensor network context, privacy can be easily understood as the data privacy. The data sensed by the sensor nodes become the private interest and should be protected. In addition to the data, there are other interests in sensor networks. For example, routing is one of the main operations. As we will discuss later in detail, the message routing paths are also very important information and may reveal some valuable system activities.

The privacy infringement occurs when the sensitive information of a certain private sector is disclosed to an untrusted or unauthorized party. Note that the sensor networks can be considered

as a collection of many tiny devices that sense, collect and transport the information (including the sensitive information) at their vicinity. The privacy problems emerge due to the deployment of sensor networks because the information, previously was isolated and secluded within the private sector, now can be more easily accessed through the sensor nodes. While sensor nodes provide the efficient and automatic data collection capabilities, they also give the chance to the adversary who can gather the information that is originally not easy to capture. If the privacy protection technologies are not properly applied, the privacy concerns will lead to users rejecting the sensor networks and thereby decrease the future deployments. This dissertation aims to provide privacy protection for the sensor network and CPS related applications.

Privacy preserving in wireless sensor network is challenging. First, in contrast to the fact that the privacy concerns in sensor network applications draw wide public discussions and attentions, the privacy preserving technologies have only received a few attentions within the sensor network research community. The main reason is that privacy is very broad term encompassing various personal factors and corresponding legislations, so that the definition of privacy may vary a lot given different application contexts. As the result, it is very hard to find the general technology abstraction of the privacy model in the sensor network.

Second, it is difficult to implement security schemes in sensor networks. People often rely on security schemes to solve the privacy problems. The security scheme implementation in sensor networks, however, is not straightforward due to the extremely resource constrained sensor node hardware platform. A typical sensor mote, such as MICAz [48], is only equipped with an 8MHz processor and 4KB RAM. Given such a less powered tiny device, it is not feasible to directly apply the existing security scheme, such as PGP. As we will show later, our experiments show that it takes more than 20s for the MICAz mote to do a simple 1024-bit RSA private key operation.

This poor performance indicates that the new security schemes become necessary to be used on sensor nodes.

The sensor network is a tool that allows us to automatically perform the sensing tasks and collect the sensed data. As we can see, while the services are provided, the potential privacy infringement threats exist at every sensor network operation. Our goal is to investigate the privacy problems and provide the right solutions. To better understand the privacy problems, let us first look at the sensor network operations because the privacy leakage can happen at any operation component. It is not difficult to learn that the basic sensor network operations are data collection and data delivering. Sensor nodes are deployed to collect the sensed data and deliver the data to the base station or the authorized user. Let us consider the privacy threats on these two components. First, the sensed data normally contain sensitive information and should not be disclosed to the unauthorized party. We thus identify the data privacy as the first privacy problem. Second, when the data are moved from one place to another, it may also reveal some valuable system information. For example, the adversary can use a radio detector to locate the transmitting sensor node in its detection range since all parties in the wireless communication share the common medium. Therefore, the radio detector can be used to find a specific routing path. Assuming the greedy shortest path routing scheme is used, any two non-parallel routing paths can immediately reveal the location of the base station. The adversary can also attempt traceback on the message route path and finally find the source node location. The above location information, many times, also contain system secrets and should not be disclosed to the untrusted third party. Therefore, we identify the location privacy as the second privacy problem in the sensor network. In this dissertation, we consider these two privacy problems as the representatives of the general privacy issues in sensor networks because they are the most important operation components.

The data content privacy threat is that an adversary can determine or partially determine the meaning of the transmitted message or the data stored in the sensor node's storage. The data content privacy can be preserved by security schemes including data encryption and access control. Location privacy is the secrecy that a certain event or data is associated to a specific location in the sensor network. The association of the location of the sensing nodes and the sensed data and events, in many applications, is sensitive and needs to be protected. Location privacy suggests a level of safety for the source against the adversary's discovery, or how hard for an adversary to trace back to the source. As we discussed previously, the location privacy has its uniqueness in wireless sensor networks.

There are more privacy problem besides the data privacy and the location privacy, but it is our understanding that the two privacy issues we have addressed are the most important. We consider this dissertation as the first step to study the privacy preserving in sensor networks. We believe the experiences of solving the two privacy issues can be used to understand and identify other privacy issues in sensor networks. In the following, we give the introduction to the two privacy problems in details.

1.2 Data Privacy Protection

Recent trends in sensor networks have seen the development of in-network data storage applications [107, 63] on sensor platforms with large storage capacity. In addition to data collection and forwarding, the sensor nodes now can store the data in local flash storage. As the result, the sensor is now responsible for protecting the data privacy from illegal accessing. It is tempting to simply implement existing access control schemes directly onto the sensor. However, due to limited power, memory and processing capabilities of the sensor hardware, this is not feasible.

Furthermore, access control in sensor networks differs from regular access control in that it is not enough to simply deny unauthorized users access to the data. An unauthorized user should not be allowed to use the network since network bandwidth is very limited and, more importantly, the battery power of each node may be depleted after malicious users aggressively effuse messages to the network.

To achieve access control, it is essential for sensor nodes to authenticate the identities of the requesters. The basic idea in this work is to authenticate the user locally by the sensors in the user's vicinity and transfer the endorsement of the local sensors to the remote sensor for data access. In this way, unauthorized data access request will be prohibited locally so that DoS attack trying to deplete the battery power of the network will be blocked locally. The access control proposed in this work is composed of several components to work in a secure fashion. First, the sensors in proximity need to exchange pairwise keys for secure communication. Second, the user needs to get authenticated by the local sensors either for local sensor data access or for remote sensor data access. Third, the local sensors also need to help the user and the remote sensor to build a pairwise key.

Current sensor security efforts like [28, 16, 15] have focused on solving the first component while [109] proposes a solution for local access control. There is no known solution for remote access control. The common primitive in these research is based on symmetric key cryptography. The conventional wisdom was that public key based solutions are too expensive to be efficiently implemented on the sensor platform. As a result, symmetric key has been widely accepted as a basic primitive in sensor security.

However, symmetric key primitives are not without their drawbacks. Symmetric key yields high memory overhead, increases complexity for key pre-distribution and key management, and

inherent security vulnerabilities. Since symmetric key based schemes mostly require complicated key pre-distribution, either for secret keys [28, 16, 15] or secret key spaces [26, 60], a compromised sensor node causes the system secret leakage and creates a security threat to the rest of communication links. Due to the complicated key pre-distribution and key management schemes, a large portion of the memory space has to be devoted to store the key information. Since the sensor architecture requires a low-power design, it is unlikely that memory scarcity will improve in recent years [39]. With the very limited memory budget, the real world deployment of symmetric key schemes becomes very impractical.

Recent new implementations of public key [41, 59] on sensors have shown that public key cryptography is feasible for sensor networks. However, there is little work in evaluating the public key solution in the context of a realistic sensor network application, and there is little experimental study to compare the performance of the public key and symmetric key systems on real sensor network platform. We have implemented RSA on MICAz sensors and ECC (elliptic curve cryptography) [43] on both MICAz and TelosB motes, which are widely used in sensor network research community and the symmetric key system on MICAz motes. Our experiment results clearly show that ECC is significantly more efficient than RSA for sensor networks.

1.3 Location Privacy Protection

A common communication paradigm is for the sensors to obtain information about objects or events and send data back to a base station (or sink) for further analysis. The wireless communication path from the object to the base station may jeopardize the safety of the object if an adversary is capable of detecting message flow traces back to the message source by moving along the reversed path. The object, e.g., an animal of an endangered species, or the vehicle of our aides,

may have to be protected for safety reasons and the related location information should not be disclosed. This concern will become even more serious for future sensor network prevalence in pervasive computing applications as the ubiquitous information collections doubtlessly encroach on the privacy of the people involved.

In this dissertation, we specifically focus on the source location privacy protect because the protection for destination location can be easily achieved by extending our work. We aim to hide the location of the message source and make it more difficult for an adversary to trace to the source location. We assume that the security infrastructure such as secure communication has already been built in. That is, no information carried in the message (e.g., packet head) will be disclosed for the adversary to gain any knowledge about where the message comes from. That is to say the data privacy has already been achieved. The adversary observes the wireless communication within a certain detection range and traces toward the message source by moving in each step to the node that involves in the currently detected message transmission.

Many message routing protocols have been proposed for sensor networks [49,55,102,103,45]. None of them is designed for location privacy protection. More recently, [74, 53] propose the Phantom routing to solve the similar privacy issue. However, as we will show in Section 5.5.3, the random-walk based Phantom routing has very poor performance in defending against the adversary's traceback even if the adversary has very limited traffic monitoring ability.

In our work, we address the location privacy issue under a complete adversary model. When the adversary only has limited traffic monitoring ability, we design the Weighted Random Stride (WRS) routing scheme by distributing messages flows to a geographic area with certain energy constraints to maximize the adversary's traceback time. When the adversary is more powerful, e.g., being capable of deploying an adversary sensor network to monitor the traffic, however, we

develop a random schedule scheme in which sensors transmit messages, either a valid message or a dummy packet, at a certain slot within a fixed time period, so that the adversary has no idea which sensor is delivering the real messages, and cannot determine the message flow without learning the message contents.

Location privacy suggests a level of safety for the source against the adversary's discovery, or how hard for an adversary to trace back to the source. Thus, the time for the adversary to trace back to the source is a natural metric for the location privacy. If the adversary has limited monitoring power, the adversary can follow any random message path and thus trace back to the message source. We use average traceback time and the possible minimal traceback time it takes for an adversary to reach the source starting from the sink as two metrics for location privacy. Average traceback time signifies an expected performance for the location privacy. The minimal traceback time, which shows the worst case scenario, assumes that the adversary has the best luck to take the route with the shortest time to get to the source. We assume that the adversary starts from the sink because in many applications the sink position is known and it is the best starting point for the adversary to get clue about the message flow. Nevertheless, our results can be extended to any starting point of the adversary.

When the adversaries has limited detecting power, we design routing algorithms to maximize the traceback time. We formulate this problem as an optimization problem constrained by the energy budgets that are allowed to use in message routing. To gain more understanding about this issue, we have tried to look at the problem from different perspectives. First we give an approximation to the performance bound in a generalized scenario as a guideline for network routing design. The traceback time is related to the number of nodes involved in routing. The number of nodes used to carry messages implies the degree of how spread out the scrambled

routes are, and how hard an adversary can catch a message for traceback. Then we show how to optimize the routing performance by considering several special cases in which fixed routes are given. The fixed routes are also categorized as routes that are well separated without intersection in the middle and splicing routes. Although this seems quite restricted, many applications fit in those constraints. For example, an application may requires the routes to be well separated so that the adversary has little chance to capture sufficient messages for message content decryption. And many applications also dictate fixed routes to avoid certain dangerous area that adversaries gather or enforce the routes to pass through certain points for various reasons such as information multi-cast or data aggregation.

We extend our adversary model to a more powerful one, that is, the adversary can deploy another sensor network to monitor all the communications in our network. We propose a random schedule scheme in which each node transmits at a certain time slot in a period so that the adversary would not be able to profile the difference in communication patterns among all the nodes. Our goal is to minimize the message transmission delay so that to keep the flooding period as short as possible, which is equivalent to find as many disjoint routing paths as possible. We give an approximation algorithm to find optimal k disjoint routing paths to deliver the data messages.

1.3.1 Contributions

In this dissertation, we propose a privacy framework to address data privacy and location privacy issues in the current and future CPS applications. The contributions of this dissertation can be summarized as followings.

We develop the open-source public-key cryptosystems [95], including RSA and Elliptic Curve Cryptography (ECC), for two popular sensor motes. Our further optimization work [92, 98], WM-

ECC, has significantly improved the ECC performance and established the leading position among the ever published open-source software. On TelosB/Tmote Sky motes, WM-ECC achieves 0.77s for signature time and 1.12s for verification time, three times faster than TinyECC, another popular ECC implementation done by researchers in North Carolina State University. On MICAz motes, WM-ECC also tops TinyECC by the margins of 0.65s in signature time and 0.48s in verification time. Besides the sensor motes, WM-ECC has also been ported to more resource-rich devices like PDAs [44], and effectively reduced the running time by five times compared to standard cryptography library on the market. The impact of WM-ECC is immediate. Due to its superior performance in execution time, small memory footprint and portability on all three popular sensor platforms, WM-ECC has being used by UCLA, USC, Michigan State University, Iowa State University, and more than a dozen of universities and companies around the world.

Based on WM-ECC, we investigate the user access control as an effective security mechanism to protect the data privacy. We consider two access control scenarios depending on the location that the interested data resides. When the information is stored in a local network node (within the one-hop wireless communication range), we explored local access control schemes [96, 91] to protect the information with the combination of communication efficiency and power efficiency. The local access control scheme effectively defends against the user collusion attack, which is a significant security vulnerability in the schemes based on symmetric-key cryptography. When the interested data resides in a node located at multi-hop away, we proposed a threshold based remote access control scheme [91, 99] that use local nodes to screen the unauthorized queries and endorse the legitimate request to the remote node. The beauty of the threshold based schemes is that data protection and network protection are achieved simultaneously. The threshold based idea can also be extended to address the denial of service (DoS) attack. In the PDF scheme [94] that is designed

to detect the adversary's fake messages, a group of neighboring nodes jointly generate a system digital signature for each event report. As the result, false data filtering becomes straightforward because only the report attached with a verified signature can be transmitted in the network.

Our work [97] in the privacy-aware routing design is the first to formulate the location privacy as an optimization problem. We explore the theoretical foundation for privacy-aware routing in sensor networks. Even though other research groups have worked on different routing schemes for location privacy preservation, little is known about the theoretical bounds for those schemes. We also show how to mathematically analyze the performance in terms of location privacy. This work does not consider all schemed for preserving location privacy, but examines only routing protocols in which messages follow predefined routes.

Finally, Snoogle [44] is the first research work to propose and build Information Retrieval (IR) for the physical world based on sensor devices. IR has only been used on large systems such as servers and desktop machines, and not on tiny, low cost, resource limited devices used in this dissertation. Prior work on sensor network data management investigated data query, or index building for sensor databases, but not IR. To address energy constraint and memory space limit issues, Snoogle also innovatively combines traditional techniques, such as Bloom filters and distributed top-k query, with the resource constrained sensor devices. While bloom filters have been used in other systems, the combination of this part into an IR based physical world search engine, which is built on top of sensor devices, is a new concept. Unlike the top-k algorithm proposed in prior work [29], our algorithm design addresses the challenges of message complexity problem on resource constrained sensor devices. The ECC-based flexible user access control schemes proposed in this dissertation are the first to integrate the resilient public key scheme into an access control security scheme for a practical sensor network application.

1.3.2 Organization

The rest of the dissertation is organized as follows. Chapter 2 presents the public key primitive (ECC and RSA) implementation, the foundation of our access control schemes, on MICAz, TelosB, Tmote Sky, and shows ECC is more efficient than RSA for sensors. In Chapter 3, we present our access control schemes, including pairwise key establishment, local access control and remote access control, to protect the data privacy. We evaluate the access control schemes by real world implementations. Chapter 4 investigate an effective way to discard adversary's false reports. Chapter 5 discusses the privacy-aware routing schemes to deter the adversary's traceback and protect the location privacy. Chapter 6 presents the Snoogle search engine that is based on sensor devices. Finally, chapter 7 concludes the dissertation.

Chapter 2

Public-key Cryptography Implementation on Sensor Platforms

Public-key cryptography (PKC) has been used extensively in data encryption, digital signature, user authentication, and so on. Compared to the popular symmetric key based schemes proposed for sensor networks, PKC not only provides a more flexible and simpler interface that requires no complicated key pre-distribution, but has stronger security resilience to node compromise attacks. It is a popular belief, however, in sensor network research community that public-key cryptography, such as RSA and Elliptic Curve Cryptography (ECC), is not practical because the required computational intensity is not suitable for sensors with limited computation capability and extremely constrained memory space. The nascent exploration has already disabused of this misconception. The recent progress in 1024-bit RSA implementation on Atmel ATmega128, a CPU of 8Hz and 8 bits [40], shows that a public key operation takes less than one second, which proves public-key cryptography is feasible for sensor network security related applications.

In this chapter, we details our implementation of 160-bit ECC cryptosystem on three com-

mercial off-the-shelf sensor motes: MICAz, TelosB and Tmote Sky, which are the size of two AA batteries integrating USB programming capability, an IEEE 802.15.4/ZigBee Compliant radio with integrated antenna. The MICAz mote features a 8-bit, 8MHz Atmel microcontroller with 4KB RAM, 128KB programmable ROM, and optional external memory for data collection. The TelosB and Tmote Sky share the same hardware platform, they both are equipped with a 16-bit, 8MHz TI MSP430 processor with 10KB RAM, 48KB programmable ROM and 1MB on-board flash memory for data collection. For the comparison purpose, we also implement the 1024-bit RSA cryptosystem on MICAz. Our results show why ECC is a better public key scheme than RSA for the sensor motes.

The fundamental operations in RSA and ECC cryptosystems are large integer arithmetics over the finite field. To efficiently perform RSA and ECC exponentiations on the low-power CPUs of sensor motes, it is essential to optimize the expensive large integer operations. In particular, multiplication and reduction are most dominant operations in both RSA and ECC. Since most CPU cycles are consumed in these two integer operations, the efficiency of these two integer operation modules directly determines the performance of the encryption and decryption. The low-power CPUs have very limited number of registers (only 32 8-bit registers in ATmega 128). The large integer operands cannot be loaded into the registers at one time, so that the latency of memory accesses have to be paid for operand loading and storing between registers and memory. The implementation challenge is to reduce the number of such memory accesses. In this work, we adopt the hybrid multiplication method [41], which is a very effective way to reduce the number of memory accesses. To precisely control the register and memory operations, we implement this module in assembly language. Our experiments demonstrate that the hybrid multiplication is at least 7 times faster than the conventional multi-precision multiplication programmed in C

language. The modular reduction can also be optimized under certain conditions. For example, when the modulus is a pseudo-Mersenne number, the reduction can be greatly optimized and be finished more than 10 times faster than the classic long division method.

In addition to the optimizations of the big integer operation. RSA and ECC can be further optimized. Montgomery reduction can be applied to efficiently calculate the RSA exponentiation. Chinese Remainder Theorem (CRT) can be used to reduce the exponent sizes and speed up the RSA exponentiation for up to 4 times. In ECC, we apply a mixed coordinate, the combination of Affine coordinate and Jacobian coordinate, to do ECC exponentiation, so that some expensive operations can be avoided (e.g., inversion) or reduced (e.g., multiplication and squaring). The rest of optimizations include Sliding-Window method [56], Non-adjacent Form [68], and Shamir's trick. It is possible to further reduce the computation time by using extended instruction set proposed in [41].

Our experiments show that ECC can efficiently run on all three sensor platforms. On MICAz motes, it takes 1.35s to generate a signature, and 1.96s to perform a signature verification. ECC is more efficient on Tmote Sky by taking the advantage of 8MHz and 16-bit CPU. The signature generation and verification on Tmote are 0.77s and 1.12s, respectively. Since TelosB mote, sharing the same hardware with Tmote, can only run at 4MHz, the performance on TelosB is 1.54s for signature and 2.25s for verification. In comparison, RSA has more computational overhead than ECC on sensor motes. Even though the RSA public key operation can be efficient, which takes 0.79s on MICAz with a 17-bit public key, its private key operation consumes 21.5s on the same hardware.

2.1 RSA Introduction

In RSA cryptography, a user, say Alice, has two keys: a public key (e) and a private key (d). Alice publishes her public key and keeps private key in secret. When Bob wants to send a message m to Alice, and does not want any other to know the message contents, he just encrypts m by using Alice's public key. Without the private key, it is computationally infeasible for others to decrypt the ciphertext. After receiving the encypted message from Bob, Alice uses her private key to decrypt the message.

The security of RSA scheme is based on the difficulty to factor a large integer (n). Here we briefly go over the key generation procedure and encryption/decryption in RSA. Alice needs to take following steps to get her public key e and private key d.

- Pick two random large prime number p and q, so that $p \neq q$;
- Compute $n = p \times q$;
- Compute the totient: $\phi(n) = (p-1)(q-1)$;
- Choose an integer e as the public key so that $1 < e < \phi(n)$, and e is co-prime to $\phi(n)$;
- Compute the private key $d = e^{-1} \pmod{\phi(n)}$.

To encrypt a message m, Bob computes $c = m^e$ and sends cipher text c to Alice. The decryption for Alice is to raise the value of her private key to the power of the ciphertext c, so that $c^d = (m^e)^d = m^{ed} = m \pmod{n}$. The decryption procedure works due to following reasons. Because $e \times d \equiv 1 \pmod{(p-1)(q-1)}$, we have $e \times d \equiv 1 \pmod{(p-1)}$ and $e \times d \equiv 1 \pmod{(q-1)}$. Applying *Fermat's little theorem*, we get $m^{ed} \equiv m \pmod{p}$ and $m^{ed} \equiv m \pmod{q}$. Applying *Chinese Remainder Theorem* (CRT), we have $m^{ed} \equiv m \pmod{n}$.

In practice, RSA must be combined with certain padding scheme to defend against security attacks, such as Adaptive Chosen Cipher Text attack. The popular padding schemes include Optimal Asymmetric Encryption Padding (OAEP) and Probabilistic Signature Scheme for RSA (RSA-PSS). For the simplicity, we do not cover the padding scheme implementation in this work.

2.2 ECC Introduction

We briefly give a background introduction about elliptic curve cryptography, and corresponding elliptic curve Digital Signature Algorithm.

2.2.1 Elliptic Curve Cryptography

In recent years, ECC has attracted much attention as the security solutions for wireless networks due to the small key size and low computational overhead. For example, 160-bit ECC offers the comparable security to 1024-bit RSA. An elliptic curve over a finite field GF (a Galois Field of order q) is composed of a finite group of points (x_i, y_i), where integer coordinates x_i, y_i satisfy the long Weierstrass form:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.1}$$

and the coefficients a_i are elements in $GF(q)$. Since the field $GF(q)$ (q is a prime) is generally used in cryptographic applications, (2.1) can be simplified to:

$$y^2 = x^3 + ax^2 + b, \tag{2.2}$$

where $a, b \in GF(q)$.

The elliptic curve points form an additive abelian group, so that the addition of any two points is a point in the group. Given two points P and Q, with the coordinates (x_1, y_1), (x_2, y_2), respec-

tively, the addition results in a point R on the curve with coordinate (x_3, y_3), where x_3 and y_3 satisfy

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3), \tag{2.3}$$

such that

$$x_3 = L^2 + L + x_1 + x_2 + a, \tag{2.4}$$

$$y_3 = L(x_1 + x_3) + x_3 + y_1, \tag{2.5}$$

where

$$L = (y1 + y2)/(x1 + x2) \tag{2.6}$$

If $x_1 = x_2$ (note $x_1 + x_2$ is 0), then R is defined as a point at infinity, O. O is an identity element of the group. Each element in the group has an inverse that satisfies $P + (-P) = O$, and $(-P) + P = O$. Also, $P + O = O + P = P$. If $P = Q$, then $R = P + P = 2P$, and coordinate (x_3, y_3) is derived by

$$x_3 = L^2 + L + a, \tag{2.7}$$

$$y_3 = {x_1}^2 + (L + 1)x_3, \tag{2.8}$$

where

$$L = x_1 + y_1/x_1. \tag{2.9}$$

The ECC relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem, that is, given points P and Q in the group, it is hard to find a number k such that $Q = kP$.

2.2.2 Elliptic Curve Digital Signature Algorithm (ECDSA)

ECC signature is based on Digital Signature Algorithm. We assume Alice sends a message to Bob. To convince Bob that the message does come from Alice, Alice needs to apply a digital signature

for the message so that Bob can verify it by using Alice's public key. Initially, Alice and Bob have to agree on a particular curve with base point P over the field $GF(p)$, and the order of P is q. When Alice sends a message to Bob, she attaches a digital signature (r,s) generated by following steps (suppose Alice has a private key x and a public key $Q = xP$).

1. Choose a random key k in $[1, q-1]$;
2. Compute kP, yield a point with coordinate (x_1, y_1). Let $r = x_1 \pmod q$. Check r, go back to the first step if the result is zero;
3. Compute $k^{-1} \pmod q$;
4. Compute $s = k^{-1}(Hash(m) + xr)$, where $Hash$ is a one-way hash function. Again, check s, go back to the first step if $s = 0$;
5. (r,s) is the digital signature.

To verify the message m and the signature, Bob needs to do following steps.

1. Compute $w = s^{-1} \bmod q$ and $H(m)$;
2. Compute $u_1 = H(m) \cdot w \bmod q$ and $u_2 = r \cdot w \bmod q$;
3. Compute $u_1 P + u_2 Q$, get the result point (x_2, y_2);
4. The signature is verified if $x_2 = r$.

Finally, Bob compares the value of x_2 and r, and accepts the message only if x_2 equals to r.

2.3 Implementation

In this section, we describe the implementation of RSA on MICAz motes and the implementation of ECC on three most popular sensor devices: MICAz, TelosB and Tmoet Sky. Given the limited processor resources, we concentrate most of our efforts on computation optimization. The fundamental RSA operation is large integer exponentiation over a finite field $GF(n)$, where n is the product of two large prime number p and q. The computation of the exponentiation can be decomposed to a series of squaring, multiplications and reductions. In addition, we also need an inversion module to calculate the public key and private key pair given two prime numbers p and q. In this section, we first present our optimization in general large integer operations. Based on that, we describe our further optimization by using Montgomery reduction and Chinese Remainder Theorem (CRT) to significantly improve the computation efficiency. The fundamental ECC operation is large integer arithmetics over either prime number finite field $GF(p)$ or binary polynomial field $GF(2^m)$ (where m is a prime). Because the two heavily used operations: multiplication and modular reduction, can be more effectively optimized if pseudo-Mersenne primes are picked for elliptic curves compared with those of binary field [41], we limit our discussion in prime number finite field $GF(p)$ in this thesis. Without further clarification, our discussion of ECC implementation is based on SECG recommended 160-bit elliptic curve: secp160r1. We first describe the optimized large integer operation modules, which can be used for both RSA and ECC cryptosystems. Then we focus on the protocol related optimizations specifically for RSA and ECC, respectively.

2.3.1 Large Integer Operations

Large integer arithmetic operations include addition, subtraction, shifting, multiplication, division and modular reduction. Here we focus on the four most important functions: multiplication (including squaring), modular division, modular reduction and inversion.

2.3.1.1 Multiplication and Squaring

The multiplication (or squaring) is the key component in RSA implementation because the exponentiation is basically computed by multiplications and squaring. We have compared three different multiplication implementations [41, 62, 59], and finally decided to use Hybrid Multiplication proposed in [41]. To ease our explanation, we use three large integers as the examples for our following discussion: $A(a_{n-1}, a_{n-2}, \cdots, a_1, a_0)$, $B(b_{n-1}, b_{n-2}, \cdots, b_1, b_0)$, and $C(n_{2n-1}, c_{2n-2}, \cdots, c_1, c_0)$, where $C = A * B$. A and B both have length of n words, each word has k-bit size. The product C has $2n$ words.

The Hybrid multiplication is the combination of Row-wise multiplication and Column-wise multiplication. The Row-wise method fixes the multiplier b_i ($0 \leq i \leq n$), and multiplies it with every word of multiplicand A. Partial results are stored in $n+1$ accumulator registers. Every time one row is finished, the last accumulator register can be stored to memory as the part of final results. On average, one memory load is required for each $k \times k$ multiplication. When integer size n is increased, the required number registers increase linearly in Row-wise method. For 1024-bit RSA, a typical multiplication is between two 128-byte large integers. Given only 32 registers in ATmega128, Row-wise multiplication can not be directly applied.

The Column-wise method, on the other side, computes the partial results of $a_i * b_j$ (where $i + j = l$) for column l. After one column finishes, the last word of accumulator registers is stored

as the part of final result. The Column-wise method only requires three accumulator registers and two more for operands. However, two memory load operations are required for each $k \times k$ multiplication.

The Hybrid method takes advantages of Row-wise and Column-wise strategies. To optimize the memory operation, the Hybrid method merges a number (d) of columns together, and then conducts Row-wise multiplication in each merged column. When d equals to 1, the Hybrid method becomes the Column-wise multiplication. When d equals to n, then it becomes Row-wise method. A larger d leads to fewer memory operations, but requires more registers. A small d, however, requires more memory operations and consumes more CPU cycles. Balancing the advantages and disadvantages, we implement the Hybrid multiplication with column width $d = 4$, which requires 9 accumulator registers, 5 operand registers, 6 pointer registers (point to A, B and C), and others for temporary storage and loop control.

We implement the Hybrid multiplication in assembly language. For the comparison purpose, we also implement a standard multi-precision multiplication program in C language. Our experiments show the standard C program needs $122.2ms$ to finish the multiplication between two 128-byte integers, while it only takes $17.6ms$ for our Hybrid multiplication to do the same computation, which is more than 7 times faster.

The squaring is a special case of the multiplication, which has the same the multiplicand and the multiplier. Given an m-bit large integer $A = (A_1, A_0)$, where A_1, A_0 are two halves, $A^2 = A_1A_1 \times 2^m + 2A_1A_0 \times 2^{m/2} + A_0A_0$. Therefore, we can take advantage of the fact that A_1A_0 only needs to be calculated once. Compared with the multiplication, the optimized squaring can reduce the computational complexity up to 25%.

2.3.1.2 Modular Division

Modular division is another expensive operation in ECC. In Affine coordinate, each ECC operation of point addition and doubling requires a modular inversion. The integer inversion is also required for ECC digital signature generation and verification. In our implementation, we adopt the Great Divide scheme proposed in [83]. We briefly explain the algorithm in the followings.

Given an denominator x and numerator y, we want to compute the modular division $\frac{y}{x}$ over $GF(p)$. This is equivalent to find r, so that

$$r \equiv \frac{y}{x} (\text{mod q}). \tag{2.10}$$

To find r efficiently, the algorithm maintains following two invariant relationship:

$$A * y \equiv U * x, \text{ and } B * y \equiv V * x, \tag{2.11}$$

where A, B, U,and V are four auxiliary variables and initialized with values x, q, y, and 0, respectively. Note the two relationship is true with the initial values. The algorithm intuition is to reduce the value of A to 1, so that the first relationship in (2.11) will become $y \equiv U * x$, and U will be the result. The procedure is conducted in following way. When A is even, we can divide A by 2. Correspondingly, U has to be divided by 2 to keep the relation true. If U is not even at that time, we can make it become even by adding U with the modulus. When A is odd, we use the 2nd relationship to help to reduce A. If B is even, we keep dividing B by 2 similarly to make B odd. Then we add the two relation together and the divide the result value by 2 at the both sides. By repeating this process, it is guaranteed that either value of A or B reduces one bit in one iteration. The procedure stops when $A = B = 1$, the first equation becomes $y \equiv U * x$. The value of U is

our final result. If we initialize U with 1, this routine can be used to calculate an inversion of x. This algorithm works when x and q are relatively prime. The Great Divide finishes division or inversion operation in $2(log(x)-1)$ steps. Great Divide is much faster than the long division method because Great Divide only needs addition operations in each iteration, while long division method requires multiplications. Unfortunately, Great Divide cannot be used in RSA to calculate the public key and private key. The reason is that Great Divide only works when the modulus is an odd number, but the totient $\phi(n) = (p-1)(q-1)$ in RSA is always even. Therefore, we use Extended Euclidean algorithm instead.

2.3.1.3 Modular Reduction

The modular reduction operation is another important module because each multiplication or squaring must be followed by a reduction operation. The classic reduction method is using long division. Although the long division method is a general method for calculating the modular reduction, it is also the slowest method. In ECC cryptosystem, the modular reduction operation is as important as modular multiplication. Each multiplication must be followed by a reduction operation. Since we choose to use pseudo-Mersenne primes as specified in NIST/SECG curves, the modular reduction can be optimized by conducting a fixed number of integer additions. Because the optimization is curve specific, we will explain in more details in the section of ECC operation.

Now, we discuss the modular reductions in RSA and ECC digital signature generation and verification. In most cases, the modulus is not a pseudo-Mersenne prime, the optimization cannot be applied for those reduction calculation. We choose the classic long division method to implement this operation. Fortunately, the number of this type of modular reduction is very limited, it does not affect the overall performance much. We briefly describe the long division method as in

Algorithm 1.

The long division producer reduces the remainder of x by one byte in each iteration.

2.3.1.4 Inversion

The multiplicative inversion is required to calculate the RSA public key and private key pair. A RSA public key e and a private key d should satisfy the condition: $e \times d \equiv 1 \bmod \phi(n)$, where totient $\phi(n) = (p-1)(q-1)$. Given a public key e, the corresponding private key is the multiplicative inversion of e. Since both p and q are prime numbers, $\phi(n)$ must be even. Thus, the efficient Great Divide scheme [83] can not be used because Great Divide requires the modulo to be an odd number. We use the classic Extended Euclidean Algorithm to compute the private key. The algorithm is described as below.

2.3.2 RSA Optimization

With the basic large integer operation modules implemented, we have conducted the first performance test for RSA public key operation (17-bit public key) and private key operation (1024-bit private key). Surprisingly, both operations are very slow. It takes 4.6s to finish the public key operation and 389s to do a private key operation.

To learn the reason for the poor performance of our initial implementation, we profiled the every operation in RSA exponentiation. We found that the modular reduction following each multiplication consumes 0.13s on the average. For 17-bit public key, there are totally 17 such reductions, which spend 2.2s in total, almost 50% of the execution time of the public key operation.

We explore two optimization schemes which aim to reduce the costs of the reduction and multiplication operations.

2.3.2.1 Montgomery Reduction

Montgomery reduction [66] is a method to efficiently perform the modular reduction without doing expensive division. For example, suppose we want to compute T modulo N, the algorithm says it is easy to compute TR^{-1} (mod N) (without any division), where R is a radix ($R > N$) and co-prime to N. We do not validate this algorithm in this chapter. Interested reader may refer to [66] for details. It seems this algorithm does not save anything because an extra step to convert TR^{-1} (mod N) to T modulo N is required. However, this method is useful if a number of computations are needed for the same modulus N. That is the reason that Montgomery reduction is widely used to reduce the reduction cost for the exponentiation operation in RSA. The efficient exponentiation by using Montgomery reduction is described as below. The idea is to convert integer b to an N-residue so that $b^a * 2^k$ (mod n) can be quickly computed without doing any reduction. As the result, we only need to do two reductions for the exponentiation. The first one is to convert b to N-residue before the Montgomery reduction, the second one is to convert the exponentiation result from N-residue back to integer. Having implemented the Montgomery reduction module, the performance of RSA public key and private key operations have been improved significantly to 1.2s and 82.2s, respectively.

2.3.2.2 Chinese Remainder Theorem (CRT)

The complexity of the exponentiation in RSA largely depends on the the size of modulus n and the exponent (either public key or private key). Chinese Remainder Theorem (CRT) can be used to effectively reduce the computational complexity of exponentiation by reducing the size of both n and the exponent. CRT can be found in any number theorem textbook, here we only give a simple example to serve for this chapter. Let number n_1, n_2 be positive integers which are co-prime to

each other, i.e., $GCD(n_1, n_2) = 1$. Let $n = n_1 * n_2$ and x_1, x_2 be integers. CRT states that if there are congruence: $x \equiv x_1 \pmod{n_1}$, $x \equiv x_2 \pmod{n_2}$, then there is only one solution x between 0 and $n-1$, inclusively. The value of x can be determined by

$$x = x_1 * r_1 * s_1 + x_2 * r2 * s_2 (\text{ mod n}), \tag{2.12}$$

where $r_i = \frac{n}{n_i}$, $s_i = r_i^{-1} \pmod{n_i}$ for $i = 1,2$. Based on the above simple version of CRT, we describe our RSA optimization (adapted from [37]) by using CRT. Note step 3 and 4 can be precomputed. Th above algorithm reduces the size of a and d in half. Consider a and d are both 1024-bit integers, the computation of a^d is reduced to 2 modular exponentiation with both base and exponent size of 512 bits. Thus, the overall computational complexity is reduced to roughly 1/4 of the original exponentiation. The CRT can also be applied for public key operation, but the computational complexity can only be reduced by 50%. The reason is that public key size is normally very small (17 bit in our experiment), so the exponent size cannot be reduced in this case. With CRT implemented, the public key operation has been reduce to 0.79s. Correspondingly, the private key operation is reduced to 21.5s, approximately 1/4 of the time before doing CRT.

2.3.3 ECC Optimization

2.3.3.1 Addition and Doubling

The fundamental ECC operation is point addition and point doubling. The point multiplication can be decomposed to a series of addition and doubling operations. As discussed in previous section, point addition and doubling in Affine coordinate require integer inversion, which is considered much slower than integer multiplication. Cohen *et al.* showed that these operations in Projective coordinate and Jacobian coordinate yield better performance [21]. They further found addition

and doubling in mixed coordinate, with the combination of Modified Jacobian coordinate and Affine coordinate, lead to the best performance [22]. Consider an ECC point in Modified Jacobian coordinate, $P_1(X_1, Y_1, Z_1, aZ_1^4)$, and a point in Affine coordinate, $P_2(x_2, y_2)$, their addition results in the third point $P_3 = (X_3, Y_3, Z_3, aZ_3^4)$ in Modified Jacobian coordinate. The result is given by following equations.

$$\begin{aligned} X_3 &= -H^3 - 2X_1H^2 + r^2, \\ Y_3 &= -Y_1H^3 + r(X_1H^2 - X_3), \\ Z_3 &= Z_1H, \\ aZ_3^4 &= aZ_3^4, \end{aligned} \tag{2.13}$$

where $H = x_2Z_1^2 - X_1$, and $r = y_2Z_1^3 - Y_1$. The result of point doubling for $P_3 = 2P_1$ is given by following formula.

$$\begin{aligned} X_3 &= T, \\ Y_3 &= M(S-T) - U, \\ Z_3 &= 2Y_1Z_1, \\ aZ_3 &= 2U(aZ_1^4) \end{aligned} \tag{2.14}$$

To estimate the computational complexity, we only consider large integer multiplication and squaring operations, and ignore those addition and subtraction since they are much faster. According to Eq.2.13 and Eq.2.14, point addition requires 9 large integer multiplications and 5 squaring, and point doubling requires 4 multiplications and 5 squaring.

The basic point operations can be further optimized for specific elliptic curves. In our case, the curve parameter a of secp160r1 equals to -3. For point doubling, M can be further reduced to

$$M = 3X_1^3 - 3Z_1^4 = 3(X_1 + Z_1^2)(X_1 - Z_1^2). \tag{2.15}$$

As the result, point doubling operation reduces to 4 multiplications and 4 squaring. Actually, aZ_3^4 does not have to be calculated in point addition, so the computational complexity reduces to 8 multiplications and 3 squaring. Our observation supports the choice of mixed coordinate, the performance of point multiplication improves around 6% compared with our previous implementation in Jacobian coordinate.

2.3.3.2 Modular Reduction

Recall that modular reduction has to be applied after every large integer multiplication, it is also a performance critical operation. By taking advantage of pseudo-Mersenne primes specified in SECG curves, the complexity of the modular reduction operation can be reduced to a negligible amount. In this section, we use curve secp160r1 as the example to show how to do efficient reduction.

Suppose we use the 8-bit architecture, the multiplication result of two 160-bit integers can be represented by

$$C(c_{39}, \cdots, c_{20}, c_{19}, \cdots, c_1, c_0),$$

where c_i $(0 \leq i \leq 39)$ is a word with 8 bits, and c_{39} is the most significant word. The 40-word integer can also be written as:

$$C = (c_{39}, \cdots, c_{20}) * 2^{160} + (c_{19}, \cdots, c_1, c_0) \tag{2.16}$$

Given the field of curve secp160r1 $q = 2^{160} - 2^{31} - 1$, we can have $2^{160} \equiv 2^{31} + 1$. Therefore,

$$\begin{aligned} C &\equiv (c_{39}, \cdots, c_{20}) * (2^{31} + 1) + (c_{19}, \cdots, c_1, c_0) \\ &\equiv (c_{39}, \cdots, c_{20}) * 2^{31} + (c_{39}, \cdots, c_{20}) + (c_{19}, \cdots, c_1, c_0) \end{aligned} \tag{2.17}$$

Since each word has 8 bits, the first term in the result of Eq. 2.17 can be further reduced to

$$\begin{aligned}(c_{39},\cdots,c_{20})*2^{31} &\equiv (c_{39},c_{38},c_{37})*2^{167}+c_{36}*2^{159}+(c_{35},\cdots,c_{20})*2^{31}\\ &\equiv (c_{39},c_{38},c_{37})*2^{38}+(c_{39},c_{38},c_{37})*2^{7}+(d_7d_6\cdots d_0)*2^{31}+(d_7d_6\cdots d_0)+(d_0)*2^{159}\end{aligned} \tag{2.18}$$

where $(d_7,\cdots,d_1,d_0)$ are 8 bits of c_{36}. Now, all terms in Eq.2.17 and 2.18 have at most 159 bit length, the reduction result is simply the addition of these terms.

2.3.3.3 Further Optimization

Examining the computational complexity, we notice that point addition is more expensive than point doubling. As we have discussed, point multiplication can be decomposed to a series of point addition and doubling, we would rather use more point doubling than point addition to compute the point multiplication. Morain *et al.* found Non-adjacent forms (NAFs) is an effective way to achieve the lightest Hamming weight for scalar k in point multiplication $k*P$, which results to use the least number of point additions to calculate $k*P$ [68]. For example, $255*P$, or $(11111111)*P$, requires 7 point additions. But if we transform it to $(10000000-1)*P$, which is $256*P-P$, only one addition is required. Note the point subtraction can be replaced by point addition because the inverse of an Affine point $P=(x,y)$ is $-P=(x,-y)$. We implement NAFs technique in random point multiplication. According to our experiments, point multiplication with NAFs contributes at least 5% performance improvement.

Recall in the digital signature procedure in ECDSA, component r is generated by a point multiplication with the fixed base point of a selected elliptic curve. To further reduce the execution time, we precompute some partial results and apply sliding window method [56] to speed up fixed point multiplication. Different from NAFs, sliding window scheme groups scalar k into a number of $s-bit$ bit-clusters, where s is also called window size. So, k can be represented by $k_m*2^{sm}+$

$k_{m-1} * 2^{s(m-1)} + \cdots + k_0$, where k_i is a bit-cluster. If we precompute the point multiplication with every possible value of k_i, the number of point addition is bounded by $\lceil \frac{160}{s} \rceil - 1$. Note the sliding window method does not reduce the number of point doubling operations. Obviously, this scheme requires extra memory space for storing partial results. In practice, we select window size $s = 4$. Correspondingly, there are 16 entries in the partial result table. Our experiments show sliding window method is more effective than NAFs for fixed point multiplication, the performance of sliding window method is more than 10% better than that of NAFs.

Our initial experimental results indicated that it took double amount of time to perform an ECDSA verification than to do an ECDSA signature: signature is 1.35s, while verification is 2.85s. The reason is that the verification requires two ECC point multiplications (while the signature only needs one point multiplication); the verifier has to perform $u_1P + u_2Q$ as shown in Section 2.2.2. To speed up the verification time, we adopt Shamir's trick [43] to do multiple point multiplication simultaneously. The idea of Shamir's trick is similar to the sliding window method discussed previously. Given t-bit u_1 and u_2, we use the window size ω and precompute the values $iP + jQ$ for $0 \leq i, j \leq 2^{\omega}$. At each of $\lceil t/\omega \rceil$ steps, we perform ω doubling and the (precomputed) additions determined by the window contents. The larger the window size (ω) is, the more memory is required for storing the precomputed values. In practice, we choose the single bit window size, $\omega = 1$. Therefore, only the value of $P + Q$ needs to be precomputed and stored. As the result, the performance of ECDSA verification has been improved more than 30%, from 2.85s to 1.96s. There is still further improvement space if multi-bit window size is used, but the trade-off is more memory overhead.

2.4 RSA Evaluation

In this section, we describe the experimental performance of 1024-bit RSA on our MICAz motes. We first present our experimental results and related issues during the implementation. We then give the performance analysis to quantify the computational complexity.

2.4.1 Experimental Results and Implementation Challenge

In the experiment, we randomly select two 512-bit prime number as p and q. For the public key operation, we choose a small exponent of $e = 2^{16} + 1$, which is commonly used value for e. Our program uses 15,832 byte code size and 3,224 byte data size. Compared with RSA implementation in [41], our code size is much larger because of the assignments of precomputation values during initialization stage. Our implementation spends 0.79s to finish a publick key operation and 21.5s to do a private key operation.

The biggest challenge to implement 1024-bit RSA on MICAz motes is the memory constraint. MICAz mote only has 4KB RAM, which is the total space can be used by data and program stack. Since the operands in 1024-bit RSA are mostly 128 integers, the subroutines, such as modular reduction, Extended Euclidean Algorithm and Montgomery reduction, have to reserve considerable amount of memory space for storing temporary results. In addition, for optimization purpose, a number of pre-computations are required. In our program, 1152 bytes of memory are used for storing system parameters, such as p, q and n, and precomputation results, such as R_p, R_q in CRT. Therefore, attentions need to be paid not to waste any memory usage. In practice, we have adopted two methods to save the memory space. First, we declare more global variables. The idea is to share the memory space among different subroutines in each module. Note this method is only good for those subroutines do not call each other. Otherwise the intermediate data will be

lost. Second, we conduct every possible precomputation so that some module may not be required during the RSA operation in the real time. For example, the Extended Euclidean algorithm is only used to find the public/private key pairs and to precompute the parameters used in Montgomery reduction. Actually we do not need this module in the real time. This helps us a lot because it consumes almost 1KB temporary space.

2.4.2 Performance Analysis

To analyze the computational complexity distribution among the components in RSA exponentiation, we profile the execution time of multiplication, squaring, and modular reduction modules, the three most time consuming operations in RSA exponentiation. The profiling information is shown in Table 2.1.

Module	Operand Sizes (bytes)	Execution Time (ms)
MUL.	128 by 128	17.1
MUL.	64 by 64	4.48
SQR.	128 by 128	14.1
SQR.	64 by 64	3.87
MOD.	256/128	132
MOD.	192/128	74
MOD.	128/64	40

Table 2.1: Execution time profiles of some important modules.

Our analysis assumes that all optimization schemes have been applied in RSA exponentiation. To simplify the presentation, we denote "MUL" as, large integer multiplication, and let "SQR" be

large integer squaring, and let "MOD" be large integer modular reduction. A "m/n" MOD means a MOD operation for a m-byte integer over a modulus with n-bytes. For example, 128/64 MOD denotes a modular reduction of a 128 byte integer with a 64 byte modulus.

Let us consider an example of RSA operation to calculate $M = C^x \pmod{n}$, where x can be either public key or private key. Following the CRT algorithm, we first do two MODs to calculate C_p and C_q. Then, we conduct two Montgomery reductions to get M_p and M_q. Finally, two MULs, one MODs and one addition are required to compute M. Note the last two steps in CRT, which requires 2 MODs, can be simplified by doing addition first and then only one MOD. Except the Montgomery reduction, both public key and private key operation need to do two 128/64 MODs, two 128×128 MULs, one 192/128 MODs operations, which totally account for $2 \times 40 + 2 \times 17.1 + 74 = 188.2ms$.

The difference of execution time between public key and private key operations is at exponentiation part. Each Montgomery reduction requires two 64×64 MULs, one 128-byte addition and possible another 128-byte subtraction. The cost of addition and subtraction can be ignored. Therefore, the execution time of each Montgomery reduction is $2 \times 4.48 = 8.96ms$. Since we choose the public key to be $2^{16} + 1$, there are totally 16 64×64 SQRs and 1 64×64 MUL in the exponentiation. According to Table 2.1, the total time for SQRs and MUL with Montgomery reduction should be $16 \times 3.87 + 4.48 + 17 \times 8.96 = 218.7ms$. In addition, two 128/64 MODs are needed to convert operands between integer and N-residue before and after each exponentiation. For CRT optimization, we need to do two 512-bit exponentiations. Therefore, the exponentiation execution time for public key operation is $2 \times (218.7 + 2 \times 40) = 597.4ms$. Combined with the rest operations in CRT, the public key operation consumes $594.4 + 188.2 = 782.6ms$, which matches our test result very well.

For the private key operation, the number of SQRs is 511 (after CRT) in each reduced exponentiation. The number of MULs depends on the Hamming weight of the exponent. Our experiment shows the average Hamming weight of D_p and D_q of our private key is 278. Hence, there are 277 MULs required in each exponentiation. Therefore, the execution time for each exponentiation is $511 \times 3.87 + 277 \times 4.48 + 788 \times 8.96 = 10279ms$. Since the exponentiation execution time in private key operation overwhelmingly dominates other operations, we only need to consider the execution time of exponentiations only. Two such exponentiations consumes 20.5s, closely matching our experiment result of 21.5s.

2.5 ECC Evaluation

In this section, we first present the performance of our implementation. Then we give an overall analysis to quantify the computation complexity.

2.5.1 The performance of ECC Implementation

In experiments, we measure execution time and code size of our implementation. We choose secp160r1 as the elliptic curve in all experiments. We use the embedded system clock (921.6kHz for MICAz and 32.6kHz for TelosB/Tmote Sky) to measure the execution time of major operations in ECC, such as point multiplication, point addition and point doubling.

We first test point multiplication operation, which is comprised of point addition and doubling. We consider two cases in point multiplication. One is multiplying large integer with a fixed point(base point), and the other one is with a random point. Fixed point multiplication allows for optimization by precomputing. We apply sliding window technique [56] and set window size to

4, i.e., precomputing $2^4 - 1 = 15$ points. In experiments, we randomly generate 20 large integers to multiply with the point and take the average execution time as the result.

Since ECC point multiplication consists of addition and doubling operations, we further evaluate these two operations individually. We generate random points and large integers for tests. Since a single operation takes very little time, to reduce the error of clock inaccuracy, we measure 100 operations every round and take the average value.

	FPM	RPM	PAdd	PDbl	SIGN	VERIFY
MICAz	1.24s	1.35s	6.2ms	5.8ms	1.35s	1.96s
Tmote	0.74s	0.77s	3.7ms	3.5ms	0.77s	1.12s
TelosB	1.44s	1.55s	7.3ms	7.0ms	1.55s	2.25s

Table 2.2: The comparison of ECC execution Time on three mote platforms, including fixed point multiplication (FPM), random point multiplication (RPM), point addition (PAdd) and point doubling (PDbl) and ECDSA signature generation (SIGN), verification (VERIFY) time.

We summarize the performance in Table 2.2, including ECC fix point multiplication (with size-4 sliding-window optimization) (FPM), random point multiplication (RPM), point addition (PAdd), point doubling (PDbl), ECDSA signature (SIGN) and verification (VERIFY). It clearly shows that the performance of ECC operation on MICAz is slightly better than that on TelosB, even though TelosB is equipped with an 8MHz, 16-bit CPU. After a careful investigation, we found the performance degradation on TelosB is due to the following two reasons. First, the 8MHz CPU (MSP430) frequency on TelosB is just a nominal value. The maximum CPU clock rate is actually 4MHz. Second, the hardware multiplier in MSP430 CPU uses a group of special peripheral registers which are located outside of MSP430 CPU. As the result, it takes MSP430 eight CPU cycles to perform an unsigned multiplication, while it at most takes four cycles to do

the same operation in ATmega CPU. The above two reasons explain why TelosB cannot perform better than MICAz.

Tmote Sky is capable of running at 8MHz CPU frequency instead of 4MHz on TelosB because it is equipped with an external resistor on the ROSC pin of MSP430 that enables the DCO to operate at a higher frequency. We simply enable the external resistor on Tmote and achieve the ECC performance twice faster than that on TelosB. As shown in Table. 2.2, it only takes 0.77s to finish a signature generation and 1.12s to verify it.

	ECC library		ECDSA		UART Comm.	
	ROM	RAM	ROM	RAM	ROM	RAM
MICAz	10,360	978	8,244	202	3,452	147
TelosB/Tmote	7,018	1,012	4,420	164	3,202	233

Table 2.3: ECC implementation code size.

Table 2.3 presents code sizes and data sizes of the ECC implementations. For TelosB and Tmote Sky platforms, the ECC library uses 7,018 byte ROM (for code) and 1,012 byte RAM (for data). Note more than 60% of data size is used to store the 15 elliptic points which are used in sliding-window optimization. When the data size budget is tight, the sliding-window optimization can be removed to have more data space. ECDSA module accounts for 4,420 bytes on TelosB and Tmote Sky. The reason is the included SHA1 module consumes around 3KB code size. Finally, for the IO purpose, we also have the UART communication module, which uses 3,202 bytes for code and 233 bytes for data. The total code size of our test program is 19,290 bytes.

Compared to TelosB and Tmote Sky, our ECC package is more space demanding on MICAz platform. The ECC library requires 10,360 bytes in code size for MICAz, 46% more than that

on TelosB/Tmote. This is due to our assembly codes for optimizing the large number integer operations. Since the CPU register number in MICAz is twice the amount that in TelosB/Tmote, more instructions are needed to handle the extra register operations. For the same reason, the code size of ECDSA requires 8,244 bytes. Overall, the test program on MICAz uses 24,258 bytes for code and 1507 bytes for data.

2.5.1.1 A Performance Anatomy of ECC Point Multiplication on MICAz

Since ECC point multiplication dominates the computational complexity in ECC signature and verification, we are curious to learn the performance anatomy in ECC point multiplication.

This analysis is based on 160-bit ECC curves. We use secp160r1 as the example. We also assume 4-bit sliding window method is used, and partial results are precomputed. The computational cost for each window unit is 4 point doubling and 1 point addition. Given a 161 bit private key, there are 41 window units. Totally , 164 point doubling and 41 point additions are required to finish 1 point multiplication.

Large (160-bit) integer multiplication, squaring and reduction are the most expensive operations in point doubling and point addition. To learn the amount of time contributed by the above three operations in a fix point multiplication. We first individually test the performance of large integer multiplication, squaring and reduction. Our results show that it takes $0.47ms, 0.44ms$ and $0.07ms$ to perform a 160×160 multiplication, squaring and reduction, respectively. Next, we count the the number of each operation required in a point multiplication. Since we adopt the mixed coordination (the combination of Jacobian coordinate and Affine coordinate), each point addition requires 8 large integer multiplications and 3 large integer squaring, and each point doubling requires 4 large integer multiplications and 4 large integer squaring. In addition, each multi-

plication, squaring or shifting operation has to be followed by a modular reduction. Our program shows the point addition requires 12 modular reductions, and the point doubling requires 11 modular reductions. In total, each point multiplication costs $164 \times 4 + 41 \times 8 = 984$ large integer multiplications, $164 \times 4 + 41 \times 3 = 779$ large integer squaring and $164 \times 11 + 41 \times 12 = 2,296$ large integer modular reductions. Plugging in the results of the individual tests, we get the total amount of time consumed on the three operations is 0.97s, roughly 78.2% of the total time to do a fix point multiplication. The rest of 21.8% of the time is spent on various operations, including inversion operation (to convert the Jacobian coordinate to Affine), addition, subtraction, shifting and memory copy, etc. Based on above analysis, we believe the performance of ECC operations on MICAz can be further improved by more refined and careful programming.

2.5.2 Performance Comparison

In the last part of the evaluation, we compare the performance of our implementation with existing research results [41, 59, 62] and give the possible explanation of the performance gap.

	MICAz				TelosB	
	WM-ECC	Sun-ECC	TinyECC	EccM2.0	WM-ECC	TinyECC
SIGN	1.35s	0.81s	1.92s	30s	1.55s	4.36s
VERIFY	1.96s	-	2.43s	-	2.25s	5.44s

Table 2.4: The performance comparison of our ECC implementation, WM-ECC, with other research results, including Sun-ECC [41], TinyECC [59] and EccM2.0 [62]. We use MICAz and TelosB as the two platforms.

We first compare the computation time of ECC operations. We denote our ECC implementation as WM-ECC, and compare the ECDSA signature generation and verification time with other

implementations in Table 2.4. Obviously, our WM-ECC is more computationally efficient than TinyECC and EccM2.0. On MICAz platform, TinyECC is 42% slower in signature generation than our implementation. On TelosB platform, the performance gap increases to 180%.

We also notice than Sun-ECC is more efficient than our WM-ECC. Their result, 0.81s for a random point multiplication, is about 40% faster than 1.35s of our result. We notice that the time for their 160×160 multiplication is 0.39ms, roughly 17% faster than our 0.47ms. In general, we believe their code is more polished and optimized in many aspects than our code. Furthermore, Our code is implemented in TinyOS, and mostly written with NesC (except several critical large integer operations), which could introduce more CPU cycles.

ECC+ECDSA	MICAz		TelosB	
	ROM	RAM	ROM	RAM
WM-ECC	18,604	1,180	11,438	1,176
TinyECC	13,858	1,440	12,564	1,526
EccM2.0	43k	820	-	-

Table 2.5: ECC implementation code size and data size comparison.

Since memory storage is extremely limited in sensor motes, the program code size and data size determine the feasibility of the ECC package. We compare our WM-ECC with TinyECC and EccM2.0. We do not compare Sun-ECC because it is not based on TinyOS so it is not comparable. To compare with the code size and data size of TinyECC that only has ECC and ECDSA modules, we combine ECC library and ECDSA of our WM-ECC, but not UART communication module. Note EccM2.0 only has the ECC module, there is no ECDSA available. Table 2.5 shows WM-ECC has the similar program code size and data size as TinyECC. The code and data sizes shown for

Comparatively, EccM2.0 consumes much more code space. Given 128KB ROM, 4KB RAM on MICAz, and 48KB ROM, 10KB RAM on TelosB, WM-ECC can easily fit in existing applications. One may notice that WM-ECC requires extra 5KB code size than TinyECC on MICAz platform. This is due to the trade-off of the computation efficiency. We have extensively optimized the large integer operations on MICAz platform. As the result, the code size is slightly inflated due to the techniques such as loop unrolling. Considering the programming space MICAz is relatively large, 128KB, we believe this trade-off of 5KB code size is worthwhile.

2.6 Conclusion

In this chapter, we present a number of optimization schemes to efficiently implement the public key cryptosystems in small, less-powerful sensor devices. We implement 1024-bit RSA and 160-bit ECC on various commdity sensor motes. Our experiments show that the times for ECC signature generation and verification are 1.35s and 1.96s respective for Mica motes, 1.55s and 2.25s for TelosB motes, 0.77s and 1.12s for Tmote Sky. Comparatively, RSA not only has much longer key size, which potentially introduces more memory and communication overhead, it also runs much slower. The running time of RSA private key operation on MICAz motes is 21.5s, even though its public key operation (with the 17-bit key size) only takes 0.79s. Considering that private key operations are also necessary in many sensor security schemes, we pick ECC over RSA as the cryptosystem for sensor networks.

Algorithm 1 Reduction by using long division.

1: Input: x, n;

2: Output: $r = x \bmod n$;

3: **while** $x \geq n$ **do**

4: Align the most significant byte (MSB) of modulus n to the MSB of x, the lower bytes of n can be filled with zeros;

5: Starting with the MSB of x, divide the first two MSBs of x by the MSB of modulus n, and get the quotient;

6: Multiply the quotient with the modulus and get a subproduct;

7: If the subproduct is greater than the remainder of x (over estimation), subtract the modulus from the subproduct;

8: Then subtract the subproduct from the remainder of x;

9: The procedure continues and goes back to step 2 if the MSB of the remainder becomes zero;

10: If the MSB of the remainder is not zero (under estimation), subtract the modulus from the remainder, and then go back to step 2;

11: The procedure stops when the remainder is less than modulus n;

12: **end while**

13: return x;

Algorithm 2 Extended Euclidean Algorithm

```
Input: e, φ(n)
Output: d
x ← 0, lastx ← 1
a ← e, b ← φ(n)
while b! = 0 do
    q ← a/b
    r ← a mod b
    a ← b
    b ← r
    temp ← lastx
    x ← lastx + q * x mod φ(n)
    lastx ← temp
end while
return lastx
```

Algorithm 3 An efficient exponentiation by using Montgomery reduction.

1: Input: b, a, n

2: Output: $c = b^a \pmod{n}$

3: $c \leftarrow b \cdot 2^k \pmod{n}$, $(2^k > n)$

4: $t \leftarrow c$

5: **for** from $i = msb(a)$ to $i = 0$ **do**

6: $\quad c \leftarrow c^2 \cdot 2^{-k} \pmod{n}$

7: $\quad$ **if** i's bit of a is set **then**

8: $\quad\quad c \leftarrow c * t \cdot 2^{-k} \pmod{n}$

9: $\quad$ **end if**

10: **end for**

11: $c \leftarrow c * 2^{-k} \pmod{n}$

12: return c

Algorithm 4 Calculate $a^d \pmod{n}$ by CRT

1: Input: a, d, n, p, q $(n = p * q)$

2: Output: $m = a^d \pmod{n}$

3: $R_p \leftarrow q^{p-1} \pmod{n}$, $R_q \leftarrow p^{q-1} \pmod{n}$

4: $D_p \leftarrow d \pmod{p-1}$, $D_q \leftarrow d \pmod{q-1}$

5: $A_p \leftarrow a \pmod{p}$, $A_q \leftarrow a \pmod{q}$

6: $M_p \leftarrow A_p^{D_p} \pmod{p}$, $M_q \leftarrow A_q^{D_q} \pmod{q}$

7: $S_p \leftarrow M_p * R_p \pmod{n}$, $S_q \leftarrow M_q * R_q \pmod{n}$

8: $m = S_p + S_q \pmod{n}$

9: return m

Chapter 3

Data Privacy Protection

A main challenge of large scale sensor networks is the deployment of a practical and robust security mechanism to mitigate the security risks exposed to the unattended and resource constrained sensor devices. The security challenges have attracted extensive attentions in the research community. Eschenauer and Gligor proposed a random graph based key pre-distribution scheme [28]. The scheme assigns each sensor a random subset of keys from a large key pool, and allows any two nodes to find one common key with a certain probability and use that key as their shared symmetric key. Based on their contribution, a number of researches [16, 26, 60, 15, 61] have been delivered to strengthen the security and improve the efficiency. Researchers found the sensor deployment information can be used to reduce the number of pre-loaded keys and meanwhile improve the key connectivity. Instead of pre-distributing random keys, schemes [26, 60] pre-loads either secret matrices or secret polynomials in the sensors to improve the connectivity and reduce the overhead. Recently, this method is also adopted in heterogeneous sensor networks [90, 89]. Although the symmetric-key-based schemes are efficient in computation, they require complicated key management that may cause high memory and communication overhead. This drawback has not yet

been investigated by experimental work so it is not clear how these schemes perform in a realistic system.

Recent progresses in implementation of elliptic curve cryptography (ECC) on sensors as we have presented in the previous chapter and other research work [62, 41, 59] prove public key cryptography (PKC) is now feasible for resource constrained sensors. Given the efficient low-layer primitive in place, the high-layer PKC-based security scheme design in sensor networks, however, is not straightforward due to the special hardware characteristics and requirements of sensor networks. Zhang *et al.* proposed several PKC-based pairwise key establishment schemes [112, 110] by using ID-based cryptography and achieve some nice security features. Unfortunately, it is still a doubt that the ID-based cryptography is feasible for resource constrained sensors. Therefore the performance of PKC-based security schemes is still not well investigated.

In this chapter, we compares the symmetric cryptography and PKC-based schemes through an experimental study on an important sensor network security problem: user access control. Our results suggest the PKC-based user access control scheme is more advantageous in terms of the memory usage, message complexity, and security resilience. Then, we explore effective and practical security solutions for sensor networks. Finally, we discuss other applications of our proposed public-key security infrastructure.

3.1 User Access Control

Recent advances in sensor systems have seen the introduction of sensor platforms with large storage capacities up to hundreds of megabytes [107, 63]. Novel sensor file systems [24] and storage solutions [107] have been proposed to take advantage of this large storage. Research in this area is motivated by fact that large amounts of energy is needed to send data back to a sink, which is even

more wasteful when not all the data is useful. Instead, it more efficient to store the data *within* the sensor network. When a user has a query, he queries the sensor network itself, and the query is routed to sensors that can provide the answer.

One aspect of the in-network data storage model is that each individual sensor has a larger role to play. Instead of simply collecting data and forwarding it to the sink, the sensor is now responsible for routing queries to the right sensor, authenticating the user, and controlling access to its data. In other words, many of the previous tasks performed by the powerful sink must now be performed by the relatively resource constrained sensor. It is tempting to simply implement existing access control solutions directly onto the sensor. However, due to limited power, storage and processing capabilities of the sensor hardware, this is not feasible. Furthermore, access control in sensor networks differs from regular access control in that it is not enough to simply deny unauthorized users access to the data. An unauthorized user should not be allowed to use the network since network bandwidth is very limited and, more importantly, the battery power of each node may be depleted after malicious users aggressively effuse messages to the network.

To protect the data, sensors have to authenticate the user, and control the access to their data. The user authentication and communication encryption have received extensive attentions [71, 32, 75] for security in large network system. Kerberos [71] has been widely used in distributed client-server authentication and session key establishment. Fox *et al.* proposed a lightweight version of Kerberos, Charon [32], for mobile devices. Both schemes are centralized; a central server has to be on-line to assist user's request. In sensor networks, SPINS protocol [75] shares the same security architecture. While the centralized schemes have many attractive security features, the communication overhead becomes a major issue when the network size scales, specially for the extremely energy constrained sensor nodes in a large network. For the same reason, the security

schemes [114, 79] relying on a central server are not desired in the security mechanism design in large-scale sensor networks. Access control in sensor networks also differs from the conventional schemes in that it is not enough to simply deny unauthorized users' accesses to the data. An unauthorized user should not be allowed to use the network since the network bandwidth is very limited and, more importantly, the battery power of each node may be depleted after malicious users flood messages to the network.

The aforementioned special sensor hardware and network requirements motivate us to design the user access control scheme in a very different fashion. Our basic idea is to authenticate the user locally by the sensors in the user's vicinity and transfer the endorsements of the local sensors to the remote sensor for data query. In this way, unauthorized data access request will be rejected locally so that DoS attacks trying to deplete the battery power of the network will be blocked locally. The access control proposed in this chapter is composed of several components. First, the sensors in proximity need to exchange pairwise keys for secure communications. Second, the user needs to get authenticated by the local sensors either for local sensor data access or for remote sensor data access. Third, the local sensors also need to help the user and the remote sensor build a pairwise key to achieve end-to-end security.

While existing symmetric key schemes [28, 16, 60, 15, 109] can achieve some of the security goals, several significant drawbacks such as high memory and communication overhead in key management, and security vulnerabilities, as we will show in our experimental study, make the symmetric cryptography bases solutions not desirable. The PKC-based pairwise key schemes proposed by Zhang *et al.* [110, 112] achieve some nice security features by using ID-based cryptography. Although the schemes are very novel, the ID-based cryptography is still not feasible for resource constrained sensors. We propose an ECC-based, practical and security resilient PKC-

based user access control suite. Our approach not only embraces the cryptographic primitive tweaking to achieve the computation and communication efficiency, but composes of a carefully designed and novel threshold-endorsement protocol to address the issue of denial-of-service (DoS) in remote access control. We have implemented all protocols on widely used MICAz and TelosB motes. Our performance evaluation compares the proposed access control suite with prior work which are based on symmetric-key and the prevalent RSA on Internet through comprehensive experiments and rigorous analysis.

3.2 System Model and Assumptions

We consider a large scale wireless sensor network deployed in a variety of environments. Data access to the stored data on each node is protected according to the attributes of the data. The examples include data type (temperature, light, or noise), data location, and data collection time. A user equipped with a portable computing device, such as a PDA, interacts with the sensor network for data query and retrieval. This device is more powerful than the sensor nodes, so it is capable of more computationally intensive tasks. User can query either "local" sensors through direct communication links, or "remote" sensors (that are outside of direct communication range) through multihop routing by intermediate sensor nodes.

We assume a certification authority (CA) is responsible for generating all security credentials for sensors. During the deployment, each sensor is pre-loaded its private key, public key and the corresponding certificate. The user acquires his certificate from the CA through an out-of-band security channel. The certificate includes an access control list which defines his access privilege. The user access list defines the user's access privilege. A typical access list is composed of *uid* and *user access privilege mask*. The *uid* is a unique number that identifies the user. The *privilege*

mask defines the certain level of the user access right. To query the sensor network, the user needs to attach the certificate with the query message. The contacted sensor checks the access list and verifies the user's privilege. The verification is performed in a distributed fashion without involvement of the CA. The contacted sensor grants the user the answers that are compliant with the access privilege. If the users cannot be verified, the query will be denied.

The adversary may launch either passive attacks or active attacks, or both. The passive attack includes message eavesdropping, traffic monitoring and analysis. For active attack, we mainly focus on following three types. The first is node compromise. The compromised sensor may capture the legitimate user information while being accessed and reveal it to the malicious third party. Second, user collusion can help malicious users to subvert the system and gain more access privilege. Third, the adversary may inundate user queries in the network to deplete the battery power of sensor nodes. We assume that at most $t-1$ sensors (where t is a security parameter) can be compromised and an unbounded number of users can collude since it is not hard for mischievous users to share information and orchestrate an aggregated analysis to the collected information.

In this chapter, we do not address disruption attacks. Disruptions occur when the adversary, by compromising a sensor node, drops legitimate messages or contributes a bogus endorsement share in remote access control (as we will describe later) to invalidate user remote queries. Even though disruption attacks in general are difficult to defend against in sensor networks, a smart adversary is not willing to launch such attacks because incidents of message dropping and user remote access failure may easily trigger system attentions and thus expose the compromised sensors.

3.3 Pairwise Key and Local Access Control

We start the discussion with the secure pairwise key establishment. A lot of sensing tasks (e.g., event detection and user remote access endorsement described in Section 3.4) are achieved through collaborations of multiple neighboring sensors, which require secure peer-to-peer communication to prevent the adversary from eavesdropping. For the same reason, the secure communication channel is also desired when a user queries sensors.

In this section, we design an ECC-based pairwise key establishment scheme for neighboring sensors. A common way to share a secret between two parties is to use Diffie-Hellman (DH) scheme. However, DH cannot be directly used in sensor networks due to the potential Man-In-The-Middle (MITM) attack. We thus develop our key establishment scheme based on ephemeral DH protocol over elliptic curve. We tweak the original DH protocol to defend against MITM attacks. As we will explain later, our scheme is to achieve the best communication and computation efficiency. This PKC-based pairwise key scheme can also be applied for local user access control with a slight modification. We give the brief security and cost analysis in the end.

3.3.1 Pairwise Key Establishment Between Two Sensor Nodes

We assume the system certification authority (CA) selects an elliptic curve E over the finite field $GF(p)$, where p is a large prime number. We denote P as the base point of E, where P has the order of q (q is a prime number too). CA keeps a system secret x, and publishes the system public key $Q = xP$. We will continue to use this cryptosystem setup throughout this chapter.

Similar to the conventional PKI, sensors' public keys need to be certified. Since it is not realistic to have an online CA that can verify the public key in real time, each sensor has to pre-load its certificate that is pre-computed by CA. We first discuss how to generate the private key,

the public key and the certificate for each sensor. We first define two one-way cryptographic hash functions, $h_1 : \{0,1\}^* \mapsto [0, q-1]$, $h_2 : \{0,1\}^* \mapsto \{0,1\}^l$, where l is the pairwise key length. Let us consider a sensor node u (we denote u as the sensor ID). CA first selects a random number c_u, generates its certificate $C_u = c_u P$, and calculates $e_u = h_1(u||C_u)$. The private key of u is derived as $q_u = e_u c_u + x$, and the corresponding public key is $Q_u = q_u P$. Note C_u, q_u and Q_u satisfy the following property:

$$Q_u = e_u C_u + Q. \tag{3.1}$$

The function of e_u is to bind sensor ID, u, with its certificate, C_u, so that the sensor cannot claim itself as another ID v. As we will explain later, e_u can be utilized to bind a user's access control list with her certificate.

Before the deployment, sensor u is pre-loaded with q_u, Q_u and C_u. Considering a typical 160-bit elliptic curve, these credentials require 100 bytes of memory space.

For two sensors u and v with (q_u, Q_u, C_u) and (q_v, Q_v, C_v) respectively, the ECC-based pairwise key establishment protocol is illustrated in Fig. 3.1. We denote $(.)_k^+$ as a symmetric key encryption operation by using key k, and denote $(.)_k^-$ as a symmetric key decryption operation by using key k. The symmetric key scheme can be any existing scheme, such as AES. Sensor u sends v the key establishment request, which includes u's public key Q_u, certificate C_u and nonce n_0. Sensor v calculates $e_u = h_1(u||C_u)$, and verifies u's public key by plugging e_u and C_u in Eq.(3.1). The request will be immediately dropped if the derived public key does not match Q_u. If the verification is successful, v picks a random r_v and generates the challenge in the following steps:

1. v multiplies r_v with Q_u to get a secret ECC point R_v.

2. The hash value of R_v, denoted as $h_2(R_v)$, is used to encrypt the randomly picked secret

(1) $u \rightarrow v : Q_u || C_u || n_0$

(2) v : verifies Q_u, picks random number $r_v \in [1, q-1], k_v \in \{0,1\}^l$

(3) $v : R_v = r_v Q_u,\ Y_v = r_v P,\ n_1 = h_2(n_0),\ \xi_v = (k_v || n_1)^+_{h_2(R_v)}$

(4) $v \rightarrow u : \xi_v || Y_v || Q_v || C_v$

(5) $u : k_v || n_1 = (\xi_v)^-_{h_2(q_u Y_v)}$, verify $n_1 = h_2(n_0)$

(6) u : verifies Q_v, picks random number $r_u \in [1, q-1], k_u \in \{0,1\}^l$

(7) $u : R_u = r_u Q_v,\ Y_u = r_u P,\ n_2 = h_2(n_1),\ \xi_u = (k_u || n_2)^+_{h_2(R_u)}$

(8) $u \rightarrow v : \xi_u || Y_u$

(9) $v : k_u || n_2 = (\xi_u)^-_{h_2(q_v Y_u)}$, verify $n_2 = h_2(n_1)$

(10) u, v : agree on $k_{uv} = k_u \oplus k_v$

Figure 3.1: ECC-Cert: ECC-based pairwise key establishment scheme between two neighboring sensors: u and v.

key, k_v. The hash of n_0, denoted as n_1, forms a nonce chain to defeat the potential security attacks.

3. v computes Y_v by multiplying r_v with the base point P.

Upon receiving ξ_v and Y_v, u can recover k_v because $q_u Y_v = q_u \cdot r_v P = r_v Q_u = R_v$, which is used to encrypt k_v and n_1 by v. After the decryption, u verifies n_1 and continues the execution of the protocol if n_1 is correct. Otherwise, u exits the protocol immediately.

In addition to the challenge generation, v also sends its public key Q_v and the certificate C_v to u. The same verification and challenge are performed by u. Finally, u and v agree on their pairwise key $k_{uv} = k_u \oplus k_v$.

The protocol presented in Fig. 3.1 is a general purpose scheme which provides security re-

$$u \rightarrow v : Q_u$$

$$v : \text{ selects a random } r_v$$

$$v : R_v = r_v Q_u,\ Y_v = r_v P,\ \xi_v = X(R_v) \oplus r_v$$

$$v \rightarrow u : \xi_v || Y_v$$

$$u : r_v = X(q_u Y_v) \oplus \xi_v$$

$$u, v : \text{agree on key } r_v$$

Figure 3.2: The optimized ECC-based pairwise key establishment scheme between two neighboring sensors: u and v, without checking the certificate.

silience even in an extremely adverse environment. Considering the fact that most pairwise key establishment happens in the network initialization period and, many times, this period of time can be considered active security attack free (e.g., there is no compromise and Man-In-The-Middle attack), the following two optimizations can be applied to achieve better efficiency. First, since all sensor nodes are honest at that time, the verification of public key is not required. Second, the challenge is only required on one direction when two neighboring sensors try to establish the key. As the result, step (6), (7), (8) and (9) in Fig. 3.1 are not required in the optimized scheme, and there is not necessary either to verify the public key in step (2).

The optimized pairwise key protocol is shown in Fig. 3.2. The optimized scheme requires only two ECC point multiplications compared to three in the general scheme. The further optimization is possible if the sensors have additional storage space. The idea is to select a set of random number $\{r_v\}$, pre-compute the corresponding points, $\{Y_v = r_v P\}$, and store them in the flash memory. When v receives the request, it randomly pick one entry (r_v, Y_v) and immediately sends Y_v to u, and then computes the challenge. In this way, the two ECC point multiplications, $R_v = r_v Q_u$ at

v and $R_v = q_u Y_v$ at u can be computed simultaneously. As the result, the processing overhead of pairwise key establishment reduces to only one ECC point multiplication. After the pairwise key is established, v erases the selected (r_v, Y_v) from the storage so that the same random number/point will not be used again. Note the optimized protocol is resilient to passive security attacks. The traffic analysis (if the adversary monitors all network activities) does not reveal any pairwise key secret.

From now on, we denote "ECC-Cert", "ECC-NoCert", "ECC-PreComp" as the general purpose scheme, the optimized scheme and the optimized scheme with pre-computation, respectively, throughout the rest of chapter. "ECC-Cert" also can be directly applied for one-hop user access authentication. In that case, the user, say Alice, has to give her access list al_A and certificate C_A, where al_A composes of the user id and the corresponding privilege mask. The contacted sensor builds Alice's public key based on al_A and C_A, and then perform the rest of authentication protocol.

3.3.2 Local Access Control

To achieve user access control, it is essential for the queried sensor node to verify the user's access privilege. After the access privilege verification, a secure communication channel has to be established between the sensor and the user for secure information delivery. The ECC-based pairwise key establishment scheme can be applied in the local access control with only a slight adjustment.

Our ECC based user access control protocol is presented in Fig. 3.3. The sensor u first verifies the user certificate, which includes the access list al_A and C_A, then challenges the user by using the method described in "ECC-Cert". Nonce N_A is used to prevent the message replay attack. Again, the pre-computation described previously is used to improve the efficiency. Note our local access

$$Alice \rightarrow u : al_A||C_A||n_0$$

$$u : \text{picks } r_A \in [1, q-1], \text{ and } k_A \in \{0,1\}^l, \text{ compute } Y_A = r_A P$$

$$u \rightarrow Alice : Y_A$$

$$u : e_A = H(al_A||C_A)$$

$$u : Q_A = e_A C_A + Q$$

$$u : R_A = r_A Q_A,\ n_1 = h_2(n_0),\ \xi_A = (k_A||n_1)^+_{h_2(R_A)}$$

$$Alice : R_A = q_A Y_A$$

$$u \rightarrow Alice : \xi_A, (n_1||al_A)^+_{k_A}$$

$$Alice : k_A = (\xi_A)^-_{h_2(R_A)}$$

$$Alice : \text{ verify } k_A \text{ by comparing the } n_2, n_2 = h_2(n_1)$$

$$Alice \rightarrow u : (n_2)^+_{k_A}$$

$$u : \text{verifies } n_2$$

Figure 3.3: ECC based local access control and pairwise key sharing scheme. We denote "Alice" as the user, and "u" as a local sensor.

control protocol does not require the user to authenticate the sensor, even though this authentication can be easily applied. One may be concerned that the compromised sensor may provide false information to the user. This problem is beyond the scope of user access control, and there is no way to prevent it.

3.3.3 Cost Analysis

The cost of the pairwise key establishment and local access control is determined by the communication and the computation overhead. The communication overhead can be measured by

the message complexity. "ECC-Cert" shows u has to send three elliptic curve points (Q_u,C_u and Y_u) and one scalar value (ξ_u). Given a 160-bit ECC cryptosystem, each point has the size of 40 bytes, and each scalar value has the size of 20 bytes. Therefore, u and v have to transmit 140 byte data, and the message complexity for u and v is 280 bytes. Comparatively, in "ECC-NoCert" and "ECC-PreComp", neither sensor needs to send the certificate, then the message complexity reduces to 100 bytes.

In ECC, the point multiplication is much more expensive than other operations, we approximately estimate the computational cost by counting the number of point multiplications. As shown in Fig 3.1, "ECC-Cert" requires three point multiplications. Comparatively, "ECC-NoCert" and "ECC-PreComp" only require two and one point multiplication, respectively. In the local access control, with the optimization of pre-computation, the sensor has the similar message overhead as in "ECC-Cert", but has less computation overhead because the pre-computation saves one point multiplication.

3.3.4 Security Analysis

In the security analysis, we consider the following potential threats that an adversary may employ to defeat the proposed challenge-response pairwise key establishment and the local access control protocols.

- **Impersonation**. Suppose an adversary forges an identity w and the corresponding public key Q_w and certificate C_w. Note any one can generate his public key and the certificate by using system public key Q, but no one can derive his private key q_w without the system secret x. It is computationally infeasible to compute his private key q_w without the system secret x. To get q_w from Q_w is equivalent to solve the discrete logarithm problem. Without

q_w, the adversary cannot correctly respond the challenge so that the pairwise key request or local query will be immediately rejected by a legitimate sensor. For the same reason, the adversary cannot impersonate the legitimate sensors and users even if he can capture Q_u, C_u, Q_v, C_v in step (1) and (4) in the pairwise key establishment protocol shown in Fig. 3.1.

- **Replay attack**. The since the chained nonces are used in the protocol in Fig. 3.1, any replayed message except in step (1) will be dropped immediately. The adversary cannot gain any advantage by replaying the message in step (1) because there is no way to respond the challenge without the corresponding private key.

- **Interleaving**. In the interleaving attack, the adversary selectively combines the messages information from previous or parallel sessions. Due to the challenge-response nature of the protocol, the adversary cannot impersonate or deceive the sensor in the interleaving attack. The reason is that the sensor ID or the user access list is bind with the certificate, so the private key is required to correctly respond the two-way challenge. In addition, the chained nonces allow the legitimate sensors immediately drop the messages from other sessions.

- **Reflection**. A reflection attack is that an adversary sends the identical message back to the message originator for the impersonation purpose. As we explained above, the adversary cannot correctly respond the challenge generated by a legitimate sensor, the attack attempt will fail. Further, a sensor can easily drop a reflected message once the wrong nonces are detected.

- **Forced delay**. The adversary may also block the message between two legitimate parties and resend it in a later time, which is so called the forced delay attack. Obviously, our challenge-response protocol is immured to this attack. The only effect of this attack is to

force the two parties to drop the protocol session, assuming the time-out mechanism has been employed in both parties.

- **Chosen-text attack**. In the chosen-text attack, an adversary tries the strategically arranged challenges and tries to extract the other parties private key. As indicated in our protocol, the sensor always uses a ephemeral random number, r_u and r_v, it is impractical for the adversary to compute the private key of a legitimate sensor.

3.4 Remote Access Control

Theoretically, a simple extension of the certificate-based local authentication scheme can be used in the remote query. In that case, the challenge-response messages between the user and the remote sensor are routed by a number of intermediate sensors on the routing path. This multi-hop communication pattern, however, poses new security and efficiency issues: (1) potential DoS attacks; (2) high communication overhead for the user authentication and end-to-end security. The two issues are not found in the local query and can not be addressed by the certificate-based scheme due to the following two reasons.

First, because the certificate-based access control achieves end-to-end security, any intermediate sensor has no knowledge about the challenge-response message and cannot detect the DoS attack had the adversary injected a large number of fake queries.

Second, the message overhead becomes critical in the multi-hop communication to reduce the energy consumption of intermediate sensors. The certificate-based scheme requires public key exchanges between two parities. In practice, the public key size (40 bytes) is larger than the typical message size in sensor networks (29 bytes). This overhead may force the sensor to use multiple

data packets to transmit the query that otherwise would be done by just one packet. While the certificate-based scheme achieves the user authentication and end-to-end security, it requires two rounds of communications that carry the public keys and incurs the large overhead.

Therefore, we develop a threshold endorsement scheme (inspired by Shamir's secret sharing [82]) to perform the remote access control. The basic idea is that any user has to be authenticated and endorsed by t local sensors before she can send the remote query. Not only do the t local sensors block any DoS attack attempt and transfer the trust (of the authenticated user) to the remote sensor, given the assumption that the adversary can not compromise t sensors, their endorsements also naturally serve as the pairwise key between the user and the remote sensor without any public key transmission. The three components: DoS prevention, user authentication, and message security are integrated organically in the remote access control scheme.

Our scheme is presented as follows. Again, we have an elliptic curve E over finite field $GF(p)$ and a base point P with the order of a prime q. CA maintains a secret polynomial:

$$f(y) = 1 + a_1 y + \cdots + a_t y^t, \tag{3.2}$$

where $a_i \in GF(q)$ for $1 \leq i \leq t$. Note that this secret polynomial is slightly different from the one in Shamir's secret sharing scheme in that the term a_0 is equal to 1, which implies one share of this polynomial, namely the share $(0, 1)$, is already known. As the result, only t shares of this polynomial, instead of $t+1$ shares, are enough to reconstruct the polynomial because the known share can serve as the $(t+1)^{th}$ share. Therefore, to prevent the secret polynomial from being revealed, the number of malicious sensors must not be more than $t-1$. In this work, we assume only up to $t-1$ sensors can be compromised.

Before the deployment, each sensor s_i (s_i denotes the sensor ID) is pre-loaded with a secret

share z_i, where $z_i = f(s_i)$. Any $t+1$ shares from $t+1$ sensors, without the known share, can reconstruct the secret polynomial by Lagrange interpolation:

$$f(y) = \sum_{i=1}^{t+1} z_i \prod_{j=1, j\neq i}^{t+1} \frac{s_j - y}{s_j - s_i}. \tag{3.3}$$

When $y = 0$, the $t+1$ secret shares satisfy:

$$\sum_{i=1}^{t+1} z_i l_i = 1, \tag{3.4}$$

where l_i is the Lagrange coefficient, and determined as $l_i = \prod_{j=1, j\neq i}^{t+1} \frac{s_j}{s_j - s_i}$. It is true that any t shares, $z_1, ..., z_t$, plus the known $(0,1)$ share, also satisfy the above equation with different Lagrange coefficients. However, the known share is not the interest of our remote access control scheme, and we focus on the $t+1$ shares from the sensors in the following discussion.

CA also defines a cryptographic hash function $\overline{H}$, mapping a number $\{0,1\}^*$ to a nonzero elliptic curve point on E. The remote access control protocol is given in Fig. 3.4. We denote $s_1, s_2, \cdots, s_t$ as the local sensors, s_r as the remote sensor. We assume that the ID of the remote sensor for data access is known by some scheme that is beyond the scope of this dissertation, e.g., resource discovery protocols. The user, Alice, first performs local access control protocol with t local sensors, $s_1, \cdots, s_t$. After the successful authentications, each local sensor s_i endorses Alice in the following way. First, s_i calculates $R_A = \overline{H}(al_A)$. Note R_A is a point on the elliptic curve E. Then s_i generates its endorsement: $z_i l_i R_A$, where the Lagrange coefficient $l_i = \prod_{j=1, j\neq i}^{t} \frac{s_j}{s_j - s_i} \cdot \frac{s_r}{s_r - s_i}$ (here we use s_r instead of s_{t+1}). In the next step, s_i sends the endorsement to Alice through the secure communication channel established in the local access control as described in Section 3.3. With the t endorsements collected, Alice calculates the elliptic point V_A, which is the summation of the t endorsements. Note only Alice knows the value of V_A. None of t local sensor knows V_A

$Alice \rightarrow s_1, \cdots, s_t : al_A || C_A$

for (each sensor $s_i, i = 1, 2, \cdots, t$)

s_i : perform user authentication

s_i : compute $R_A = \overline{H}(al_A)$

$s_i \rightarrow Alice : z_i l_i R_A$

$Alice$: gets $V_A = \sum_{i=1}^{t} z_i l_i R_A$

$Alice \rightarrow s_r : al_A || l_r || (al_A || query)^{+}_{h_2(V_A)}$

s_r : compute $R_A = \overline{H}(al_A), V'_A = R_A - z_r l_r R_A$

$s_r : al_A = ((al_A || query)^{+}_{h_2(V_A)})^{-}_{h_2(V'_A)}$

$s_r \rightarrow Alice : (reply)^{+}_{h_2(V_A)}$

Figure 3.4: ECC-based local threshold endorsement scheme to establish remote pairwise key between the user and the remote sensor.

because each sensor only knows its own share of V_A. Now, V_A becomes the shared secret between Alice and the remote sensor s_r. Alice encrypts her access list and query by using $h_2(V_A)$, and then sends the encrypted query along with her access list and l_r ($l_r = \prod_{j=1}^{t} \frac{s_j}{s_j - s_r}$, also calculated by Alice) to the remote sensor s_r. Upon the receipt of the remote access request from Alice, s_r first calculates $R_A = \overline{H}(al_A)$ and computes $V'_A = R_A - z_r l_r R_A$. According to (3.4), V'_A should be equivalent to V_A because:

$$\sum_{i=1}^{t} z_i l_i R_A + z_r l_r R_A = (\sum_{i=1}^{t} z_i l_i + z_r l_r) \cdot R_A = R_A. \tag{3.5}$$

Therefore, s_r can successfully decrypt al_A and *query*. Finally, s_r replies *Alice* with the query result, again encrypted by $h_2(V_A)$.

In summary, the main idea of remote access control is to design a mechanism that allows a set of local sensors (because we do not trust a single sensor) to transfer the trust (if the user is authenticated) to the remote sensor, so that the remote sensor does not need to perform the interactive user authentication employed in local authentication, which requires several rounds of communications. This endorsement scheme can be combined with existing en-route filtering schemes, such as SEF [104] and IHA [115], to further prevent the adversary from injecting the data queries through a compromised sensor.

Our scheme can also be extended to work in a sparse network, where t local sensors are difficult to find at one time. In that case, the user moves around and finds t sensors at different locations. All these t sensors perform the same location authentication as described. To produce the endorsement shares, t sensors need to communicate with each other and exchange their ID list and agree on the remote sensor s_r. Note the communications cannot be initiated by sensor themselves since multi-hop communications have to be endorsed as we described previously. For this reason, the user moves back and force, as a carrier, to distribute the node IDs to each of t sensors. Once t sensors share their IDs and agree on s_r, the rest of scheme is the same as described previously.

3.4.1 Cost Analysis

To endorse the user, each local sensor only needs to perform one ECC point multiplication and one hash function $\overline{H}$. $\overline{H}$ is a special hash function that maps $\{0,1\}^*$ onto the elliptic curve E. According the study by Boneh *et al.* [9], this special hash function can be efficiently achieved by two steps: first we hash onto a certain subset $F \subseteq \{0,1\}^*$;then we use a deterministic encoding function to map F onto E. The message complexity for the threshold local endorsement is small.

Each sensor only needs to send an elliptic curve point to the user, which has the message size of 40-bytes (for the 160-bit ECC).

3.4.2 Security Analysis

The proposed remote access control scheme is resilient to any sensor compromising attack with no more than $t-1$ compromised sensors due to the property of the threshold cryptography. Each sensor s_i has its own unique secret z_i. Any $t-1$ or less shares of secrets are not enough to recover the secret polynomial [82], and cannot be utilized to deduce the value of z_r hold by the remote sensor.

As described in the protocol, the user knows each share of endorsement: $z_i l_i R_A$, and even $z_r l_r R_A$. Combining all these shares only allows the user to establish shared secret with the remote sensor. These shares can not be used to generate the endorsement for any other access list. Suppose the user has a forged access list al'_A, and the corresponding $R'_A = \overline{H}(al'_A)$. To generate the endorsement shares $z_i R'_A$ $(1 \leq i \leq t)$, the user has to know z_i. However, it is computationally infeasible to retrieve z_i from $z_i l_i R_A$. Meanwhile, the knowledge of $z_i l_i R_A$ cannot be used to derive $z_i l_i R'_A$. The reason is that R_A, R'_A are random elliptic curve points, it is computational infeasible to derive $r_A, r'_A \in GF(q)$, so that $R_A = r_A P$ and $R'_A = r'_A P$. As the result, it is impractical to derive $z_i l_i R'_A$ from $z_i l_i R_A$. For the same reason, the user cannot reuse the acquired secret endorsement to access a different remote sensor.

Since each endorsing sensor establishes a secure communication channel with the user during the local authentication, the adversary cannot capture any share of the endorsement by eavesdropping. Therefore, only the user and the remote sensor share the secret, which is to build the secure communication channel for the remote access.

Finally, we specifically discuss the following potential attacks for the impersonation attempt. Since the first part of the protocol is the user local access control that is executed between the user and the local endorsing sensors, our analysis mainly focuses on the second part, which is between the user and remote sensor.

- **Impersonation**. Since the user has to be authenticated by a group of local sensors before he can access the remote sensor, the impersonation attack is easily defended by the local screening. When the user himself is malicious, the impersonation can be in a different form that the user forges his access list after he is authenticated by the local sensors. However, as we have explained above, the malicious user cannot decrypt the reply from the remote sensor because he does not possess the private key associated to his forged access list.

- **Replay attack**. The remote query answer replied by the remote sensor is encrypted by the secret key, the adversary cannot capture any information through the replay attack. The remote access control protocol, shown in Fig. 3.4, can be easily modified by including a nonce to allow the remote sensor to detect the reply attack.

- **Interleaving**. There is only one round communication between the user and the remote sensor. The remote sensor receives the query, and then encrypts the reply by using the constructed pairwise key. Without the pairwise key, which is jointly constructed by the local endorsing sensors, the adversary cannot decrypt the reply.

- **Reflection**. The reflection attack cannot be a threat because the protocol between the user and the remote sensor is not a challenge-response authentication. The user cannot understand the remote access request sent by himself, and neither can the remote sensor understand the reply message.

- **Forced delay**. The adversary cannot gain anything from the forced delay attack. As we explained, the reply message from the remote sensor is encrypted.

- **Chosen-text attack**. Our protocol can be easily improved to defeat the chosen-text attack by including a random number, e.g., nonce, in the remote query access request from the user and the reply message from the remote sensor.

3.5 System Implementation

We implement our ECC-based user access control on MICAz motes [48], the most recent MICA family motes from UC Berkeley. MICAz is powered by a ATmega128 micro-controller, which features an 8MHz, 8-bit RISC CPU, 128K bytes flash memory (ROM) and 4K RAM. The RF transceiver on MICAz is IEEE 802.15.4/ZigBee compliant, and can achieve maximum 250kbps data rate. The MICAz runs TinyOS [88] version 1.1.15.

3.5.1 User Module and Other Components

Our user module is composed of two parts. We choose an HP iPAQ pocket PC as the user computing module to perform all backend computations. The HP iPAQ features a 522MHz ARM920T PXA270 processor, 64MB RAM and 128MB flash memory. The HP iPAQ is powered by Microsoft Windows Mobile 5.0. Since the iPAQ wireless communication module is not compatible with IEEE 802.15.4/ZigBee on MICAz, we use a MICAz sensor mote to bridge the communication between the user and the sensor motes. The MICAz mote is responsible for communications with the sensor motes in the network. All the data processing is performed at the iPAQ. The two parts communicate through a USB cable. The MICAz mote acquires the USB port through the

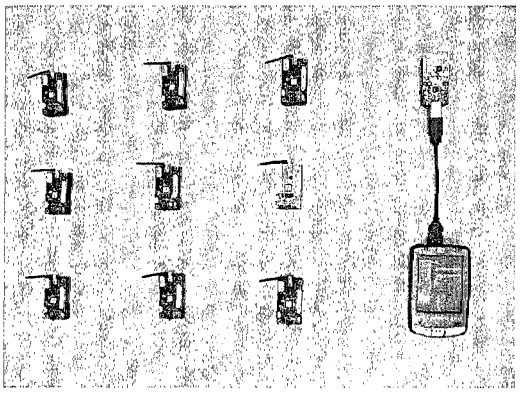


Figure 3.5: An example of user access control experiment setup including local sensors and the user module. An HP iPAQ is used as the interface for user to interact with the network (inject query and get back the data reply) through the sensor attached to the iPAQ.

MIB520 programming board [48]. However, the iPAQ does not have the USB hoster. We solve this problem by mounting a USB Hoster Compact Flash Card on the iPAQ. To utilize the Serial Forwarder facility from TinyOS 1.1.15 to regulate the communication between the MICAz and the iPAQ, we install Mysaifu Java Virtual Machine [69] on the iPAQ. Fig. 3.5 shows the testbed and the user module setup for our experiments. We implement the same ECC primitive on the iPAQ. Given the powerful processor and plenty of memory, the ECC performance on iPAQ was expected to be much faster. To our surprise, initial test showed the ECC point multiplication still costs 200ms, only 6 times faster than MICAz with a more than 60 times faster CPU. The further investigation reveals that C compiler for the mobile devices has poor optimization capability, so that the multi-precision integer operation is not optimized. Therefore, we again re-write the critical components in ARM assembly language. The judicious decision improves the performance from 200ms down to 40ms. We summarize the ECC performance results on both platforms in Table 3.1.

For the hash function, we adopt SHA-1 160-bit implementation from a standard crypto library. For MAC (Message Authentication Code) module, we adopt the RC5 block cipher from

Platform	PrivKey	PubKey	PM	Sign	Verify
MICAz	20 bytes	40 bytes	1.24s	1.35s	1.96s
iPAQ	20 bytes	40 bytes	40ms	48ms	68ms
TelosB	20 bytes	40 bytes	1.44s	1.60s	2.26s

Table 3.1: The comparison of ECC execution Time on various platforms, including MICAz, HP iPAQ and TelosB, for ECC point multiplication (PM), signature generation (Sign), signature verification (Verify) time.

TinySec [54]. Both hash and MAC modules are computationally efficient. It takes several miliseconds to do a hash operation. The RC5 encryption and decryption take less than 1 ms.

3.5.2 Other Sensor Platform

Our ECC based access control schemes are not only practical for MICAz motes, but can be deployed on other sensor platform. We have successfully ported our whole software suite to TelosB motes, the latest research oriented motes developed by UC Berkeley. TelosB is powered by MSP430 micro-controller. MSP430 incorporates an 8 MHz, 16-bit RISC CPU, 48 KB flash memory (ROM) and 10K RAM. The RF transceiver on TelosB is IEEE 802.15.4/ZigBee compliant, the same as on MICAz. Therefore, TelosB and MICAz motes can be mixed together to form a heterogeneous sensor network.

Since TelosB mote has a different hardware architecture, all hardware dependent security primitives have to be re-written for TelosB. We adopt the same optimization techniques explained previously, and find ECC is also practical for TelosB motes. The ECC performance on TelosB is shown in Table 3.1. Overall, ECC operation on TelosB is only slightly slower than that on MICAz.

3.6 Analysis and Evaluation

We evaluate our access control schemes using a combination of theoretical analysis and actual implementation on a sensor platform. The symmetric key schemes compared are: Random-key [28], PIKE [15], Blom [26], and Blundo [109]. Random-key, PIKE and Blom are used to compare the performance of our public key solution when performing pairwise key establishment. Note that of the different symmetric schemes considered, only Blundo is explicitly designed for access control. Thus, we only compare our local access control solution with Blundo.

The metrics used to compare pairwise key establishment are **memory overhead**, **message complexity** and **security resilience**. Since all symmetric-key-based key establishing schemes require key pre-distribution, the memory overhead measures the amount of memory space required for each sensor to achieve a certain degree of key connectivity with its neighboring nodes. The more keys pre-distributed, the higher key connectivity can be achieved. The message complexity measures the amount of communications required for a certain sensor node to establish pairwise keys with its neighboring sensors. In security resilience against the node compromise, we measure the fraction of the compromised communication links as a result of sensor compromise. The communication links here are the direct communication links between any two neighboring sensors.

We implement Random-key scheme and Blundo user access control scheme as the real world comparison. We use the following four metrics: **key establishing time**, **memory overhead**, **message complexity** and **energy consumption**. The key establishing time measures the time duration for a random sensor to establish secret pairwise key with its neighbors. Similarly, the memory overhead measures the exact amount of data space required (in the real implementation) in the access control. The message complexity then shows the amount of messages transmitted during

the key establishing procedure. The energy consumption estimates the average communication energy consumed during the key establishment.

Finally, we implement all components in the proposed remote access control. By focusing on the processing delay, we demonstrate the delay is small, which makes our scheme practical in the real world.

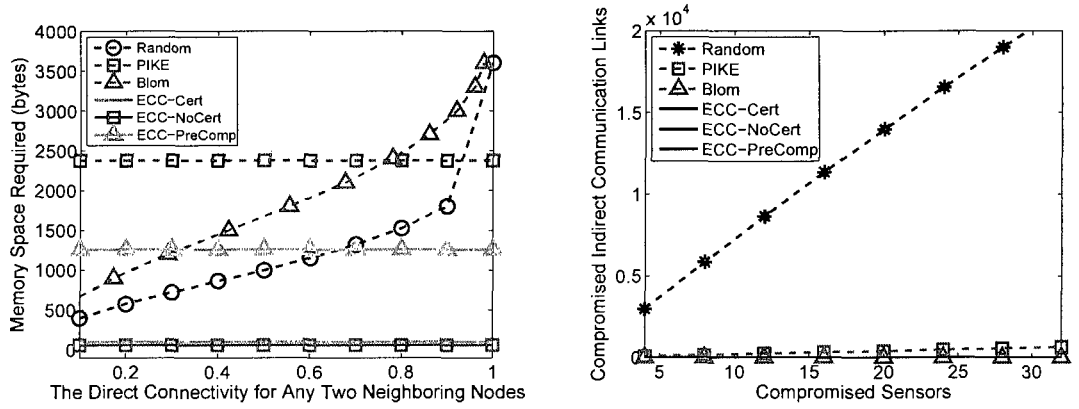
3.6.1 Analytical Results

3.6.1.1 Pairwise Key

Random-key [28] can be considered as a base line pairwise key establishment protocol. Each sensor is randomly pre-distributed with a number of secret keys from a system key pool. Any two neighboring sensors try to find a common key to establish a pairwise key by exchanging the key indices. Blom [26] is a variation of Random-key scheme. Instead of pre-distributing random keys, Blom pre-distributes secret vectors from the key spaces (or matrices) maintained by the system. Any two sensors having the vectors from a common key space can establish the pairwise key. PIKE [15] (we only consider PIKE-2D in this work) is different from the above two schemes in that each sensor, identified by a two-dimensional ID, is pre-distributed at least one common secret key with determined $2 \times \sqrt{N}$ sensors (where N is the number of sensors in the network), which have either the same row-ID or column-ID. Any two sensors establish the pairwise key through the sensor that has the same row-ID with one of the two sensors and the same column-ID with the other.

We provide three variations of our ECC-based pairwise key schemes: ECC-Cert, ECC-NoCert, and ECC-PreComp, which were discussed in Section 3.3.

Our analysis is based on a randomly, uniformly deployed sensor network with 10,000 nodes.



(a) (b)

Figure 3.6: (a). The trend of percentage of total communication links compromised with the increasing number of sensors compromised. (b). The memory space required for any two nodes to to establish a direct pair-wise key under different key connectivity rate.

On average, each sensor has 20 neighbors. The above parameters are selected according to [8] so that the sensor network can be connected with a probability greater than 99%. The senor node IDs have the size of 2 bytes. The random keys have the size of 10 bytes. With additional 2 bytes for key indices, each pre-distributed random key requires 12 bytes for memory space. We assume the key pool size is 10,000 for both Random-key and PIKE. We choose 160-bit ECC as our public key primitive. Accordingly, an ECC certificate has 40 bytes, an ECC public key has 40 bytes, and an ECC private key has 20 bytes.

The ability to establish a direct pairwise key (not through the third party) between two neighboring sensors is very important, since direct key sharing not only reduces the communication overhead, more importantly, also improves the security resilience. Fig. 3.6(a) shows the memory overhead required by the key establishing schemes to achieve a direct key between two sensors with different probability.

To increase the probability of establishing direct pairwise keys, Random-key scheme needs to pre-distribute more keys in each sensor node. We can see from Fig. 3.6(a), the memory overhead

is increasing linearly when the required key connectivity increases from 0.1 to 0.9. This trend becomes exponential when the connectivity is larger than 0.9. To achieve 100% connectivity, each sensor has to be pre-loaded with 300 keys, which requires 3.6KB memory space. Considering MICAz only has 4KB data space, the 300 keys almost use up all available memory and leave almost no space for the application programs. Thus, the Random-key scheme obviously is not practical to achieve 100% direct key connectivity.

Compared with Random-key scheme, the memory overhead of PIKE only depends on the network size. Given 10,000 sensor nodes, each sensor has to be pre-loaded with $2 \times (\sqrt{10000} - 1) = 198$ keys. Therefore, the memory overhead for PIKE is constantly $12 \times 198 = 2,376$ bytes. Blom scheme with $\lambda = 29$ and $\omega = 50$ (please refer [26] for the details) also introduces high memory overhead as shown in Fig. 3.6(a), specially when the high key connectivity rate is required.

Compared to symmetric key schemes, our ECC-based schemes overall have less memory overhead, specially when the key connectivity is high. In ECC-NoCert, each sensor only needs to store its private key and public key pair, which have the combined size of 60 bytes. In ECC-Cert, each sensor has to store one more certificate, so the memory overhead becomes 100 bytes, 40 bytes more than that of ECC-NoCert. ECC-PreComp has more memory overhead because each sensor needs to store the pre-computed random numbers (20 bytes each) and corresponding elliptic curve points (40 bytes each). Given average 20 neighbors, each sensor at least stores 20 pre-computed values, which account for 1200 bytes more overhead. As the result, the memory overhead for ECC-PreComp are 1260 bytes. Note the memory overhead of the public key base schemes do not change for achieving different key connectivity.

When the sensors are captured and compromised, the relative communication links are also compromised. These compromised links include the direct communication links connected to

the compromised nodes and indirect communication links due to the leakage of the system secret, such as the subset of system key pool in Random-key scheme. To simplify the analysis, we assume the the compromised nodes are evenly and randomly scattered in the network. Fig. 3.6(b) plots the number of compromised indirect communication links due to the node compromise.

Our ECC-based public key scheme is ideal under such situation. There is no indirect link compromised due to the node compromise. In PIKE scheme, each sensor serves the intermediary for other two sensors. Suppose a sensor with ID (i, j) is compromised. As the result, all potential links between any sensor on the ith row and any sensor on the jth column will be compromised. Given $\sqrt{N}$ sensors on the ith row and $\sqrt{N}$ sensors on the jth column, there are totally N potential links. Considering the network connectivity is $20/N$, on average 20 indirect links can be compromised for each compromised sensor. In Random-key scheme, the communication links, which are not directly connected to the compromised nodes, may also be compromised because the secret keys used in these indirect links might be revealed to the adversary when the nodes are captured. Let the number of captured nodes be x. Given system parameters: total key pool number P and pre-distributed key number k, the expected fraction of the compromised communication links is $1-(1-\frac{k}{P})^x$ [26]. Fig. 3.6(b) indicates that Random-key scheme is more vulnerable to the node compromise attack. When 32 nodes are compromised, more than 20,000 links can be compromised. PIKE scheme performs much better, but the number of compromised indirect links still grows linearly as the number of compromised sensor increases. As we indicated above, our ECC-based schemes are resistant to the node compromise. There is no indirect link can be compromised due to the node capture.

We find Blom scheme is resistant to the node compromise as no indirected link is compromised. As indicated in [26], however, this feature does not hold when the number of compromised

nodes keeps growing. The security resilience degrades exponentially when the fraction of the compromised node reaches the certain threshold.

Careful readers may argue that the network parameter selection have an impact on the results of the above memory overhead and security resiliency analysis. For example, the memory overhead of some symmetric-key schemes, such as Random-key and Blom, are low when the key connectivity is low. However, the random graph theory [8] tells that, to have a securely connected sensor networks, the key connectivity has to maintain a certain level. In our example, given 10,000 sensors and 20 neighbors for each, the key connectivity must be greater than 90% in order to have a securely connected network with the probability of 99%. It is true that the requirement for the key connectivity reduces to 50% if the average degree of each node increases from 20 to 36. However, the sensor network in the latter case is almost twice denser than the former one. As the result, the hardware cost is nearly doubled because 16 more sensors are needed in each neighborhood area. We have not found a good way to convert the hardware overhead to the memory overhead and make the comparison against our previous memory overhead analysis, but we believe the hardware cost is an important performance metric and cannot be ignored. In this chapter, we use the memory overhead as an example to present the extra cost incurred in the symmetric key schemes.

3.6.1.2 Local User Access Control

User access control requires the sensor nodes to authenticate the user and verify the user's access privilege. A symmetric-key user access control based on Blundo's scheme is proposed by Zhang *et al.* [109]. The Blundo's scheme is very similar to Blom's scheme as we explained previously. The system maintains a symmetric bi-variate polynomial. Each sensor or user is pre-loaded with a secret share of the polynomial. Any sensor and the user can establish a pairwise key by plugging

other's public information, such as sensor ID or user access list, into the secret polynomial share. The access control can be achieved by integrating the user access list to the polynomial share, so that the user has to show the genuine access list, otherwise the user can not establish the pairwise key with the sensor and can not pass the authentication. The drawback of this scheme is that it has very limited security resilience against the user collusion attack. The reason is that the system secret, polynomial shares, has to be given to the user. Multiple malicious users may easily gather the information, reconstruct the secret polynomial, and finally compromise the system security. Here we want to emphasize that only public key scheme can fundamentally solve the security hole of the user collusion attack.

We do not compare our ECC-based local user access control to Blundo access control [109]. We instead perform comparison experiment to study other advantages of our ECC-based local access control. We reserve this part to the next subsection.

3.6.2 Experimental Results

Here, we demonstrate the advantages of our proposed public key schemes through real world experiments. For the comparison purpose, we also implement Random-key scheme and Blundo's scheme based access control scheme on real sensor motes.

3.6.2.1 Experiment Test-bed and Parameter Setting

We implement the baseline symmetric key scheme, Random-key, on the same test-bed for the comparison of pairwise key establishment. We use 10 MICAz motes to form a sensor neighborhood. Each sensor can directly communicate with any of other nine neighbors. We select the key pool size of 10,000. Each key, with the size of 10 bytes, is identified by a two-byte key

index. We first generate 10,000 random keys at a laptop computer. Each mote is randomly pre-distributed with 150 keys. In the experiment, we have adopted the simple scheduling method to avoid message collision, which emulates the optimal communication environment for key establishing. We randomly pick one out of ten motes to initiate the pairwise key establishment with all its neighbors. Even with 150 keys pre-loaded, they are not enough for any mote to establish direct pairwise key with all the neighbors. Therefore, multiple rounds of key establishment have to be performed. After the first round direct key establishment, the initiating mote notifies the neighbors that have already established direct pairwise keys with it and starts the second round of key establishment. The key establishing protocol is exactly the same in the second round except the initiator has changed. Each of the neighbors that have established the direct key is required to perform the indirect key establishing in the second round. Two rounds of key establishing still may not achieve 100% key connectivity for the original initiator. More rounds of such operation could be necessary. In our experiment, however, we limit it to three rounds. That means any two neighboring motes at most have two helpers for establishing indirect pairwise key. This arrangement is supported by the fact indicated in Random-key [28] that the number of pairwise key established through more than 3 hops is negligible.

Finally, we implement the Blundo access control on our test-bed. We first generate a random symmetric polynomial. The coefficients have the size of 10 bytes. The polynomial degree is adjustable for the target security resilience against the node compromise. Each mote is pre-distributed with a secret polynomial share, which is generated by simply plugging in the mote ID. The amount of memory space for storing the polynomial share is determined by the polynomial degree. Similar as the access control test-bed implemented by our ECC public scheme, we use the HP iPAQ as the user module. Again, the iPAQ is attached to a MICAz mote.

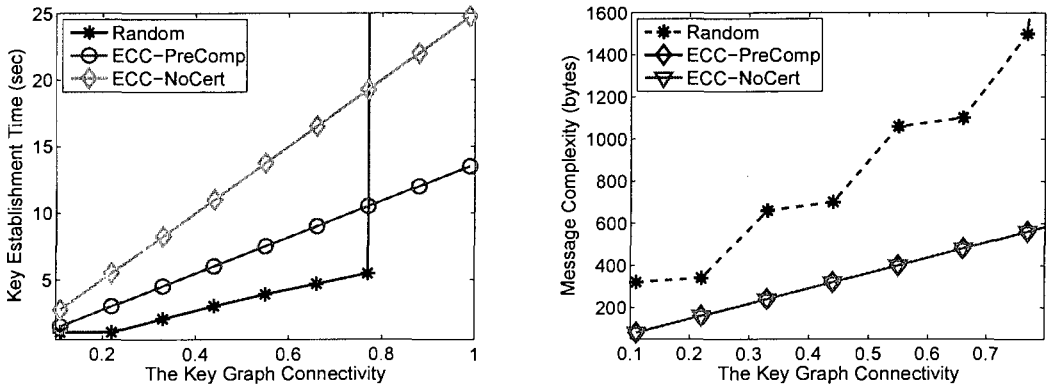
For all schemes conducted on our test-bed, we repeat the tests for 20 times, and record the average values.

3.6.2.2 Pairwise Key Establishment

Fig. 3.7(a) illustrates the processing time delay in pairwise key establishing for achieving different degree of key connection. We select two ECC-based schemes: ECC-PreComp and ECC-NoCert for this experiment. It clearly shows that ECC-PreComp is much faster than ECC-NoCert since the former scheme only requires one ECC multiplication for both neighboring sensors, while the latter one requires two. In reality, ECC-PreComp is very practical because the pre-computation only introduces a very limited memory overhead compared to that in symmetric key schemes.

Compared to the PKC-based schemes, Random-key scheme has lower processing overhead when the requirement of key connectivity is low. However, this advantage does not hold if more than 80% key connection is required. The reason is that the number of pre-distributed keys is not enough for establishing pairwise keys with all its neighbors. The key establishing time thus increases to infinity. As shown in Fig. 3.7(a), the time jumps to infinite large at key connectivity of 0.8. We restrict the pre-distributed key number due to the limited 4KB memory space in MICAz motes. In our experiment, with 150 key pre-distributed, a mote can only establish direct pairwise key with two out of its nine neighbors. The other pairwise keys are established through the second and the third rounds of key establishing procedure.

Fig. 3.7(b) further reveals that ECC-based pairwise key scheme has much less message complexity than Random-key scheme. To establish a pairwise key, two neighboring motes only need to transmit 120 bytes for both ECC-NoCert and ECC-PreComp schemes. In Random-key scheme, the broadcasting node has to send all key IDs in its key ring. Given 150 keys and 2 bytes each for



(a) (b)

Figure 3.7: (a) Key establishing delay for different key graph connectivities. (b) The message complexity for achieving the target key connectivity.

key index, the broadcasting mote transmits 300 byte message. All listening neighbors also need to respond the key establishing broadcast, by either replying the challenging message (if there is shared key), or notifying there is no shared key. This message overhead has to be paid in all three key establishing rounds. In wireless sensor networks, high message complexity increases the chance of message collision and thus causes network congestion. The low message complexity is a significant advantage for ECC-based pairwise key establishing schemes.

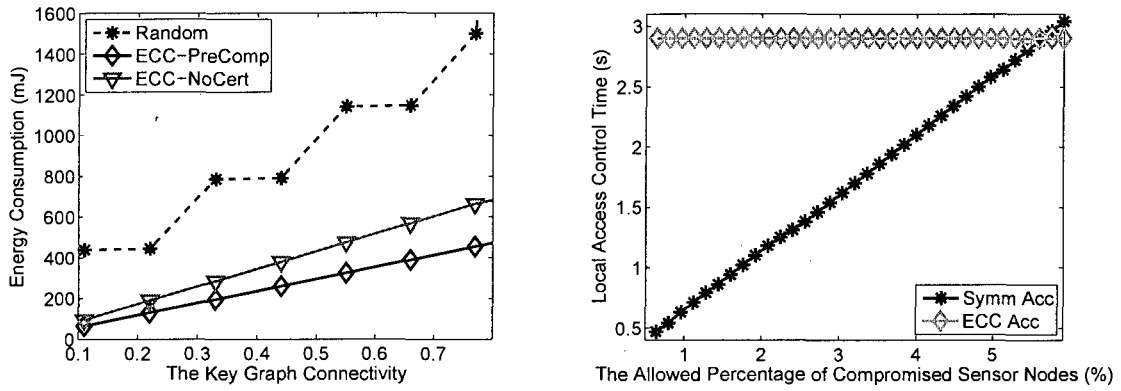
Finally, we compare the energy consumption, including communication energy and computation energy, during the pairwise key establishment. We estimate the communication energy consumption by multiplying the total amount of communications by an average communication energy consumption of $18\mu J$/bit [13]. Since the symmetric-key encryption and decryption are very efficient, we ignore the computation overhead of Random-key scheme. Comparatively, it takes several seconds in the public-key-based schemes, so the computation energy consumption cannot be ignored. The ECC computation energy consumption E can be calculated by $E = U \cdot I \cdot t$, where U is the voltage, I is the current and t is the time duration. According to the MICAz data-

sheet, U is 3.0V (two AA batteries), and I is 8mA (the current draw in the active mode). We plot the results in Fig. 3.8(a). The dash-line is the communication energy cost of Random-key scheme. The two solid lines are the combined communication energy and computation energy consummation of two ECC-based schemes. The figure clearly identifies the key drawback of Random-key scheme. The symmetric-key-based scheme consumes more than twice amounts of communication energy than the ECC-based scheme even though the public key scheme consumes more energy in computation. The reason is that message broadcast is required in Random-key scheme. As the result, all neighboring sensors need to listen the broadcasts all the time and consume the energy for receiving the messages.

We argue that processing time is not a significant metric for sensor network application as opposed to the memory usage, message complexity, robustness to sensor compromising. Pairwise key sharing is usually conducted in the initialization phase of sensor network deployment as papers on symmetric key pairwise key sharing argue. Public key protocol for key sharing takes a few more seconds to finish than symmetric key protocols, which is tolerable for network initialization and also for online pairwise key sharing, since it is done only once between two sensors. From the experimental data, we clearly see that our protocol performs better than symmetric key protocols in terms of memory usage, message complexity (and thus equivalently energy consumption and system lifetime), and robustness to network compromising.

3.6.2.3 Local Access Control

We first measure the authentication delay in the local access control. Since the parameter selection in the Blundo's scheme based local access control depends on the security resilience against the node compromise. We test both schemes under various security resilience requirements. The

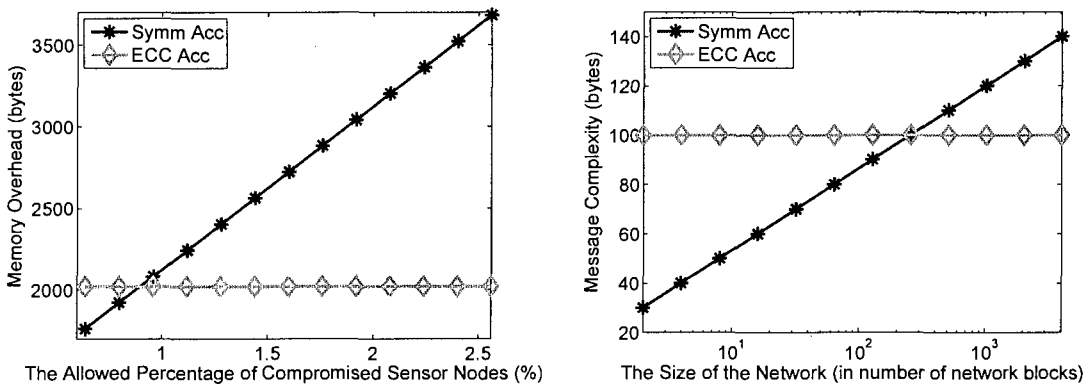


(a) (b)

Figure 3.8: (a). The energy consumption (including communication and computation) for achieving the target key connectivity. (b). Local authentication time vs. security resilience (% of compromised sensors).

user authentication delay is shown in Fig. 3.8(b). When the security resilience is low, up to 5.5% percent of sensor nodes allowed to be compromised, the Blundo access control is more efficient than our ECC-based scheme. The reason is that the polynomial operations are much faster than ECC exponentiation. However, the processing overhead of the Blundo based scheme increases as the requirement of security resilient increases. When the requirement of security resilience is more than 5.5%, the processing overhead of the symmetric-key-based scheme becomes slower than our PKC-based scheme. The reason is that the processing overhead of our ECC-based scheme does not change, it always provides the security equivalent to the discrete logarithm problem.

Note, the security concern of user collusion attack has not been revealed yet by this experiment. This security issue has to be considered in real world deployment. Therefore, either higher degree random polynomial or multiple polynomial have to be selected to improve the security. As a result, the processing overhead of the Blundo based access control will be higher. On the contrary, the ECC-based access control scheme does not suffer from user collusion attack, so our scheme can be directly applied to the real world deployment.



(a) (b)

Figure 3.9: (a). The memory space required to finish the local user authentication, regarding the different security resilience, in term of the percentage of sensors allowed to be compromised. (b). The message complexity in user authentication, regarding the different network size, in term of the number of network blocks, where each block is the user access area unit.

Fig. 3.9(a) shows the comparison of data size of two local access control schemes in the real implementation. It clearly shows that the memory overhead scales linearly in the Blundo based scheme for satisfying different security resilience. The degree of the random polynomial is larger for higher security requirements. As a result, the sensors need more space to store the corresponding coefficients. The data size of the ECC-based scheme, as can be easily predicted, does not change at all.

When Fig. 3.8(b) and Fig. 3.9(a) show that the Blundo access control has poor security scalability in processing time and memory overhead, Fig. 3.9(b) displays that it also has poor network scalability in message complexity. Since the Blundo access control scheme uses "Cell Merging" and "Block Compression" [109] to reduce the number of polynomial possessed by the user. The user has to traverse a Merkle-hash tree. The traversal path length is determined by the tree size, which is in turn determined by the number of location blocks, or the network size. Again, our ECC-based user access control has the advantage of excellent network scalability; the message

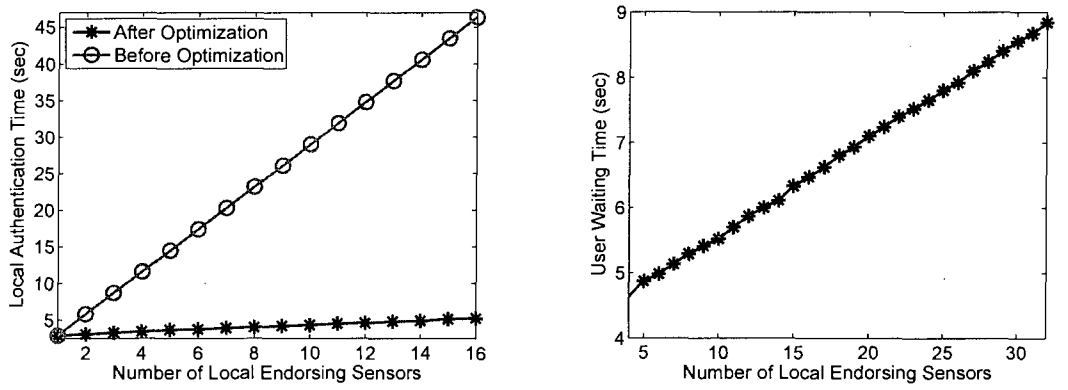
complexity is independent to the network size, fixed at 100 bytes. The figure clearly shows that the Blundo based scheme has more complexity than our public key scheme when the network size is just over 100 blocks. This fact proves our scheme is more favorable for large network deployment.

3.6.3 Remote Access Control

In this subsection, we evaluate our remote access control scheme. We first provide the micro-benchmark for the local authenticate and threshold endorsement generation. Then, we provide the overall estimation of the remote access performance. In the experiments, we mainly focus on the user perceived remote access processing delay. Our first hand experimental results suggest the PKC-based remote access control scheme is very practical.

3.6.3.1 Local Endorsement

The local endorsement procedure can be further divided into user local authentication and endorsement generation. We have already demonstrated the performance of user local authentication in the previous section. To be authenticated by multiple local sensors, a simple and effective optimization can be applied to allow the user to be authenticated in parallel rather than one-by-one. The user first sends its certificate to all the endorsing sensors, so that the endorsing sensor can verify the certificate and generate the challenges simultaneously. Then the user collects all the challenges from each member of endorsing group and responds them one-by-one. This optimization is valid for user authentication because the user device is much more powerful than sensors. As we showed previously, ECC multiplication on iPAQ is more than 30 times faster than MICAz mote. Therefore, the ECC operation overhead on user device is negligible compared to that of



(a) (b)

Figure 3.10: (a). The user local authentication time, before and after the optimization, by multiple local endorsing sensors. (b). Key establishing time with the remote sensor when the number of endorsing sensor changes from 4 to 32.

sensors. This also explains why such optimization does not work in pairwise key establishment between one sensor and its neighbors.

Fig. 3.10(a) displays time consumption when the user is authenticated by multiple endorsing sensors. For the comparison purpose, we also show the authentication delay without the optimization. Obviously, the optimized scheme is significantly more efficient. Before optimization, it takes more than 45s for the user to finish authentication with 16 endorsing sensors. This delay dramatically reduces to 5s after the optimization.

After finishing the user authentication, the endorsing sensors perform threshold endorsement to establish pairwise key between the user and the remote sensor. We continue the above authentication experiment. Each endorsing sensor immediately computes its endorsement share and then sends to the user sequentially. Fig. 3.10(b) shows the user waiting time to receive all the endorsement shares. With the number of endorsing sensors changing from 4 to 32, the time duration linearly grows from 4.5s to 8.9s. The measurement includes the user local authentication time (local pairwise key establishing time). The performance is consistent with that displayed in

Fig. 3.10(a). The threshold endorsement requires each sensor to perform one more ECC point multiplication at the cost around 1.3s as showed previously.

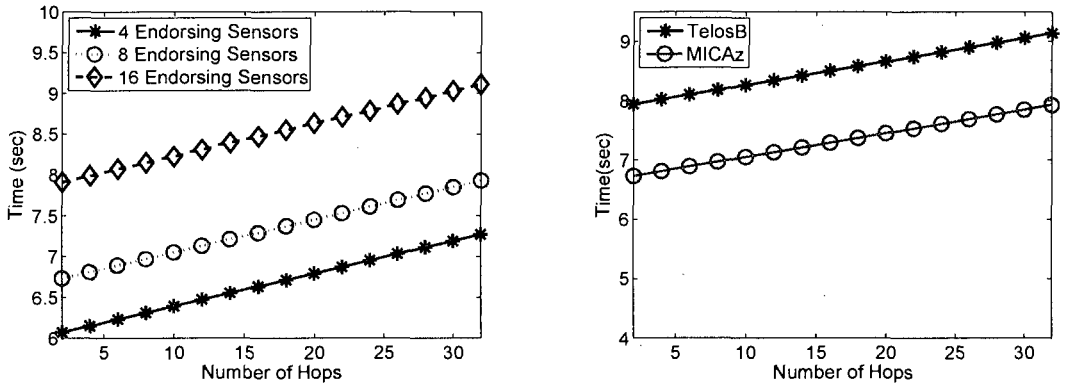
Upon the receipt of the remote access query, the remote sensor has to verify the authenticity of the remote query by decrypting the message using its own secret share as presented in section 3.4. The computational complexity of this operation is independent to the number of local endorsing sensors. The only expensive operation at the remote sensor is one ECC point multiplication. It takes only 1.4s for the remote sensor to calculate its secret share and verify the query.

3.6.3.2 Complete Remote Access Control

Finally, we are eager to investigate the overall performance of the remote access control, including the threshold signature generation, message propagation, and remote sensor verification. We assume the local endorsing sensors have already established pairwise key with each others. To simplify the experiment, the user directly sends the query to the remote sensor. Then we add the estimated hop-by-hop forwarding delay to estimate the performance for various hop distances. The estimated forwarding delay is the communication delay in sensor RF transceiver. Our estimation fixes the amount of communication delay to 17.5ms. [1]

Fig. 3.11(a) shows the estimated overall user remote query response time, given the size of local endorsing group with 4, 8 and 16, respectively. We find the overall remote query delay is short. When the remote sensor is located at 20 hops away, the user query response time is 6.8s. When the larger size of the local endorsing sensor group is required, the additional overhead increases moderately.

[1] Based on our experimental result of forwarding a 60 byte payload in MICAz motes.



(a) (b)

Figure 3.11: (a). Remote query time delay. (b). Comparison of remote query delay between MICAz and TelosB.

3.6.3.3 Porting to Other Sensor Platforms

Finally, we demonstrate that our ECC-based user access control suite can also be efficiently deployed on a different sensor platform, TelosB mote. Fig. 3.11(b) illustrates the performance comparison between the two platforms for remote access control with the setup of 8 local endorsing sensors. The overall access control performance on the two platforms is very close, although the performance on TelosB is slightly worse because ECC on TelosB is slightly slower. In practice, MICAz and TelosB can be deployed together to form a heterogeneous sensor network for user access control purpose because they share the same RF transceiver.

Chapter 4

False Data Filtering

The repertoire of sensor network applications requires an inclement and human unattended environment, such as battlefield surveillance, wild animal habitat monitoring, and environmental monitoring. Given the extremely constrained hardware resources of the sensor nodes, the adversary Denial-of-Service (DoS) attack becomes a serious security threat. For example, in the applications of the remote access control as we have discussed in the previous sections, intermediate forwarding sensors have no knowledge about the message payload (because the content is encrypted) so that they cannot determine whether or not the forwarded packets are legitimate. The adversary can first compromise an individual low-power sensor, and then inundate the whole network by injecting large amounts of bogus data packets into the network through the compromised node. These bogus messages flood the network, deplete the battery power of the sensor nodes, and finally paralyze the whole network.

This problem has attracted many attentions in the past several years. Most of prior work [104, 115, 101, 106], except [110], on sensor network message authentication and bogus data filtering mainly rely on symmetric key schemes. Ye *et al.* [104, 101] proposed a statistical en-route

false report filtering scheme (SEF). The scheme requires each report be endorsed by multiple sensor nodes by encrypting the report with their random pre-distributed symmetric keys. The intermediate nodes on the route compare their own keys with those used for encrypting the report, and check the corresponding encryption if matched keys are found. Since the authentication capability of the intermediate nodes depends on the probabilistic key sharing, only a portion of bogus messages can be detected and dropped. If the communication is between two remote sensor nodes, the receiver still cannot know, with a certain probability, whether or not the message is valid. Zhu *et al.* [115] proposed an Interleaved Hop-by-hop Authentication scheme (IHA) to detect the false report. The protocol requires that the sensor nodes maintain a pre-route interleaved associations so that any sensor shares each secret with its upper associated node and lower associated sensor. The problem of this approach is that it is not practical for large sensor networks. Many times, the message routing paths are not determined due to the unpredictable nature of wireless communications. The association requires global knowledge of the networks, which is very difficult to get for large scale sensor networks. Further, this scheme only filters the false report which is sent to the sink. The sensor nodes have no ability to authenticate the messages between the sensor nodes since the corresponding association knowledge is not available.

Unlike the symmetric key based schemes, the public key approach [110] proposed by Zhang *et al.* yields better security resilience. Unfortunately, the bilinear pairing based scheme is too expensive to be afforded by the low-power sensor hardware. Another straightforward public-key based approach is to use the public-key infrastructure (PKI) that is widely used on Internet, e.g., X509. However, PKI cannot be directly used on sensor networks due to following three issues. First, public key size is normally large, such as 128 bytes for 1024-bit RSA. Sensors are extremely resource constrained devices. The distribution of public keys in sensor network would cause high

communication overhead, which in turn will reduce the battery life. Second, the public key has to be certified before it can be used to verify a signature. It is difficult to have an on-line CA in sensor networks. The workaround solution is to attach a certificate with the public key. But again it would cause more communication overhead since the certificate has the same data length as the public key. Third, the simple scheme is not resilient to defend against DoS attacks. If a sensor is compromised, the adversary then uses it to send a large number of messages with legitimate signatures (of the compromised sensor).

In this chapter, we propose a Public-key based false Data Filtering scheme (PDF), which leverages threshold cryptography and Elliptic Curve Cryptography (ECC). As we will show, ECC is more affordable than other public key schemes for sensors. With carefully devised ECC-based security protocols and optimized ECC primitive implementation on sensor nodes, ECC is very practical on extremely resource constrained devices. In PDF, any event report message requires an attached digital signature which is signed by system private key. Due to the threat of node compromise, any single sensor cannot be trusted to keep the system private key and be allowed to generate the system signature. Instead, with the assumption that the adversary can not compromise up to t sensors, we design a threshold endorsement scheme. We first pre-distribute a unique system secret share to every individual sensor during the network deployment. Upon the detection of an event, the group of sensor nodes that detect the event collaborate together and jointly generate a system signature. The intermediate sensor nodes can easily validate the event report by efficiently verifying the attached signature. Unlike the symmetric key based schemes that only support false data filtering for the sink bounded messages, PDF supports any point to point communication in the sensor network.

Since it is computationally infeasible for the adversary to forge a digital signature without

knowing the system secret, any false report will be detected with 100% probability. PDF is also resilient to sensor compromising attack. The threshold cryptography guarantees the system secret will not be revealed as long as no more than $t-1$ (t is a system parameter) sensors are compromised. We have implemented all the components for the false data filtering scheme on the real world sensor nodes and shown the performance of the public-key based scheme is practical.

4.1 Network and Security Model

We consider a large scale wireless sensor network deployed in a variety of environments. Sensor nodes are the low-cost wireless devices and have very limited hardware resources including processor, memory and energy. Upon detection of an event, the sensor nodes generate event report packets and send them back to the sink through multihop routing. For the event detection that needs the collaboration of a group of nearby sensors, we assume the sensor clustering protocol, as proposed in prior work [45, 105, 4, 5, 17, 3, 2], has been already deployed. The event report is generated by the sensor cluster and transmitted to the sink by the multi-hop routing protocol. We assume the sensor network routing scheme, such as Directed Diffusion [49], LEACH [45] or GPSR [55], is also deployed.

The sensor network security is managed by a Certification Authority (CA), which is responsible for generating all security credentials and distributing the secret keys. Due to the constrained resources and costly wireless communications on sensors, the CA can not be online and accessible as the way it runs in Public Key Infrastructure (PKI). Instead, the CA only runs during the network deployment, system rekeying, or sensor replenishing period. Since the CA has to be off-line in most of time, each sensor has to be pre-loaded with its private key, public key and certificate before the deployment. Each sensor uses these keys to build the secure communication channels

with its neighboring sensors as well as perform future sensing tasks.

An adversary is assumed to use all possible means to attack the message authentication mechanism in the sensor network. To capture the system secret, the adversary may launch either passive or active attacks. A typical passive attack is message eavesdropping. The active attacks, however, may include Man-In-The-Middle (MITM) and sensor compromise. Due to the limited hardware resources, sensor nodes may be compromised upon capture. In this chapter, we assume the adversary can retrieve all secret information from compromised sensors. However, we assume that at most $t-1$ sensors can be compromised. The assumption is reasonable because compromising sensors takes time and effort. In addition to the system secret capture, this chapter focuses on the adversary DoS attack. The adversary may forge the event reports and inundate these messages in the network in order to deplete the batter power of sensors and finally paralyze the network.

Finally, we assume the event detected by a sensor group (or cluster) with t members is always genuine. It is true that the adversary may generate a fake event or a forged value to confuse the base station. The adversary can influence the group decision through various attacks, such as the Sybil attack. However, it is obviously out of the scope of the security problem addressed in this chapter, and prior work [31, 72] has already proposed schemes to defend against such attacks. We thus do not explicitly address the security problem in event detection.

4.2 Public-key based False Data Filtering (PDF)

In this section, we present PDF, a public-key based false data filtering scheme. The basic idea is to generate a system signature for each event report so that any intermediate node with the system public key can easily verify the event report and drop the false data packets. While public key signature generation and verification have been well established in Internet, its application

in wireless sensor network poses a unique challenge. To generate a system signature, the sensor node has to have the system private key. However, any single sensor cannot be trusted to hold the secret because it is vulnerable to adversary's compromise attack. Our PDF solves the problem by using Shamir's secret sharing. Instead of giving the system secret to each individual sensor, PDF distributes the secret in the following way: each sensor holds a unique share of the secret and any t sensor can collaborate together and reconstruct the secret. Therefore, each event report message has to be endorsed by t sensor nodes. The t endorsing sensors actually jointly generate a system signature for the endorsed packet.

We first briefly introduce Shamir's secret sharing scheme. Second, to achieve the least overhead as possible, we then adopt the ECPVS signature scheme [14]. Third, we present the threshold endorsement false report filtering scheme. Finally, we provide the cost and security analysis, as well as the extension of probabilistic verification to reduce the computation cost. Our discussion in this chapter assumes that sensors have already established the pairwise keys with their neighbors by using the pairwise key estabablishment schemes discussed in Chapter 3.

4.2.1 Shamir's Secret Sharing

We assume CA maintains a system secret polynomial:

$$f(y) = a_0 + a_1 y + a_2 y^2 + \ldots + a_{t-1} y^{t-1}. \tag{4.1}$$

$a_0, a_1, \ldots a_{t-1}$ are random number picked in $GF(q)$. System secret x is picked as $x = a_0$.

During the sensor network deployment, each sensor (identified by s_i) is pre-distributed with a secret share of x. In particular, the secret share for sensor s_i is $x_i = f(s_i)$. Any t sensor nodes can reconstruct the system secret by Lagrange interpolation: $x = \sum_{i=1}^{t} l_i x_i$, where $l_i = \prod_{j=1, j \neq i}^{t} \frac{s_j}{s_j - s_i}$

is Lagrange coefficient. However, it is computationally infeasible for any $t-1$ or less sensors to reconstruct the system secret.

4.2.2 ECPVS Signature Scheme

The typical digital signature scheme in ECC is the elliptic curve version of Digital Signature Algorithm (DSA), also know as ECDSA. ECDSA produces 40 byte signature, which is much smaller than 128 byte signature of RSA. However, we are still concerned that the 60-byte message payload (combining a 20-byte message and its 40-byte ECDSA signature) is still too large for a typical data packet for sensor network (e.g., 29 bytes in TinyOS for MICAz motes). Therefore, we adopt ECPVS signature scheme which offers smaller signature size than ECDSA.

We describe the ECPVS [14] signature scheme as following. Given a message M, we divide M to $C||V$, where C and V are two parts of the message M, and $|C|+|V| \geq |M|$, because it is necessary to arrange some redundant information to be included in C. For example, C holds some secret information and the signer identity, while V holds the sender identity, message description, time stamp, etc. We assume the signer has her private key x, and the corresponding public key $Q = xP$. The signer performs the following steps to sign the message.

1. Choose a random key k in $[1, q-1]$;
2. Compute kP, resulting a point with coordinate (x_k, y_k), let $r = x_k$. Check $r \pmod q$, go back to the first step if the result is zero;
3. Compute $e = ENC(r, C)$;
4. Compute $d = H(e||V)$;
5. Compute $\sigma = x \cdot d + k \bmod q$;

6. (e, σ) is the digital signature.

The ENC denotes a symmetric-key encryption algorithm. Similarly, we later denote ENC^{-1} as a decryption operation, which usually uses the same symmetric-key encryption algorithm. The signer sends $< V, e, \sigma >$ to the receiver. To verify the message $M = C||V$ and the signature, the receiver needs to do following steps.

1. Compute $d = H(e||V)$;

2. Compute $R = \sigma P - dQ$;

3. Compute $C = ENC^{-1}(X(R), e)$;

4. Check the redundant information in C.

4.2.3 Threshold Signature Generation

Our event report signature generation scheme combines the ECPVS digital signature and Shamir secret sharing scheme [82] to generate the threshold signature. Examining the ECPVS protocol presented in Section 4.2.2, the signer has to have secret k and x. Considering a group of local sensors are the signer, the challenge of signature generation is how the group jointly constructs k and x (step 1 of ECPVS signature generation), the encryption of the content C (in step 3), and the calculation of *sigma* (in step 5). Note that any member of the group should not learn and reveal any information about k and x, assuming the adversary may capture all the communications inside the group.

We adopt Shamir's secret sharing scheme [82] to share system secret x. To achieve that, CA maintains a secret polynomial: $f_x(y) = x + a_1 y + \cdots + a_{t-1} y^{t-1}$. Before being deployed, each sensor s_i receives a secret share of $f_x(y)$, which is denoted as x_i, and $x_i = f_x(s_i)$. Any t sensors can

reconstruct the system private key: $x = \sum_{i=1}^{t} x_i l_i$, where l_i is the Lagrange coefficient. Any $t-1$ or less sensors, on the other hand, can not compromise system secret x because of the threshold property.

Shamir's secret sharing system discussed above, however, can not be used to share the secret random number k. The reason is that ECPVS signature scheme requires the signer should pick a different random k for a different signature. Otherwise, an adversary may easily derive system secret x by only capturing two signatures generated from the same k. To share a different random secret k among the group of sensors each time, we adopt the Joint Shamir Random secret sharing scheme [82]. This scheme allows all participating sensors to generate their own random secret polynomials (similar to $f_x(y)$) each time. To share a random secret k, each sensor in turn acts as a dealer to distribute the share of the secret (of his own polynomial) to the other members in the group. It should be emphasized that the polynomial shares must be distributed through the secure communication channels. We assume sensors already establish the pair-wise keys with their neighboring sensors by using existing schemes [28, 26, 16, 60, 15, 99]. In particular, sensor s_i generates its secret random polynomial $f_{s_i}(y)$, and distributes the share of secret $f_{s_i}(s_j)$ to sensor s_j $(1 \leq j \leq t, j \neq i)$. Then, each sensor receives $t-1$ secret shares from the other $t-1$ sensor in the group, and one share of its own. By combining these t secret shares, each sensor s_i computes its own share of k, denoted as k_i, and $k_i = \sum_{j=1}^{t} f_{s_j}(s_i)$. The shared secret, as the random number k, is actually embedded in the polynomial that is the summation of t secret random polynomial generated by each of t sensors, $g(y) = \sum_{i=1}^{t} f_{s_i}(y)$. The secret k is determined by: $k = g(0)$. In this way, no sensor in the group knows the value of k. Any t sensors, however, can jointly reconstruct k by using Larrange interpolation: $k = \sum_{i=1}^{t} k_i l_i$. Again, l_i is the Lagrange coefficient.

With both k and x shared, the event report threshold signature generation scheme is illustrated

for (each sensor $s_i, i = 1, 2 \cdots, t$)

$s_i \rightarrow s_1 : P_i = k_i l_i P$

$s_1 \rightarrow s_2, s_3, \cdots, s_t : R = \sum_{j=1}^{t} P_j$(i.e., $R = \sum_{j=1}^{t} k_j l_j P$)

for (each sensor $s_i, i = 1, 2 \cdots, t$)

$s_i : e = MAC(X(R), C)$

$s_i : d = H(e||V)$

$s_i : \sigma_i = x_i l_i d + k_i l_i$

$s_i \rightarrow s_1 : \sigma_i$

$s_1 : \sigma = \sum_{i=1}^{t} \sigma_i$

(i.e., $\sigma = \sum_{i=1}^{t} x_i l_i d + \sum_{i=1}^{t} k_i l_i = xd + k$)

Figure 4.1: Event report threshold signature generation scheme by t sensor nodes, $s_1, s_2, \cdots, s_t$.

in Fig 4.1. We assume t sensors, $s_1, \cdots, s_t$, detect the event, denoted as $M = C||V$, where C can be the secret event measures and V can be general event description. We also assume s_1 is elected as the group leader. First, t sensors construct kP. Each sensor s_i sends its share $k_i l_i P$ to group leader s_1 (l_i is the Lagrange coefficient), which in-turn sums the t shares to get kP (by Lagrange interpolation), denoted as R. Then, s_1 broadcasts R to the rest $t-1$ sensors. Each sensor uses R to generate e and d as shown in Fig 4.1, computes its share of the system signature: $\sigma_i = x_i l_i d + k_i l_i$, and send it to s_1 through the secure communication channel. The summation of t shares of signatures produces the system signature: $\sigma = \sum_{i=1}^{t} x_i l_i d + \sum_{i=1}^{t} k_i l_i = xd + k$. Finally, s_1 sends (σ, e, V) to the destination, either the sink or other remote sensor node.

An important security measure of the above joint signature generation protocol is not to reveal

any of system secrets, x, k, and individual sensor secret shares, s_i, x_i, at any step of the protocol. Otherwise, if the group leader is compromised, then system secrets are compromised. When kP is constructed to encrypt C (in step 2 of ECPVS), each sensor s_i submits $k_i l_i P$ instead of explicit k_i, so that secret share k_i is protected by the security property of discrete logarithm problem, i.e., it is infeasible to derive k_i from $k_i P$. When the signature σ is built (in step 5 of ECPVS), the partial signature σ_i submitted by each sensor is the linear combination of two unknown secrets x_i, k_i, so the group leader has no way to derive the values of x_i, k_i from σ_i. Overall, all secrets are well protected during the signature generation by the group.

4.2.4 Cost Analysis

The t endorsing sensors have to jointly generate a random value k for each event report. To share a random k, each participating sensor s_i first generates its own random polynomial $f_{s_i}(y)$, and calculate the secret shares for other members in the group. For the group with t members, each sensor has to compute t shares of the $t-1$ degree polynomial, including the one for itself. We will show in the evaluation that the polynomial calculation is efficient for the motes. For the message complexity, each sensor sends $t-1$ secret shares to the $t-1$ members, and receives $t-1$ shares from the $t-1$ members. Therefore, each sensor has to send and receive $2(t-1)$ messages.

Note the share of k can be pre-computed. The group of sensors can run the secret sharing protocol at the idle time before the event is detected, so the shares of a new k is ready for the next endorsement as long as the different events do not occur at the same location at the same time. Another way to reduce the communication overhead for k sharing is to eliminate the k sharing procedure by using pre-computation. If sensor nodes have enough storage space, CA can pre-compute different polynomials and pre-load the shares into the sensors during the deployment.

Each share is associated with an index number. To endorse a new signature, the group of sensors only need to negotiate a new index, and use that share to construct a new random k. In this way, the message complexity can be reduced to the minimum.

After the shares of k are ready, the most expensive computation for each sensor s_i is one ECC point multiplication to compute P_i as shown in Fig. 4.1. For the message complexity, each sensor needs to send or receive two points and one scalar value, which includes its share P_i, the value of kP, and its share of σ.

The event report message consists of σ, e and V. Since V has the half size of e, the total message length is the size of two and half scalars. The computational cost to verify the report, as shown in Section 4.2.2, is two ECC point multiplications.

4.2.5 Security Analysis

Our security analysis of the threshold signature generation scheme focuses on following two threats. We first check whether or not the adversary can infer the system secret by compromising one or more sensors and collaborating with other sensors in signature generation. Second, we examine the security resilience of secret k sharing because the compromise of k will lead to the whole system secret compromise. Note, the security resilience of sharing secret secret x by any t sensors is guaranteed by Shamir's secret sharing scheme. As long as there are no more than t sensors are compromised, there is no feasible solution to get x.

A compromised group leader certainly may cause greater security threat than other sensors since the leader collects more information, so the following analysis is based on the assumption that the group leader is compromised. The group leader (s_1) receives t shares of kP from other endorsing sensors and derive the value of kP, but the values of these points do not reveal any

information of k_i or k due to the security property of ECC. The group leader s_1 also receives t shares of system signature. In each share, $\sigma_i = x_i l_i d + k_i l_i$, there are two unknown values: x_i and k_i. Any single or multiple shares combined does not reveal any information of x_i and k_i. Therefore, s_1 has no way to determine the system secret x and the random k without physically compromising the rest $t-1$ endorsing sensors. As we can see, even though s_1 can be compromised, the adversary still cannot obtain the system secret to generate the signature for his injected data.

The shared random number k has a critical security role in PDF scheme. As we discussed previously, the compromise of k directly leads the compromise of system secret x. Therefore, any one or more (less than t) sensors must not get any information of k during the signature generate, otherwise the compromise attack may allow the adversary to acquire k. In Joint Shamir Random Secret sharing scheme [82], secret k is embedded in the polynomial that is the combination of each secret polynomial of t sensors. Each group member s_i holds a secret share of k, k_i. As long as at least one sensor is not compromised, the adversary can not get t secret shares, and thus can not reconstruct k. Further, since sensors do not directly send their secret shares k_i to the group leader, (instead they bind their k_i with x_i and the endorsed messages abstract d), the traffic monitoring does not give the adversary any chance to obtain the secret shares. Note the communication channel between sensors are encrypted to raise the security threshold and defend against other security attacks, including message injection and impersonate attacks.

One may wonder whether the compromised group leader can generate the signature for the forged messages because it can collect t partial signatures. This forged signature generation attempt, again, will fail because the partial signature submitted from each group member is bind with the endorsed event. In particular, the message abstract d is bind with each partial signature. If these partial signatures are used on a forged message, ECPVS verification will fail and the

forged message will be immediately dropped by the forwarding sensors.

Finally, PDF scheme does not prevent the adversary's disruption attacks. The disruption attack happens when the adversary (by compromising one or more sensors) intentionally submits a corrupted partial secret or signature and then disturbs the signature generation. As the result, the generated threshold signature is invalid, and the legitimate event report will be dropped by the forwarding sensors. Several schemes have been proposed to identify the compromised sensors in prior studies [108, 111, 27], the security solution for defending against the disruption or false negative attacks is still an open research problem [104], and is considered as one of our future work. It should be emphasized that the disruption attack may trigger the system attention from the base station or sink because the network abnormality can be detected if many legitimate reports are dropped due to the attack. Then, the administration personnel can physically locate the compromised sensors and remove them from the network.

4.2.6 Probabilistic False Data Filtering

Given the event report with system signature, any intermediate forwarding sensor can easily verify the signature and decide whether or not to drop the packet. Theoretically, starting from the source node (s_1) to the destination, only one verification is enough to filter the possible false data packet. The signature verification at every hop is not necessary. However, considering the adversary's DoS attack can occur at any location in the network, one signature verification is not adequate because the adversary can inject the false data after the node that verifies the signature. Therefore, we propose the probabilistic false data filtering to balance the trade-off between computation overhead and the DoS attack prevention.

We denote p_f, a system wide parameter, as the en-route verification probability. Any inter-

mediate forwarding sensor, with the probability of p_f, verifies the system signature by using the verification method presented in section 4.2.2. The verifying sensor first calculates $d = h(e||V)$, then deduces $R = \sigma P - dQ$ (P is the base point, and Q is the system public key). The value of X-coordinate of R is used to recover C, which is the part of original message M. Finally, the verifying sensor compares the redundant information in C with V. The event report message will be regarded authentic if the verification is successful. Otherwise, the message will be immediately dropped.

4.3 Performance Evaluation

We evaluate our proposed PDF scheme by implementing all components on the real world experiment test-bed, including sensor confidential generation and pre-loading, security communication channel establishing, random secret number sharing, and threshold signature generation.

4.3.1 Experiment Testbed and Parameter Setting

Our experiments use MICAz [48] motes as the sensor platform. MICAz is powered by an ATmega128 micro-controller, which features an 8MHz, 8-bit RISC CPU, 128K bytes flash memory (ROM) and 4K RAM. The RF transceiver on MICAz is IEEE 802.15.4/ZigBee compliant, and can achieve maximum 250kbps data rate. Our MICAz motes run TinyOS [88] version 1.1.15.

We implement ECC public key primitives on MICAz motes. We choose SECG recommended 160-bit elliptic curve, secp160r1, in our ECC implementation. The 160-bit ECC offers the same security level as the 1024-bit RSA does [73], which is a more popular public key scheme and widely used in e-commerce. The performance of threshold signature generation and public key verification directly determines the performance of PDF. The current ECC implementation in the

public domain suffers very poor performance if ported directly. It is reported in [62] that it takes more than 30s to generate a public key. To significantly reduce the computation time for ECC exponentiation, we have adopted a number of optimization techniques customized for the 8-bit architecture, including Hybrid Multiplication and Pseudo Mersenne modular reduction for large integer multiplication, Mixed Coordination for efficient ECC additions and doubling, etc. Due to the space limit, this chapter omits the detail description of the optimizations. We refer interested readers to [93] for details. We summarize the key performance results in Table 4.1.

Platform	FPM	RPM	Sign	Verify
MICAz	1.24s	1.35s	1.35s	1.96s

Table 4.1: The performance 160-bit ECC on MICAz mote, including fix point multiplication (FPM), random point multiplication (RPM), signature generation (Sign) and signature verification (Verify).

We run the experiment in an office room with the dimension of 15ft by 10ft. The sensors are evenly placed on a table with the average distance of 2ft with each other. To achieve the better communication efficiency, we change the default TinyOS data packet payload size to 68 bytes (including 4-byte control information) from the original 29 bytes. This allows us to transmit an ECC public key (40 bytes) in one data packet. One possible trade-off for payload size extension is that the packet may suffer transmission errors more easily than that of the default size. As the result, the communication efficiency can be affected when packet loss happens. Our experiment results, however, show the packet loss is rare after the payload size extension. It could be the reason that our sensor deployment condition is too ideal to show the difference after payload size extension. It is one of our future work to study the communication efficiency in an outdoor and realistic deployment condition.

With all security components implemented, the program has the code size (ROM) of 35,108

bytes and the data size (RAM) of 2,648 bytes. Given the capacity of 128KB programming (ROM) size and 4KB data (RAM) size, we only use less than 30% of the programming size. The rest space can be reserved for other applications or future expansion. One may be concerned that we have consumed about 65% of the data size so that other applications may have memory shortage. One feasible solution is to move the constant variables (for ECC parameters) from RAM to on-board permanent storage (EPROM or flash). Further, more optimized and careful programming can also ease the memory shortage.

Our evaluation focuses on the time consumption, including the communication delay and the computation delay. We do not explicitly give the performance of power consumption, because the combination of message complexity and time consumption can always be approximately translated to the power consumption. In the experiment, we have also adopted the simple scheduling scheme so that the probability for the packet corruption due to the collision is very small. During the experiment, we repeat each test for 20 times, and record the average time consumption. We finally discuss the PDF scheme message overhead and its scalability when the network size grows.

Since the pairwise key performance evaluation has been studied in Chapter 3, we omit this part in this chapter.

4.3.2 Evaluation of Threshold Signature Generation

In this subsection, we evaluate the false data filtering performance. We first present the performance of the two components in PDF: threshold signature generation and signature verification. We then use the results to estimate the overall performance with different hop-by-hop authentication probabilities.

It is important that, in the threshold signature generation, the group of t local sensors need

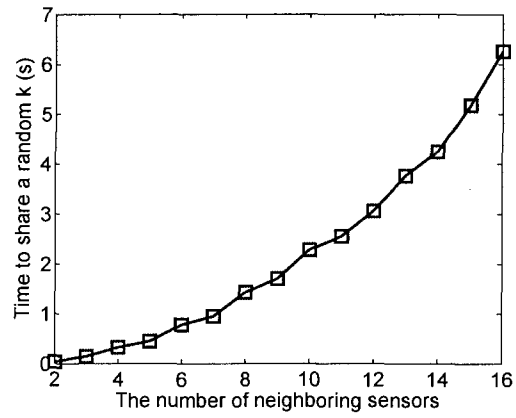


Figure 4.2: The time duration for the group of sensors to share a random secret k.

to share a different random secret k for each signature. Therefore, we first evaluate the cost for random secret (k) sharing. In the experiment, we first schedule all the motes to generate their random secret polynomials simultaneously, as well as the 20 byte secret shares for each of the other sensors in the group. Then, all the motes in turn unicast their secret shares to the corresponding sensors. We measure the time consumption in the whole process. The experiment results are illustrated in Fig 4.2. We find the cost for sharing a random secret is not negligible but reasonable. For a group of 8 sensors, it takes only 1.8 seconds. The time consumption increases quadratically with the sensor group growing because the key graph edges increase with $O(n^2)$ (suppose n is the number of endorsing sensors). As the result, the communication complexity is $O(n^2)$. For a sensor group with 16 nodes, it then takes 5.8 seconds to share a random k.

Note the random k sharing protocol can be executed in the idle time before the event is detected, so that the random secret can be immediately used for endorsing the event upon detection. Therefore, the time duration for the threshold signature usually does not include the time delay for sharing k unless more than one different events occur simultaneously at the same location. Based on the above reason, our experiment for measuring the time delay for the threshold signa-

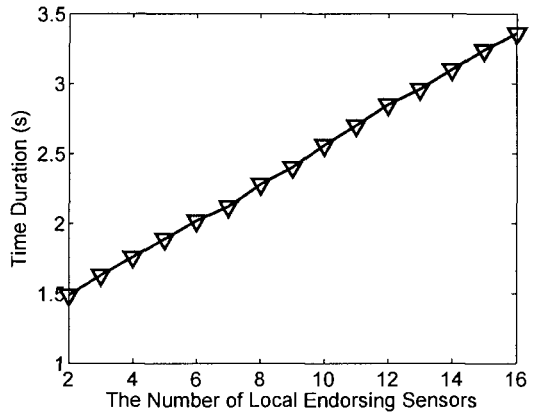


Figure 4.3: The time duration for the group of local sensors to generate threshold system signature for the event report.

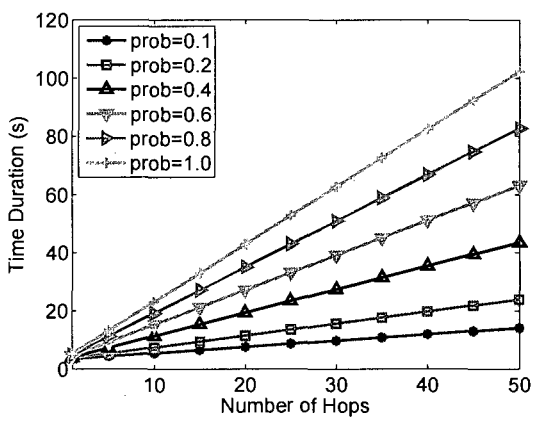


Figure 4.4: The overall time duration of false data filtering performance under different probabilistic filtering value.

ture generation does not include the random k sharing time. We present the experiment results in Fig 4.3. In general, the threshold signature generation is efficient because each endorsing sensor only needs to do one ECC point multiplication. With 8 local endorsing sensors, the time duration is 2.3 seconds. The time linearly increases to 3.3 seconds when the number of endorsing sensors becomes 16.

The system signature verification is equivalent to an ECC signature verification operation. The verification time for an intermediate forwarding sensor is 1.96s.

We are eager to investigate the overall performance of PDF, including threshold signature generation and the probabilistic false data filtering. In our evaluation, we assume that the event detecting sensors have already established pair-wise key with their neighbor endorsing sensors. We also assume these sensors have already shared a random secret k, which is used to generate the threshold signature. We fix the number of endorsing sensors to 16. Fig. 4.4 demonstrates the overall performance of the false data filtering scheme under different hop-by-hop verification probabilities. As we can see, as long as the system parameter is properly selected, e.g., the verification probability is 10% or 20%, the overall performance of PDF is reasonably practical. Given the event report destination within less than 20 hops, the end-to-end delivery time is less than 10s. While the delivery distance increases to 50 hops, the delivery time moderately increases to around 20s.

4.3.3 PDF Message Overhead and its Scalability

In addition to the time delay, PDF scheme also introduces extra messages to the sensor network. Since message complexity analysis for each group sensor was presented in the previous section, we here discuss the overall message overhead that PDF brings to the system. The overall message overhead is important because it shows how much communication cost the system has to pay to deploy PDF scheme to defend against the adversary's DoS attack.

The extra messages required in PDF scheme are used to share the random secret k, generate the group signature and the event signature attached to each event report message. Note we do not count the message overhead in pair-wise key establishing because it provides basic security infrastructure to the sensor network and can be included in any other security scheme besides PDF. Suppose there are t sensors in the group. As we indicated in the previous section, to share a random

secret k, each sensor needs to send $t-1$ messages, so t sensors totally send $t(t-1)$ messages. As showed in Fig. 4.1, the signature generation scheme, each group sensor sends two messages to the group leader, and the group leader sends one message to the rest of group. The number of message combined is 3(t-1). In total, the number of extra messages to generate a signature in PDF scheme is (t+3)(t-1).

As the ECPVS scheme shows in Section 4.2, once the signature is generated, the group leader sends the event report message in the format of (V, e, σ), where V is the public part of the message, e is the encrypted C, and σ is the group signature. Since the original message is $C||V$, the extra part sent in PDF scheme is just σ, which has the length of 20 bytes in 160-bit ECC system. Note this overhead is counted as per hop. If the average hop number is h, the total amount of message overhead for signature transmission is $20h$.

The above analysis reveals that the message overhead for signature generation is not related to the network size, and is only determined by the size of group (t). In event report transmission, PDF puts 20 bytes overhead in each event report. Considering that the average event report delivery distance may increase when the network size grows, PDF scheme may introduce the network size related message overhead. This overhead, however, can be very minimal as 20 bytes can be transmitted in the same message with the moderate payload size inflation.

Chapter 5

Location Privacy

Sensor networks will be prevalent in the near future for various applications, including object and event monitoring. A common communication paradigm for sensors is to obtain information about objects or events and send the data back to a base station (or sink) for further analysis. The wireless communication path from the object to the base station may jeopardize the safety of the object if an adversary, who is capable of detecting the message flow, traces back to the message source by moving along the reversed path. The object, e.g., an animal of an endangered species, or a vehicle of military aides, may have to be protected for safety reasons and the related location information should not be disclosed. This concern will become even more serious for future sensor network prevalence in pervasive computing applications, as the ubiquitous information collections doubtlessly encroaches on the privacy of the people involved.

In this chapter, we explore the location privacy problem in sensor networks. We aim to hide the location of the message source and make it more difficult for an adversary to trace messages back to the source location. We assume that a security infrastructure, such as secure communication, has already been built in. That is, no information carried in the message (e.g., packet head) will

be disclosed, allowing the adversary to gain any knowledge about where the message comes from. The adversary observes the wireless communication within a certain detection range and traces toward the message source by moving, in each step, to the node that transmits the detected target information.

The location privacy in sensor networks is very different from Internet anonymity and privacy problems which have received extensive attention [18, 19, 78, 77, 50, 36]. The location privacy discussed in this dissertation has two fundamental differences from prior work. First, Internet anonymity relies upon channel secrecy (e.g., secret keys) to protect logical location privacy, while location privacy in this dissertation addresses the issue of physical location privacy. For example, there is a strong connection between the message header and the identity of the Internet users, while this kind of binding does not exist in wireless sensor networks. Instead, the location of the source sensor node is detected by the radio signal rather than the message content, given the assumption that all messages are encrypted. Second, there is no power constraint for Internet users, but energy is one of the most critical issues in sensor networks. In the Internet, a user may choose any number of proxies [77] or join in a large and geographically diverse crowd [78] to achieve anonymity. On the contrary, the energy budget in sensor networks is extremely constrained.

Many message routing protocols have been proposed for sensor networks [49, 55, 102, 103, 45]. None of them are designed for location privacy protection. Kamat *et. al* [53] proposed Phantom routing to solve a similar privacy issue. However, as we will show in Section 5.5.3, the random-walk-based Phantom routing has poor performance in defending against the adversary's traceback, even if the adversary has very limited traffic monitoring ability. More recently, [64, 84] propose source location protection schemes under a global traffic analyzer. The two approaches only partially solve the problem. The ConstRate and k-anonymity [64] schemes rely on global sensor

stimulation and are very resource demanding. FitProbRate [84], however, sacrifices location privacy for short message delivery delay. As we will present in Section 5.6, our solution minimizes message delay while still achieving the perfect location privacy in the presence of a global attacker.

Several papers ([25, 38, 113, 1, 52, 34]) discussed privacy and anonymity issues in wireless communications, and propose solutions by manipulating the message contents. The approaches proposed in [25, 38, 113] either encrypt or modify the message content (data cloaking) to confuse the adversary and achieve privacy. The Mist Routers [1] offered both location privacy and anonymous communication in ubiquitous computing environments by combining a hierarchical mixed network and a message encryption scheme. In comparison, [52, 34] address the privacy issue from the traffic analysis perspective. Jiang *et al.* [52] proposed a cover mode to keep the protected message flow indistinguishable from the rest of the traffic. Fu *et al.* [34] designed a digital filtering technology to defeat the flow marking attacks that could degrade anonymity. In contrast to their schemes, this dissertation addresses the location privacy threat due to the physical wireless medium that allows the adversary to perform traffic analysis to derive the message flows.

The papers most relevant to our work about privacy in sensor networks are [74, 53, 64, 84]. Ozturk *et al.* addressed concern about the originator location privacy [74] in sensor networks. They identified the location privacy issue by using a vivid example *Panda-Hunter Game*, then discussed a possible encryption and routing scheme to prevent the adversary (hunter) from locating the panda. Kamat *et al.* [53] continued the work and proposed the Phantom Routing scheme. Message delivery in Phantom Routing is conducted in two phases: First, messages are routed a fixed number of hops by using random walk; Second, after finishing random walking, messages are delivered to the sink by using flooding or single path routing. Compared to the routing scheme (e.g., shortest path routing) without any privacy protection, Phantom routing can achieve a certain

degree of location privacy, even though the performance is not satisfying (as we will show in our simulation results). The drawback of this approach is lacking the intuition of routing strategy. In comparison, this dissertation presents the theoretical foundation in designing a privacy-aware routing in sensor networks. More recently, [64, 84] proposed location privacy protection schemes under the presence of a global eavesdropper, the second adversary model considered in this chapter. Mehta *et al.* presented two techniques: periodic collection and source simulation. However, the paper does not present the detailed routing scheme that delivers data to the sink during the collecting period. Meanwhile, the source simulation scheme is limited to applications which the source moving pattern is pre-known. The FitProbRate scheme proposed by Shao *et al.* greatly shortens the message delay with the price of sacrificing source location privacy. In comparison, we strive for achieving the minimum message delay and perfect location privacy at the same time under the presence of a global eavesdropper.

We start the discussion from a simple model where there is only one source node and one adversary, and the adversary always starts the traceback from the sink location. As we will show in Section 5.2.1 and Section 5.4.4, our theoretical model can also be applied for multiple adversaries and multiple data sources. The time for the adversary to trace back to the source is a natural metric for location privacy. Even if the adversary has limited monitoring power, the adversary can follow any random message path and thus trace back to the message source. We use average traceback time and the possible minimal traceback time it takes for an adversary to reach the source as two metrics for location privacy. Average traceback time signifies an expected performance for location privacy. The minimal traceback time, which shows the worst case scenario, assumes that the adversary has the best luck possible, taking the route with the shortest time to find the source.

We address the location privacy issue under a complete adversary model. When the adver-

sary has limited detecting power, we design routing algorithms to maximize the traceback time. We formulate this problem as an optimization problem constrained by the energy budgets that are allowed for use in message routing. To gain more understanding about this issue, we have tried to look at the problem from different perspectives. First, we give an approximation of the performance bound in a generalized scenario as a guideline for network routing design. Our result indicates that the traceback time is proportional to the number of nodes involved in routing. Given a certain sensor density, the number of nodes participating in message routing indicates the degree of how dispersed in the message routes, which produces longer and more scrambled routing paths that delay the adversary's traceback progress. Then, we show how to optimize the routing performance by considering several special cases in which fixed routes are given. The fixed routes are also categorized as routes that are well separated, without intersection in the middle and splicing routes. Although this seems quite restricted, many applications fit in these constraints. For example, an application may require the routes to be well separated so that the adversary has little chance to capture sufficient messages for message content decryption. In addition, many applications also dictate fixed routes to avoid certain dangerous areas where adversaries gather, or to force the routes to pass through certain points for various reasons such as information multicast or data aggregation.

When the adversary is more powerful, e.g., being capable of deploying a sensor network to monitor the traffic, we propose a random schedule scheme in which each node transmits at a certain time slot in a fixed period such that the adversary would not be able to profile the difference in communication patterns among all the nodes. Obviously, this scheme requires a large number of sensors to participate in the message transmission between the source and the sink, so that only a very small portion of these sensors (which are on the routing path) transmit the valid messages;

others just send dummy messages. From the adversary's point of view, the sensors in the whole area are flooding messages and no routing path can be inferred from the communication pattern. As radio communication consumes a significant amount of energy in sensors, our goal is to minimize the message transmission delay so as to keep this "flooding" period as short as possible. There are two ways to reduce the message transmission delay: either increase the data rate or use more routes between the source and the sink. Considering that the message rate at the forwarding nodes cannot be changed (otherwise the adversary would easily identify the message forwarding nodes and then the routing path), the problem of minimizing the message transmitting delay is equivalent to finding as many disjoint routing paths as possible so that more message packets can be routed in parallel. We give an approximation algorithm to find the optimal k disjoint routing paths to deliver the data messages.

5.1 Network and Adversary Model

We consider a wireless sensor network consisting of sensor nodes hat are uniformly and randomly scattered in a sensor field. Each node has the capabilities to collect data and route data to the sink in a multihop fashion. We assume sensor nodes are evenly distributed in the sensor field and do not move after being deployed.

We consider two types of adversary models. First, we focus on the single-adversary model. It will be shown in the next section that the (limited) multiple-adversary model still obeys the general performance of our adversary model. Once an adversary gets close to the source, the source will be disclosed. This may not be true in all cases, but in many scenarios the adversary is capable to detect the source by other means (other than eavesdropping) within a certain range. We describe the adversary's radio detection model as follows. The adversary may carry a portable or car

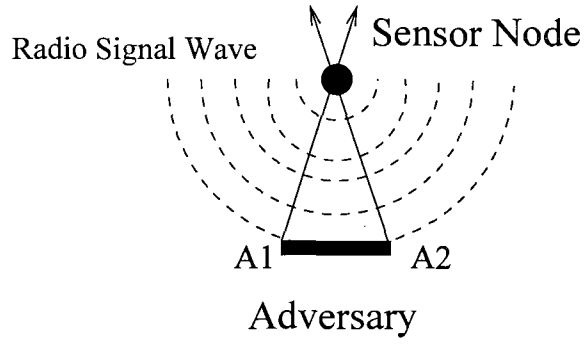


Figure 5.1: Adversary's radio detection model: The portable or car based Radio Direction Finder is equipped with multiple antennas, shown as A1 and A2. With multiple separate receivers, the adversary can easily use triangulation to locate the transmitting sensor node.

based Radio Direction Finder [46]. This type of device normally is equipped with two or multiple separate antennas. As shown in Figure 5.1, the adversary has two antennas $A1$ and $A2$. Upon receiving radio signal from the antennas, the adversary can easily triangulate on the transmitter. It is also very possible that two or more adversaries work together. By applying current sensor node localization techniques, they can easily pin-point the location of the transmitter. Once detecting a message signal, the adversary quickly moves to the transmitter's location and starts the next message detecting. By repeating this procedure, the adversary can trace back on the message routing path and finally locate the source node. In this work, we assume the adversary's radio detection is always successful and correct. Second, we extend our discussion to more powerful adversaries. In the worst case, the adversaries may deploy a similar sensor network to monitor every activity at every location. Under such situation, any routing scheme proposed for the first adversary model will fail to protect the location privacy because the source sensor node activity will be immediately detected by the adversary deployed sensors.

Many routing schemes are constrained by the energy consumption. We use a very simple energy consumption model: each node sending a message (i.e., a packet) costs one unit of energy. The energy consumption for receiving and the node sleeping/wake-up schedule can be carefully

considered to fit into this model. We omit this detail for space constraint. In the rest of the chapter, the amount of messages sent in total and the energy consumption are all normalized. We assume each data packet has enough space to carry one message. Then the amount of consumed energy for a message is equal to the path length. Thus we use energy and path length interchangeably.

We model network routes in a directed graph. An edge (A, B) exists if and only if AB is a valid link in one of the routes. Our goal is assign message flow to all the links (the route segments) so that the traceback time can be maximized. After the message flow is assigned, the routing becomes simple: each node randomly picks a downstream node for message relay according to the flow distribution. In the rest of the chapter, except specified, all the routing schemes follow this message distribution model.

In [100], Wright *et al.* described the *predecessor* attack and the *set-up* attack that are effective against various anonymity schemes, including Crowds [78], DC-Net [19], Onion routing [77] and MIX-net [18]. Similarly, their proposed attacking techniques rely on message content analysis, except for multiple collaborating adversaries and timing analysis. As indicated previously, we do not consider this type of attack since we assume that proper encryption has already been applied to the message content (including packet header) so that no content information is revealed. As we will show, our discussion and proposed schemes do address the multiple adversary problem and timing analysis attack. In particular, our analysis of optimum routing schemes under the adversary model with limited detecting power is also valid when there are multiple adversaries conducting traceback simultaneously, and so is our proposed random schedule scheme when our sensor field is globally monitored by an adversary sensor network. To defend against the timing analysis threat under the global adversary model, our random schedule scheme is designed to hide the real message routing path and therefore defeat the adversary's timing analysis attack.

5.2 Performance Bound Analysis

Given a sensor network, we are interested in finding the ultimate location privacy we can achieve. In this section, we first develop the performance bound under the assumption that the adversary has the same radio detection range as the sensors' transmission range. Then, we relax the constraints of the adversary's model and allow the adversary to trace back more than one hop each time. Finally, we present our simulation results from our discrete event-based simulations. The performance bound is an approximation of the adversary traceback time; it is by no means an accurate result.

5.2.1 Performance Bound for General Routing Schemes

To study the performance bound of general routing schemes, we consider a sensor field with randomly and evenly distributed N nodes participating in message routing. Let $Freq(i)$ be the frequency of messages seen at sensor node i. We denote L as the average routing path length, and normalize the sensor node's transmission range to 1. Therefore, L is actually the number of hops between the source node and the sink, averaged over all routes. In this dissertation, we assume the message rate, m, is small enough so that the time interval for sending any two consecutive messages is much larger than the time that it takes the adversary to travel from one node to another. We denote T_c as the traceback time for the adversary to traverse a routing path with L sensors. At node i, it takes $\frac{1}{Freq(i)}$ units of time for the adversary to catch the next message. In total, the traceback time is:

$$T_c = \sum_{i=1}^{L} \frac{1}{Freq(i)}. \tag{5.1}$$

Note the Eq. 5.1 is very general and can be applied to any routing scenario, including multi-path and random routing. When the routing paths are not evenly distributed, and the messages are not evenly dispersed, it is possible that the adversary traceback time on different routing paths can

be different. In that case, Eq. 5.1 is still valid even though the value of T_c would be different for different paths.

For each message generated from the source node, on average it will be propagated L hops along the path from the source node to the sink. Within a time unit, each of m messages reaches L sensor nodes in the sensor field. On the other hand, the total number of routed messages within a time unit can also be given by $\sum_{i=1}^{N} Freq(i)$. Therefore

$$\sum_{i=1}^{N} Freq(i) = m \cdot L. \tag{5.2}$$

If the routing paths are evenly distributed in the sensor field, and the source node randomly and uniformly picks a path for each message, the participating sensor nodes have approximately the same message frequency $\overline{Freq}$. Then Eq. (5.1) and (5.2) will become

$$T_c = L/\overline{Freq}, \tag{5.3}$$

$$N \cdot \overline{Freq} = m \cdot L. \tag{5.4}$$

Combining Eq. (5.3) with (5.4), we have

$$T_c = N/m. \tag{5.5}$$

Note that the above results also apply to the multiple adversary model. Suppose K adversaries collaborate and trace back the messages at the same time. In the best case (for traceback), the adversaries are tracing on K independent routing paths. The traceback is is $1/K$ times of that of one adversary. Therefore, the traceback time for multiple adversaries still obeys the general performance of the single adversary model.

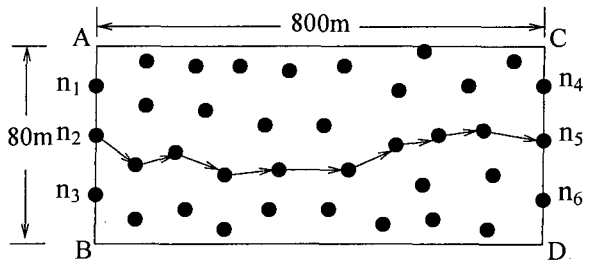


Figure 5.2: Network setup for performance bound simulation.

5.2.2 Performance Bound Analysis

In the previous subsection, we assume the adversary is tracing back one hop each time. Given a longer radio detection ability, the adversary can trace back h hops ($h > 1$) each time. Therefore, Eq. (5.1) should be rewritten as:

$$T_c = \sum_{i=1}^{\lceil L/h \rceil} \frac{1}{Freq(i)}. \tag{5.6}$$

Combining Eq. (5.6) with Eq. (5.4), we have

$$T_c = \frac{N \cdot \lceil L/h \rceil}{m \cdot L} \approx \frac{N}{h \cdot m}. \tag{5.7}$$

Compared with Eq. (5.5), Eq. (5.7) introduces one more factor h. The average traceback time is inversely proportion to adversary detection range h.

Eq. (5.5) and (5.7) reveal that the adversary's average traceback time is determined by the number of nodes involved, the message rate, and the adversary's detection ability. Considering the message rate and the detection model are relatively stable, the only solution that increases location privacy is to have more sensor nodes involved in message routing, which means the routing paths should be dispersed into a larger area.

5.2.3 Simulation Results

We have built a discrete event simulator to study the performance bound of general routing schemes. As shown in Fig. 5.2, we set up a rectangular sensor field with length of 800 m. The sensor node's transmission range is 20 m. In order to simulate the scenario where each involved sensor node has the same message frequency, we design the simulation scheme as follows. On edge AB, we deploy a number of source nodes (the number depends on the length of AB) so that the distance between every two consecutive source nodes is 20 m. For example, given the length 80 m in Fig. 5.2, we deploy three sensor nodes n_1, n_2, and n_3. Then, we deploy the same number of destination nodes on the other edge CD, with the destination nodes paired with different source nodes. For example, n_1 and n_4 form one pair, n_2 and n_5 form another pair. For each time unit, we randomly pick a source node on AB and send a message to its paired destination node on CD. The message routing follows the geographic routing scheme. The adversary can start from any position on CD. A traceback procedure ends as soon as the adversary reaches any position on AB. In order to change the number of nodes involved in routing, we change the width of the network field with the same node density. In a larger network field, we can use more routes and thus more nodes for routing. We use three possible adversary detection ranges in our simulation: 20 m, 30 m and 40 m.

We present our simulation results in Fig. 5.3. Instead of using traceback time T, we actually use the number of messages, for simplicity and accuracy. Eq. (5.5) can be rewritten as:

$$m \cdot T_c = N. \tag{5.8}$$

$m \cdot T_c$ in the left hand side of Eq. (5.8) is the number of messages the adversary needs in order to reach the source node. Fig. 5.3 shows that the adversary's traceback time grows linearly with the

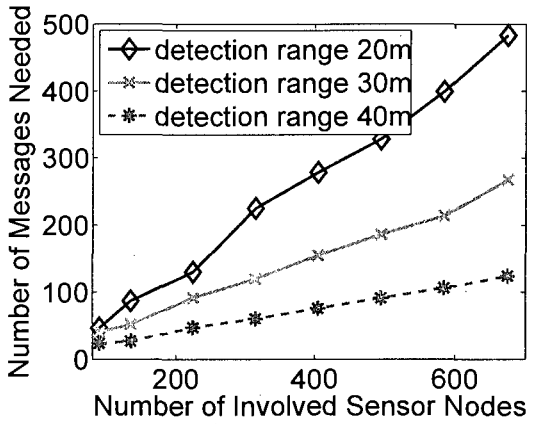


Figure 5.3: The adversary's traceback time vs. the number of sensor nodes under three different adversary detection ranges.

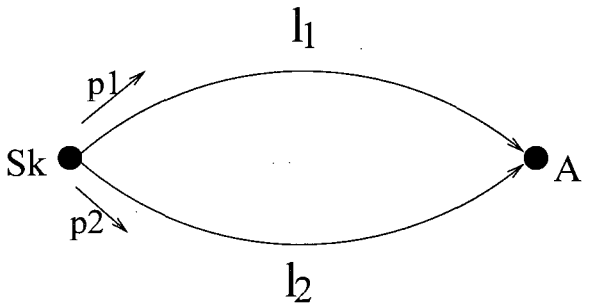


Figure 5.4: Message distribution scheme with only two paths.

increasing number of involved sensor nodes under all three different detection models. Moreover, the slope for the detection range of 40 m is approximately twice the slope for the detection range of 20 m, which also matches Eq. (5.7).

5.3 Average Traceback Time

We have given the approximate performance estimation for any routing scheme, but how to design a routing strategy to maximize the traceback time is still a question. In this section and the next section, we explore the optimal routing strategies under two different performance metrics: average traceback time and minimal traceback time. This section presents the optimal routing scheme

that maximize the average traceback time. We assume the routes are well separated so that there is no transmission interference between any node pair from any two routes, and that the adversary tracing on one route is not able to detect the messages on another route. We start from a simple example with two routing paths. Then, we generalize the problem with n routes.

Suppose we have the routing scenario shown in Fig. 5.4. Source node s_k has the choice to send messages to either of two routing paths with length[1] l_1 and l_2 (from now on, we use l_1 and l_2 to represent the two paths, respectively). Suppose s_k chooses l_1 with probability p_1, and chooses l_2 with probability p_2 ($p_1 + p_2 = 1$). Path l_1 and l_2 intersect at point A, where the adversary is located. Once the adversary starts tracing on one routing path, she will not be able to detect the message on the other path. Therefore, the adversary traceback time along l_1 is l_1/p_1. Similarly, the traceback time along l_2 is l_2/p_2. Starting from point A, the adversary has probability p_1 to get a message coming from l_1 and probability p_2 to get a message coming from l_2. The adversary's average traceback time, T_a, can be given by:

$$T_a = p_1 \cdot \frac{l_1}{p_1} + p_2 \cdot \frac{l_2}{p_2} = l_1 + l_2. \tag{5.9}$$

Let E be the amount of energy required to deliver a message from the source to the sink. We assume that the two routes can be chosen from a range of routes with length between l_0 and l_m ($E \leq l_m$). Given the following constraints:

$$\begin{aligned} & l_m \geq l_1, l_2 \geq l_0, \\ & p_1 + p_2 = 1, \\ & p_1 l_1 + p_2 l_2 \leq E \leq l_m, \end{aligned} \tag{5.10}$$

[1]By length we mean the number of hops on that route.

the average traceback time T_a is maximized when $l_1 + l_2$ achieves its largest possible value. Without loss of generality, we assume $l_1 \leq l_2$. To maximize $l_1 + l_2$, we first increase l_2. Notice that the largest possible value of l_2 is l_m, and $l_1 \leq \frac{E - p_2 l_2}{p_1}$, so we have

$$T_a = l_1 + l_2 \leq \frac{E + l_m(2p_1 - 1)}{p_1} = \frac{E - l_m}{p_1} + 2l_m. \tag{5.11}$$

Since $E - l_m \leq 0$, the maximum value of T_a is achieved when $p_1 = 1$. Therefore, $Max(T_a) = E + l_m$. Note that the value of $Max(T_a)$ cannot be reached unless $l_m = E$. The reason is that if $p_1 = 1$, then $p_2 = 0$, and we cannot use Eq. (5.9) to calculate traceback time. Instead, the traceback time $T_a = l_1/p_1 = l_1$.

Now, let us consider the routing scenario with n paths. The average traceback time $T_a = l_1 + l_2 + \cdots + l_n$, and our goal is to maximize $l_1 + l_2 + ... + l_n$. We still assume that each path can choose a length between l_0 and l_m ($E \leq l_m$).

Theorem 5.1 *Given n routing paths ($l_1, l_2, \cdots, l_n$) connecting the source node s_k and point A, messages can be routed from s_k to point A through any of the paths. Suppose that these n routes do not intersect at anywhere except at point A. The adversary can then detect the message from any path at point A. Once the adversary starts the traceback procedure on one of the n paths, she cannot detect the message signal from the other paths. Let $P = \{p_1, p_2, \cdots, p_n\}$ be the message probability distribution on $\{l_1, l_2, \cdots, l_n\}$ (note $p_1 + p_2 + \cdots + p_n = 1$). Therefore, the adversary's average traceback time $T_a = l_1 + l_2 + \cdots + l_n$. If we have the following energy constraints:*

$$\begin{aligned} &l_0 \leq l_1, l_2, \cdots, l_n \leq l_m, \\ &l_1 p_1 + l_2 p_2 + \cdots + l_n p_n \leq E, \end{aligned} \tag{5.12}$$

the maximum average traceback time $Max(T_a) = (n-1) \cdot l_m + E$.

Proof: We can choose $l_2 = l_3 = \cdots = l_{n-1} = l_m$ and $l_1 = E$. Then $T_a = l_1 + l_2 + ... + l_n = (n-1) \cdot l_m + E$. This can be achieved by distributing all of the flow to l_1 and assigning message probability 0 to $l_2, l_3, \cdots, l_n$. The average traceback time is maximized because there must exist a path with length no greater than E (which is l_1), and all other paths have the maximal length. ■

Now, let us consider another variation of the problem. Suppose we have n fixed routes with fixed length $l_1 \leq l_2 \leq \cdots \leq l_n$, and the adversary chooses any path with equal probability $1/n$, which is the case when the adversary starts its tracing from a random point in the middle of the network. The best strategy for distributing the message flows is to assign probability 1 to l_1 and probability 0 to all other routes, which makes the average traceback time $T_a = (l_1/p_1 + l_2/p_2 + \cdots + l_n/p_n)/n$ to be infinity.

The above analysis states that many routes have to be left unused or used very rarely to maximize the average traceback time. This is true if the adversary does not change position and always waits for the next message on the previous selected traceback path. However, the adversary is normally smarter. Instead of remaining static at one point and waiting for the next message, the adversary may roam around to discover other traceback routes which carry messages more frequently. In case the adversary finds the route that is assigned for message routing with probability 1, the traceback time would immediately be increased to $T_a = l_1$. Therefore, we believe the average traceback time cannot characterize the real scenario. In the next section, we propose a more realistic performance metric: minimal traceback time.

5.4 Max-Min Traceback Time

In the previous section, we have seen that the average traceback time leads to an unreasonable solution and could not characterize the real scenario. Here we propose another more realistic

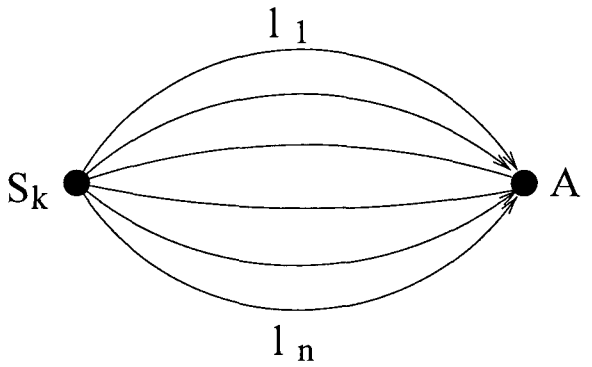


Figure 5.5: *n* routing paths are arranged to be parallel with each other.

performance metric for location privacy: minimal traceback time, which captures the worst case scenario. Routing schemes with good performance in terms of the average traceback time may perform poorly in the worst case. For example, consider the optimal routing scheme for average traceback time described in the previous section. In the worst case, the adversary may pick the shortest routing path with length $l_1 = E$ and message probability $p_1 \approx 1$. The adversary's minimum traceback time is $l_1/p_1 \approx E$. Thus, in the worst case, the optimal scheme performs no better than a single routing path with the length of E.

In the following, we first consider the message routes that are well separated so that they have no common node other than source and sink, then we investigate the splicing routes that are tangled together. For well-separated routes, we consider which routing scheme is optimal given energy consumption constraints. We look at two scenarios: a route can take an arbitrary length and a set of fixed routes, and we find the optimal message flow distribution for them. In the splicing route case, we also look at a set of fixed routes to see how to distribute flows.

5.4.1 Max-Min Trace-back Time for Length-adjustable Routes

In order to maximize the adversary's minimum traceback time, we should avoid following two situations: (1) the majority of messages are routed through minority routes; (2) one or several

routing path lengths are significantly shorter than the rest of the routing paths. Given the same power constraints as in Eq. (5.12), we arrange the n routing paths in the way shown in Fig. 5.5. All routing paths are parallel with each other without any intersection between s_k and A. Since the length of routing paths is adjustable, we let $l_1 = l_2 = ... = l_n = E$. The source node s_k randomly and uniformly distributes the messages to these n routes. Obviously, the adversary's traceback time on all n routing paths is nE. Therefore, the adversary's minimum traceback time under this routing scheme is nE. Now, we show that nE is the Max-Min traceback time.

Theorem 5.2 *Given n routing paths $(l_1, l_2, \cdots, l_n)$ connecting the source node s_k and point A, messages can be routed from s_k to point A through any path. Let $P = \{p_1, p_2, \cdots, p_n\}$ be the message probability distribution for paths $\{l_1, l_2, \cdots, l_n\}$ (note $p_1 + p_2 + \cdots + p_n = 1$). If there are the following energy constraints:*

$$\begin{aligned} &l_0 \leq l_1, l_2, \cdots, l_n \leq l_m, \\ &l_1 p_1 + l_2 p_2 + \cdots + l_n p_n \leq E, \end{aligned} \tag{5.13}$$

the Max-Min traceback time $T_{Max-Min} = nE$.

Proof: For any routing path distribution l_i and p_i, $1 \leq i \leq n$, we want to find $Max\{Min\{\frac{l_i}{p_i}\}\}$. Suppose we have the constraints given in (5.13). Let $a_i = l_i/p_i$, $1 \leq i \leq n$, and the energy constraint can be written as $a_1 p_1^2 + a_2 p_2^2 + ... + a_n p_n^2 \leq E$. Suppose there is a path k $(1 \leq k \leq n)$, $a_k = Min(a_i)$. We have $a_1 p_1^2 + a_2 p_2^2 + ... + a_n p_n^2 \geq a_k p_1^2 + a_k p_2^2 + ... + a_k p_n^2 = (p_1^2 + ... + p_n^2) a_k$. Therefore, $a_k \leq \frac{E}{p_1^2 + ... + p_n^2}$. Since $p_1^2 + ... + p_n^2 \geq \frac{(p_1 + ... + p_n)^2}{n} = 1/n$, so $a_k = Min\{l_i/p_i\} \leq nE$. Finally, $Max\{Min\{\frac{l_i}{p_i}\}\} = nE$. ■

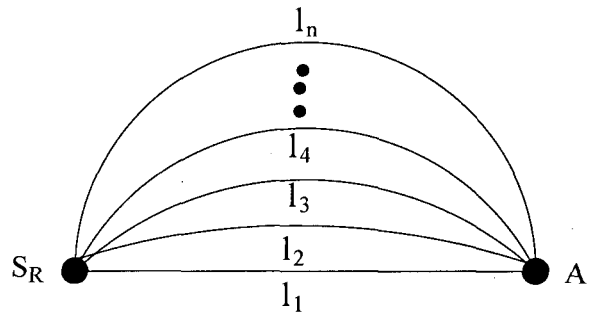


Figure 5.6: n length-fixed routing paths between s_k and A.

5.4.2 Max-Min Traceback Time for Length-fixed Routes

Suppose there are n fixed routes with length $l_1 \leq l_2 \leq \cdots \leq l_n$. They are well separated from each other so that any pair of routes intersect only at the source and the sink. Our goal is to find the optimal message probability distribution $\{p_1, p_2, \cdots, p_n\}$ that maximizes the adversary's minimum traceback time under the energy constraint $l_1 p_1 + l_2 p_2 + \cdots + l_n p_n \leq E$.

As we have discussed in the previous section, for the n routes with the energy constraint E, the *Max-Min* value of the adversary's minimum traceback time is achieved when the traceback time is the same for every path. Likewise, to achieve maximal minimal traceback time, we have to force all the routes to have the same traceback time. If we do not have the energy constraint, a possible solution is to assign the following message distribution: $p_1 = \frac{l_1}{l_1+l_2+\cdots+l_n}, p_2 = \frac{l_2}{l_1+l_2+\cdots+l_n}, \cdots, p_n = \frac{l_n}{l_1+l_2+\cdots+l_n}$. It is a valid message distribution because $p_1 + p_2 + \cdots + p_n = 1$. Now, the corresponding energy consumption becomes:

$$p_1 l_1 + p_2 l_2 + \cdots + p_n l_n = \frac{l_1^2 + l_2^2 + \cdots + l_n^2}{l_1 + l_2 + \cdots + l_n}. \tag{5.14}$$

Therefore, the solution is feasible when the energy consumption in Eq. 5.14 is less than or equal to E. Obviously, if our energy budget is sufficient (satisfies the above condition), this routing scheme maximizes the adversary's minimum traceback time. This can be explained as follows.

The above scheme achieves the same traceback time– $l_1 + l_2 + \cdots + l_n$ on all n routing paths. If we try to increase the traceback time on a specific route i, we need to reduce the amount of messages on route i. Those messages that originally go through route i should be re-distributed to other routes. Then, the route that gets these extra messages will have a larger message probability. As a result, the corresponding traceback time will be less than the original value. Therefore, the traceback time $l_1 + l_2 + \cdots + l_n$ is the optimal value when E is large enough to cover the routing energy expenditure.

However, since our energy budget is usually tight, which means the value of E is less than the value in Eq. (5.14), then how do we distribute the messages? Without loss of generality, for a given E, assume we can find k such that the first k routes satisfy the energy constraint by using the above routing strategy, but the first $k+1$ routes exceed the energy constraint E by using such a scheme. In mathematical expression, we have

$$\begin{aligned} &\frac{l_1^2 + l_2^2 + \cdots + l_k^2}{l_1 + l_2 + \cdots + l_k} \leq E, \\ &\frac{l_1^2 + l_2^2 + \cdots + l_{k+1}^2}{l_1 + l_2 + \cdots + l_{k+1}} > E. \end{aligned} \tag{5.15}$$

If we only use the first k routes, we can achieve the adversary's minimum traceback time as $l_1 + l_2 + \cdots + l_k$. Notice that we have not used up our energy budget yet, so we can do better because we have not used the rest of the $n-k$ routes yet. Imagine we can move a portion of messages from the first k routes to route $k+1$, so that the traceback time for each of k routes increases at the same rate while the total energy consumption just reaches the value of E. If we use T_s to represent the new traceback time for the first k paths, $p_1, p_2, \cdots, p_k$ can be written as

$\frac{l_1}{T_s}, \frac{l_2}{T_s}, \cdots, \frac{l_k}{T_s}$, respectively. Therefore, we have

$$\begin{aligned} \frac{l_1^2 + l_2^2 + \cdots + l_k^2}{T_s} + p_{k+1} l_{k+1} = E, \\ \frac{l_1 + l_2 + \cdots + l_k}{T_s} + p_{k+1} = 1. \end{aligned} \tag{5.16}$$

Combining the above equations, we get

$T_s = \frac{l_1(l_{k+1}-l_1)+l_2(l_{k+1}-l_2)+\cdots+l_k(l_{k+1}-l_k)}{l_{k+1}-E}$. At this time, $p_{k+1} = \frac{T_s-l_1-l_2-\cdots-l_k}{T_s}$, so the adversary's traceback time on route $k+1$ is

$$l_{k+1}/p_{k+1} = \frac{T_s}{(T_s - l_1 - l_2 - \cdots - l_k)/l_{k+1}}. \tag{5.17}$$

Since $l_1 + l_2 + \cdots + l_k < T_s < l_1 + l_2 + \cdots + l_k + l_{k+1}$, $T_s - l_1 - l_2 - \cdots - l_k < l_{k+1}$, the adversary's traceback time on route $k+1$ is longer than T_s. Now, we need to prove that T_s is the optimal solution that we can achieve.

Theorem 5.3 *Let k ($0 \le k \le n$) be an integer such that the following inequalities are satisfied:*

$$\begin{aligned} \frac{l_1^2 + l_2^2 + \cdots + l_k^2}{l_1 + l_2 + \cdots + l_k} \le E, \\ \frac{l_1^2 + l_2^2 + \cdots + l_{k+1}^2}{l_1 + l_2 + \cdots + l_{k+1}} > E. \end{aligned} \tag{5.18}$$

Assume $T_s = \frac{l_1(l_{k+1}-l_1)+l_2(l_{k+1}-l_2)+\cdots+l_k(l_{k+1}-l_k)}{l_{k+1}-E}$. *Then*

$$p_i = \begin{cases} l_i/T_s & 1 \le i \le k \\ \frac{T_s-l_1-l_2-\cdots-l_k}{T_s} & i = k+1 \\ 0 & i > k+1 \end{cases}$$

gives the optimal message probability distribution on all of the routes.

Proof: Assume we have another routing scheme that can achieve a longer value of the adversary's minimum traceback time. Compared with the above scheme, the new scheme should achieve a

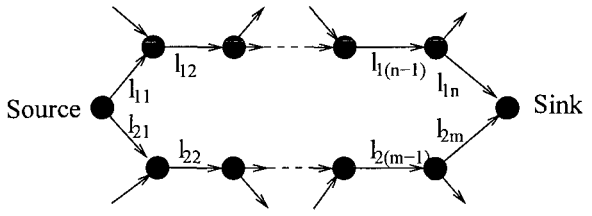


Figure 5.7: A portion of a splicing network.

longer traceback time on *each of* the first k routes. Therefore, the message probability on *each of the* first k routes should be reduced to smaller values, which is equal to "moving" some portion of messages from the first k routes to the rest of the $n-k$ routes. Since we know that the total energy consumption for our proposed scheme is E and in the new scheme we transport message flows from a shorter route to a longer route, the energy consumption for the new scheme will increase, that is, be greater than E, which means that the new scheme violates the energy constraint. Therefore, our proposed scheme is the optimal solution with respect to all the constraints. ■

5.4.3 Max-Min Traceback Time for Splicing Network

In many situations, it is not easy to find and deploy well-separated routing paths such as those in Fig. 5.6 due to sensor field size and the sensor nodes' power constraints. Considering that a long routing path may require a number of remote sensor nodes to participate in the message forwarding task, it is not only a disadvantage in power saving (the operations switching between sleep and active status consumes a lot of power), but also brings about security concerns. Although we disperse our messages into as many routing paths as possible to prevent the possible adversary's traceback, we do want to restrain the messages to a limited area.

A general routing scenario can be shown by a directed graph in Fig. 5.7. Since we are using splicing network routes, the routing scheme is a little bit different from the previous ones. Each node determines which neighbor to send a message to according to some probability. Our goal is

to find the message probability distribution that maximizes the adversary's traceback time in the worst case. As we explained in the previous subsection, the max-min traceback time is achieved when the adversary has the same amount of traceback time on all paths. Note that such an optimum message distribution can be calculated at a centralized node, such as the sink. Since sensor nodes are static, the network topology information can be used to derive the optimum message distributions. Next, we show how to quantitatively determine the message distribution.

As an example, we only focus on two of the routing paths from the source to the sink. Each path is composed of a number of edges. Suppose the upper path (route 1) has n edges, while the lower path (route 2) has m edges. We denote l_{ij} as the length of the jth edge of path i, p_{ij} as the message probability of the jth edge of path i. Therefore, the adversary's traceback time on the upper routing path can be written as $\frac{l_{11}}{p_{11}} + \frac{l_{12}}{p_{12}} + \cdots + \frac{l_{1n}}{p_{1n}}$. Similarly, the traceback time for the lower path is $\frac{l_{21}}{p_{21}} + \frac{l_{22}}{p_{22}} + \cdots + \frac{l_{2m}}{p_{2m}}$. Thus, we have the equation:

$$\frac{l_{11}}{p_{11}} + \frac{l_{12}}{p_{12}} + \cdots + \frac{l_{1n}}{p_{1n}} = \frac{l_{21}}{p_{21}} + \frac{l_{22}}{p_{22}} + \cdots + \frac{l_{2m}}{p_{2m}}. \tag{5.19}$$

Since the edges normally do not change after the sensor network is deployed, the values of length l_{ij} are constants. We only need to determine the message probabilities of the edges. Based on the observation that a message routing graph is very similar to multi-loop electric circuits (considering the message flow as the electric currents, and the edge length as the electric voltage), it is natural to apply Kirchhoff's Rules [42] to solve the message probabilities in the routing graph. First, let us define three terms similar to those in the electric circuits, *junction*, *branch* and *loop*.

Definition 1 *A junction is a sensor node where at least three routing paths meet. The exceptions are the source node and the sink. No matter how many routing paths they are connected to, the source node and the sink are always regarded as junctions.*

Definition 2 *A branch is a routing edge or several serially concatenated edges between two junctions. A branch may consist of several edges because the nodes on the concatenation points are not junctions. In other words, those edges have the same message probability and can be treated as one routing path unit.*

Definition 3 *A loop is composed of two routing paths between a starting junction and an ending junction. Both routing paths begin at the starting junction and end at the ending junction, and they do not intersect at any other junction. Messages can be routed on either path from the starting junction to the ending junction. Each routing path may consist of one or more branches.*

Here, our *loop* is different from a conventional "routing loop", which means the situation where a node receives a message which was previously forwarded by itself. We assume a "routing loop" is prevented in our routing protocol and will never happen. In our routing scheme, messages are always moving forward from the source to the destination. For example, the two routing paths in Fig. 5.7 form a *loop*. Similar to the multi-loop circuit, we can utilize Kirchhoff's Rules to find the message probability for each branch. Here we re-write Kirchhoff's Rules for routing in a splicing network:

Kirchhoff's First Rule: the junction rule. The sum of the message probability coming into a junction is equal to the sum leaving the junction.

Kirchhoff's Second Rule: the loop rule. The adversary's end-to-end traceback time on two paths of a loop is the same.

Based on Kirchhoff's Rules, we can write the junction equations and loop equations by following three steps:

- On the directed routing graph, label the message flow and flow direction in branch;

- Use Kirchhoff's first rule to write down a message probability equation for each junction. In general, if there are J junctions in a routing graph, we need to write $J-1$ junction equations. The equation for the remaining junction is redundant and can be derived from the other $J-1$ equations.

- Use Kirchhoff's second rule to write down loop equations for as many loops as needed to include each branch at least once. To find a loop, we need to pick a starting node and an ending node, then try to find two different paths which both begin and end at these two nodes. At the same time, they do not meet at any third node. When writing the loop equations, we need to make sure equations are independent with respect to each other. A loop equation is guaranteed to be independent as long as there is at least one new branch (that has not previously appeared in other equations) in the loop. In general, if there are B branches and J junctions in a routing graph, in total we will have $B-J+1$ independent loop equations.

Solving the above equations, we can get the optimal message distribution for each path in the splicing network.

5.4.4 Multiple Source Objects

In the previous two sections, we have explored the optimal routing strategies in a network where there is only one data source and one adversary. In the real world, this kind of network model is rare and restricted. One may wonder whether privacy-aware routing is necessary if there are multiple data sources in the network because the routing messages from multiple data sources that may already confuse the adversary. In this section, we extend our discussion to a network with multiple objects. We explain why multiple data sources cannot confuse the adversary's tracing,

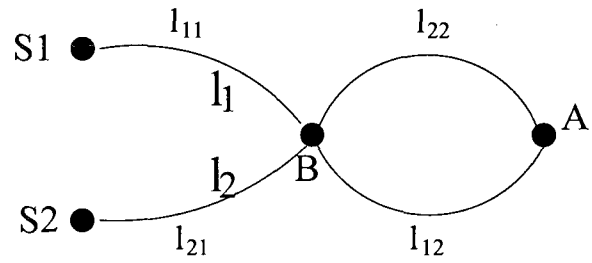


Figure 5.8: Two data sources.

so that the location privacy issue is still valid even with multiple source. Our result can also be applied to mobile objects.

Without loss of generality, we start our discussion with two data objects. If the two objects (under sensing) are located far away from each other and their message routing paths do not intersect at all, it is identical to our single source network model and all of our results can be applied. Therefore, we assume the two routing paths intersect at least once as shown in Fig. 5.8. Suppose that two data sources, s_1 and s_2, send messages to the sink (where the adversary is located) along the routes l_1 and l_2, respectively. l_1 and l_2 intersect at B before they reach the sink.

As discussed in Section 5.3, if s_1 is the only data source, the adversary traceback time to s_1 is l_1. Similarly, the traceback time to s_2 is l_2. If l_1 and l_2 do not intersect, the average traceback time to either s_1 or s_2 is $\frac{l_1+l_2}{2}$ (assume the data rate is the same). Now we examine whether multiple data sources confuse the adversary's traceback, or increase the traceback time. When two routes intersect at B, l_1 is divided into l_{11} and l_{12}, and l_2 is divided into l_{21} and l_{22}. Since the data rate from s_1 and s_2 is the same, the adversary at A has the same probability to detect messages from l_{12} and l_{22}. Therefore, the traceback time from A to B, denoted at T_{AB}, is $\frac{l_{12}+l_{22}}{2}$. Similarly, at point B, the adversary has $\frac{1}{2}$ probability of tracing on either l_{11} or l_{21}. The expected traceback time for the adversary to reach either of the two data sources from B, denoted as T_{BS}, is $\frac{l_{11}+l_{21}}{2}$. In total, the expected traceback time to reach either s_1 or s_2 from A is $T_{AB} + T_{BS} = \frac{l_{12}+l_{22}}{2} + \frac{l_{11}+l_{21}}{2} = \frac{l_1+l_2}{2}$. This

result concludes that the adversary's expected traceback time (to reach either source) does not increase when there are two data sources with intersecting routing paths.

We have studied how multiple sources affect the traceback time to any one of the data sources; now let us focus on the traceback time for a specific data source. We still use the routing example in Fig. 5.8. Without a data source s_2, the traceback time to s_1 is l_1. After s_2 is introduced, the average traceback time from A to s_1 (suppose the adversary takes the route $l_1 1$ at B), denoted as T_{AS_1}, changes to $\frac{l_{12}+l_{22}}{2} + l_{11}$. The difference, denoted as T_{diff}, can be computed as

$$T_{diff} = l_1 - \frac{l_{12}+l_{22}}{2} + l_{11} = \frac{l_{12}-l_{22}}{2}. \tag{5.20}$$

Therefore, after the second data source is introduced, the change of the traceback time to the first source depends on the difference between l_{12} and l_{22}. Note that both l_{12} and l_{22} are routes between B and A. Using the general routing schemes in sensor networks, the length of l_{12} and l_{22} should be very close to each other. T_{diff} thus is approximate to 0. Finally, we conclude that multiple data sources do not help confuse the adversary's tracing and increase the traceback time.

5.5 Privacy-aware Routing Schemes

Inspired by the traceback time analysis for the routing strategies, we discuss two privacy-aware routing schemes in this section. The first routing scheme is called Random Parallel (RP) routing. The strategy is to randomly disperse the source messages into a number of pre-determined parallel routing paths, so that the adversary's traceback progress is deterred due to the fact that the adversary can only perform traceback on a certain routing path. As discussed previously, the pre-determined routing paths are difficult to deploy in a large scale sensor network. Therefore, we propose the second routing scheme, Weighted Random Stride (WRS) routing. WRS routing

allows the messages to be routed in a splicing network, which is more practical and natural for sensor networks and requires only a little deployment information.

5.5.1 Random Parallel Routing

Random Parallel routing is a straightforward privacy-aware routing scheme which is shown in Fig. 5.6. Every sensor is pre-assigned n parallel routing paths starting from that sensor and ending at the sink. We assume the arrangement of these n routes satisfies the energy budget. As we discussed in the previous section, the message distribution strategy at the source node is to give the adversary the same traceback time on any routing path. In particular, when the energy budget is large enough, the message probabilities $p_1, p_2, ..., p_n$ are arranged in such a way that $l_1/p_1 = l_2/p_2 = ... = l_n/p_n$. The adversary traceback time on any path is l_1/p_1.

In RP, any two paths should be well separated so that the adversary cannot detect the message transmission on multiple paths at the same time. In practice, the message routing should be restricted to a small area due to the power constraint and security concerns. For simplicity, we use a rectangular routing zone for each sensor. Once the size of the rectangular routing zone is fixed, the number of routing paths and their lengths can be determined. As a result, the message distribution probability for each random parallel path can be determined during the deployment. The main advantage of RP routing is that the messages can be evenly and well dispersed in the designated routing zone to deter the adversary's traceback progress. However, the RP routing method itself reveals the approximate location of the source node to the adversary. Suppose the adversary starts at the sink; he can quickly identify the direction of the source node by only tracing back several messages on any one of the routing paths. Since all routing paths are parallel, the direction of any routing path will lead the adversary to quickly locate the source node. Another disadvantage

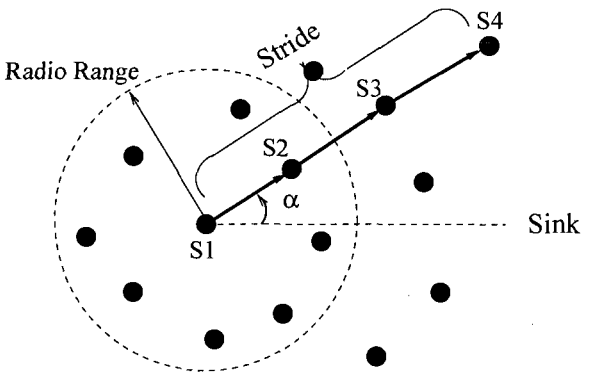


Figure 5.9: Weighted Random Stride routing scheme.

of RP routing is that each sensor has to have global routing path knowledge because the parallel paths are different for different source nodes.

5.5.2 Weighted Random Stride Routing

The intuition of the Weighted Random Stride (WRS) routing scheme is based on the MAX-MIN rule in the splicing network, as discussed in the previous section. The goal is to give the adversary the same traceback time on different tracing paths between any two sensor nodes in the network. As we discussed previously, given the network global topology, we can apply Kirchhoff's Rules to derive the message distribution for every routing path. In practice, however, it is very difficult to derive the results for a large scale sensor network due to a number of restrictions. For example, the global topology of sensor locations is very hard to get, and the topology itself also changes a lot over time due to the nature of wireless links. We propose an efficient, light-weight, yet robust WRS scheme to approximately achieve the above goal. The design of the WRS routing scheme considers the fact that that sensor network is a splicing network. Instead of distributing the messages to a number of fixed parallel paths as described in RP, WRS scheme allows each individual sensor to make the routing decision locally and independently, with very little deployment information.

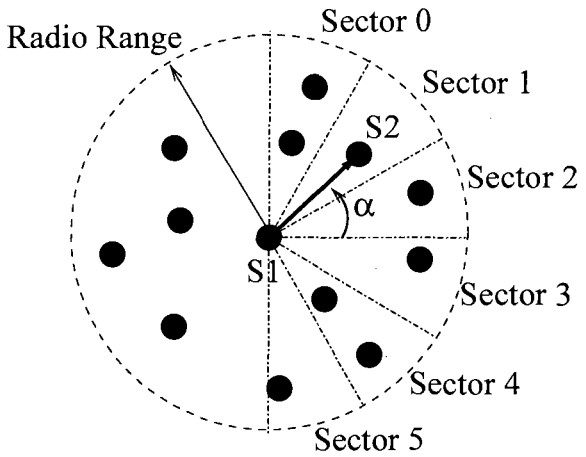


Figure 5.10: Pick the next hop with weighted probability.

To ease the explanation, we use the example shown in Fig. 5.9 to describe WRS routing. There are two parameters specified in message routing: the forwarding angle and the stride. The forwarding angle is the angle between the projected forwarding route and the line connecting the forwarding node and the sink. When a sensor node S_1 transmits a message to the sink (here S_1 can be either a source node or an intermediate forwarding node), it first randomly picks a forwarding angle α, and selects the neighbor S_2 (matching the forwarding angle) as the next hop. The stride is defined as the number of hops associated with the forwarding angle selected by the transmitting node S_1. In this example, S_1 selects the stride value 3. When S_2 receives the message from S_1, it notices that the stride is not finished yet, so S_2 picks its neighbor S_3 as the next hop since S_3 fits the forwarding angle. This procedure continues until the message reaches S_4. S_4 finds that the stride is finished, so it randomly picks another forwarding angle and starts a new stride.

It is not difficult to see that a larger forwarding angle leads to a potentially longer routing path. Therefore, different forwarding angles should be picked with different probabilities. In WRS, nodes are arranged to pick a larger forwarding angle with a higher probability. In this way, more messages will be distributed to longer paths so as to deter the adversary's traceback. For practical

reasons, we do not require the node to store all forwarding probabilities for every different angle. Instead, we make the following arrangement as shown in Fig. 5.10 to simplify the procedure. We divide the right half-disc of the node radio coverage (suppose the sink is on the right side, so the node always picks the next hop that is located in the right half-disc) into a number of sectors (six in our example). Now, we randomly pick a sector instead of an angle. Once a sector is picked, the forwarding node selects its neighbor in the corresponding sector that makes the largest forwarding step. Similarly, the probability of selecting the sectors is different. For the example as shown in Fig.5.10, sectors 0 and 5 are most likely to be picked, while sectors 2 and 3 have the lowest probability. In our simulation below, the probability of selecting sector 0 and 1 is three times and twice of that of selecting sector 2, respectively.

5.5.3 Evaluation

To evaluate the proposed the privacy-aware routing schemes, we implement both RP routing and WRS routing in our customized simulator. For the purpose of comparison, we implement a baseline Random Walk (RW) routing scheme which is adopted by Phantom routing [74, 53]. In RW routing, the forwarding node randomly and uniformly picks one of its neighbors as the next hop. To make sure the messages will finally reach the sink, each intermediate node always forwards to the neighbor that is closer to the sink.

5.5.3.1 Simulation Setup and Metrics

We deploy a large scale sensor network. Sensors are randomly and uniformly distributed in the sensor network. The radio transmission range of the sensor is fixed at 10 m. On average, each sensor has about 20 neighbors. Due to power constraints, message routing should be restricted in

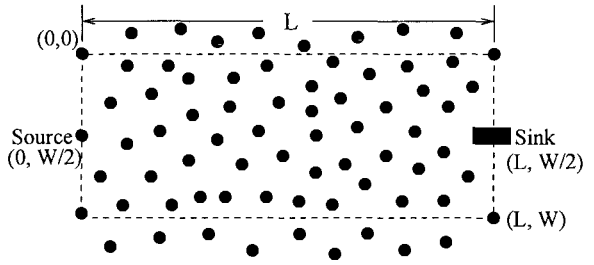


Figure 5.11: A rectangular routing zone: the length L is the distance between the source node and the sink, W is the width.

a routing zone. As shown in Fig. 5.11, we assign a rectangular routing zone for the source node. All messages transmitted from the source node should be confined in the rectangular area. The length of the field, L, is the distance between the source node and the sink. In the simulation, we fix L to be 800 m. W is the width of the field. The value of W is determined by the energy budget in the network. In the simulation, we change the width from 200 m to 600 m for comparing the performance under different energy budget setups.

Once the width of the routing zone is determined, the routing paths in RP routing can be fixed. In the simulation, we arrange any two adjacent routing paths in RP routing to be separated from each other by 20 m so that the adversary can only trace the message on one routing path as long as his radio detection range is no more than 20 m.

In the simulation, we fix the message rate of the source node at a fixed value, so that we use the number of messages as the metric to measure the adversary traceback performance. We record the number of messages the source node has sent until the adversary successfully locates the source node. There is only one adversary in the simulation. Two radio detection ranges, 10 m and 20 m, are considered.

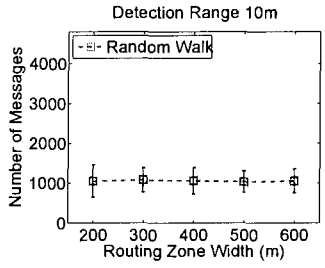


Figure 5.12: The adversary's traceback time with Random Walk routing, when the detection range is 10 m.

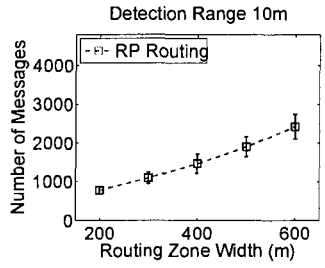


Figure 5.13: The adversary's traceback time with Random Parallel routing, when the detection range is 10 m.

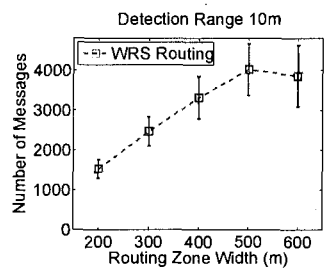


Figure 5.14: The adversary's traceback time with WRS routing, when the detection range is 10 m.

5.5.3.2 Simulation Results

We perform the first set of adversary traceback simulation, with the adversary detection range of 10 m, for RW, RP and WRS routing, respectively . The routing zone length (the distance between the source node and the sink) is fixed at 800 m. The width is changed from 200 m to 600 m for different energy budget. In the simulation, the adversary always starts tracing from the sink. Once the adversary detects a message transmission, he immediately moves to the location of the transmitting node and waits for the next detection. The traceback ends as soon as the adversary successfully reaches the source node. For each test, the adversary successfully performs traceback for 1000 times. We record the average traceback time (in term of the number of messages) and the standard deviation.

The result of the adversary traceback performance is illustrated in Fig. 5.12, Fig. 5.13 and Fig. 5.14, respectively. Fig. 5.12 clearly shows that the privacy preservation characteristic of RW routing does not change when the routing zone width changes. The adversary traceback time stays around 1000 messages when the routing zone width expands from 200 m to 600 m. This phenomenon indicates that pure random walk routing is independent of the routing zone size. The random walk scheme is not aware of the routing zone change and cannot exploit the extra energy

budget to prevent the adversary's traceback.

In comparison, the traceback time in RP routing increases as the routing zone becomes larger. The reason is that the routing paths are well dispersed in RP routing. When the zone size increases, the source node will have more routing paths to which to distribute the messages. Therefore, the adversary has less probability of detecting the message at a specific location, so that the traceback time is longer. Fig. 5.13 demonstrates that the adversary consistently needs more messages to perform a trace as the routing zone width increases. Given the exact same routing zone width changing from 200 m to 600 m, the adversary traceback time increases linearly from 775 messages to 2424 messages, a much better performance than RW routing.

In WRS routing, we set the stride value to 5. Similarly to Fig. 5.10, each node has 6 forwarding sectors, the probability ratio of selecting the forwarding sector is 3:2:1, which means the probability of choosing sector 0 and 5 is three times more than that for sector 2 and 3. Differently from the RP routing scheme, WRS allows most of the sensor nodes in the routing zone to participate in the message forwarding. Recall that there are a fixed number of routing paths in RP routing, so the number of participating sensor nodes is limited to those on the routing paths. Therefore, WRS routing yields better traceback time performance than RP routing because the adversary is more confused by many more forwarding sensor nodes from different directions. As we can see in Fig. 5.14, the adversary has to spend more time to successfully determine the the source node location. When the zone width is between 200 m and 500 m, it takes more than twice the traceback time as in RP for the adversary to locate the source node. One may notice that the traceback time decreases when the routing zone width changes from 500 m to 600 m. We call this phenomenon saturation. In our simulation, we find that saturation happens when the zone width is around 500 m. The reason is that the messages cannot reach the additional area when the zone

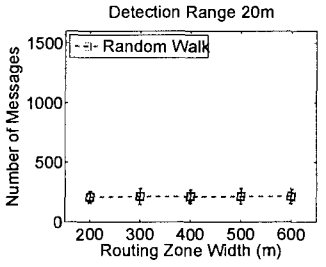


Figure 5.15: The adversary's traceback time with Random Walk routing, when the detection range is 20 m.

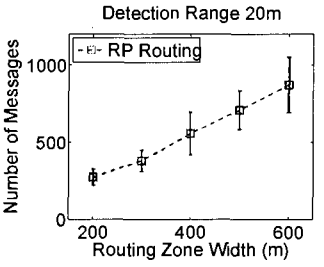


Figure 5.16: The adversary's traceback time with parallel routing, when the detection range is 20 m.

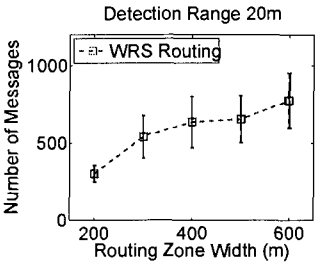


Figure 5.17: The adversary's traceback time with random routing, when the detection range is 20 m.

width increases from 500 m to 600 m. In other words, WRS cannot take advantage of the extra energy budget under this situation. We argue that the energy budget is normally very tight so that the chance of saturation is very rare.

In the second set of traceback simulation, the adversary's detection range is doubled to 20 m. Fig. 5.15, Fig. 5.16 and Fig. 5.17 illustrate the adversary's traceback performance with 20 m detection range. As we can see, compared to the first set of results, the traceback time in RW and WRS routing reduces more than four times. The reason is that the adversary's effective detection area size increases quadratically when his detection range extends linearly. Interestingly, we find the adversary traceback time in RP routing does not reduce as much as that in RW and WRS. Recall that we intentionally arrange the routing paths to be separated for approximately 20 m from each other in RP. When the adversary's detection range increases from 10 m to 20 m, the adversary can detect the messages on at most three consecutive paths. That explains why the traceback time in RP reduces by about three times when the adversary detection range becomes 20 m.

As explained in Section 5.4, many times the minimal traceback time is more critical and practical. Finally, we examine the worst case traceback time for the three routing schemes when

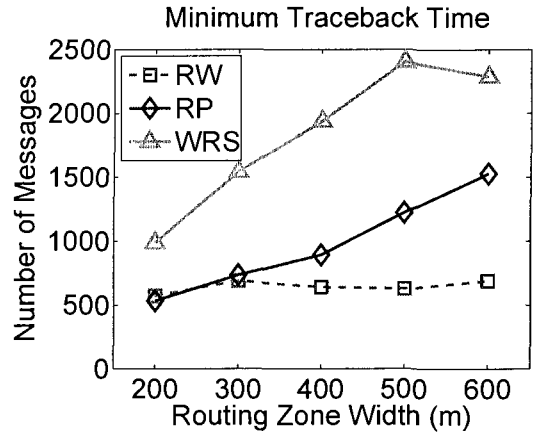


Figure 5.18: The adversary's minimum traceback time with the detection range of 10 m.

the adversary's detection range is 10 m. Among the 1000 adversary's traceback simulation, we pick the fastest traceback and plot the figure shown in Fig. 5.18. As we can see, RW routing has the worst performance in the worst case. It only takes 570 messages for the adversary to reach the source location. When the routing zone width increases to 600 m from 200 m, this number increases only slightly to 688 messages. Interestingly, RP routing has similar worst case performance as that of RW when the routing zone size is small. However, with the routing zone size enlarged, the worst case traceback time increases quickly. For example, when the width is broadened to 400 m from 200 m, the worst case traceback time increases to 890 from 531 messages. Compared to RW and RP routing, WRS achieves the best worst case performance as expected. When the routing zone width is within 200 m to 500 m, the worst case traceback time increases from 985 messages to 2406 messages, about twice the number of messages in RP. Again, saturation happens when the width becomes 300 m, and the minimum traceback time is moderately reduced to 2287 messages, which is still much higher than RP.

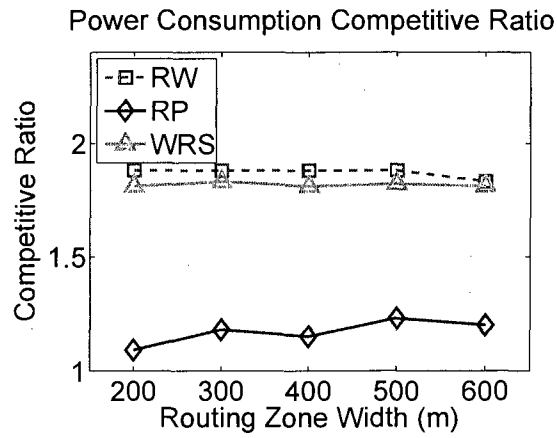


Figure 5.19: The power consumption comparison among *RW*, *RP* and *WRS* routing schemes.

5.5.4 Power Consumption Overhead

Both the *RP* and *WRS* routing protocols improve location privacy by dispersing the messages into different routing paths. Compared with message routing in the greedy shortest path routing normally used in sensor networks, the messages in RP and WRS travel a longer distance (or more hops) and therefore consume more energy. Now, we investigate the power consumption overhead in both privacy-aware routing schemes.

Since the amount of energy consumption is proportional to the number of hops in the routing path, we denote $C_p = L_p/L$ as the power consumption competitive ratio of the privacy-aware routing scheme to shortest path routing, where L is the distance (or hop counts) between the source node and the sink, and L_p is the average routing path length in the specific routing scheme, either *RW*, *RP* or *WRS*.

We run the simulation for all three routing schemes: *RW*, *RP* and *WRS*, as well as the shortest-path routing scheme as the base scheme. We continue to use the rectangular sensor field with length of 800 m and the width changing from 200 m to 600 m. In each of above simulation, 1000 messages are routed from the source to the sink, the average number of hops are recorded, and

corresponding power consumption competitive ratios are presented in Fig. 5.19.

It is not a surprise to see that all three privacy routing schemes consume more energy than the base shortest path scheme. What surprises us is that *RW* has a larger power consumption overhead than *RP* and *WRS*, while its anti-traceback performance is much worse (as we discussed previously). The reason can be explained as follows. In *RW*, each forwarding node equally and randomly selects one of its neighbors (who have a shorter distance to the sink) as the next hop, so the next hop node may not be the one (among the neighbors) that is closest to the sink. As a result, the message forwarding efficiency could be low because it may cost two hops to forward a message which otherwise could be directly routed in just one hop.

Comparatively, the power consumption overhead in *RP* is very small, just 23% more than the base routing scheme. At the first glance, *RP* seems more appealing due to the advantage of its low power consumption overhead. However, as we discussed in Section 5.5.1, *RP* is not suggested for practical sensor deployment because all routing paths are parallel with each other, so the routing paths in *RP* and the corresponding source node location may be easily derived by an adversary after collecting initial network traffic activities.

The *WRS* scheme, on the other hand, has a larger power consumption overhead and needs around 82% over the base scheme. In fact, the energy overhead of *WRS* is the trade-off for location privacy. Given the location privacy protection performance of increasing the adversary traceback time from 10 to 40 times (for the corresponding network settings), we believe the approximately 82% energy overhead is a good price for the privacy.

5.6 Adversary Sensor Network

In this section, we extend our discussion to an extreme adversary model. Instead of placing a certain number of monitoring subjects, the adversary is able to deploy a sensor network to monitor the activities of the sensors in any location in the network. The adversary network is not purposed to detect what our network is monitoring, but it is interested in what assignment our network is involved with and in particular the location of the object that is our network's concern. In this scenario, the adversary is extremely powerful in identifying the monitored object by profiling the network communication activities and analyzing and mining the spatio-temporal relationship among all network communications.

We observe that all of the sensors should transmit their packets at the same rate to prevent the adversary network from detecting any anomaly that may be identified as the data source or the monitored object. Any node (or location) exhibiting more messages in a period encourages close scrutiny and is exposed to a risk of disclosing the monitored object. The solution we propose in this section is to regulate the sensor message transmission rate in a controlled way so that each node (or location) cannot be distinguished by examining the message rate in a period. Each sensor has a scheduled time slot to transmit a fixed amount of messages during a predefined period. In the next period, the sensor will transmit again in the same scheduled time slot. If the sensor has a data message to transmit or relay, it has to wait for its time slot. Otherwise, the sensor still needs to transmit dummy messages if no data messages are available. In this way, all of the sensors have the same message transmission rate in a period. Again, the transmitted messages are all encrypted in a certain way so that the adversary is not able to know the content of any message, but the recipient of the next hop sensor knows a message is destined to it by listening to the message

head.

We assume that the clocks on each sensor are well synchronized so that they agree on the message transmission schedule. The scheduled time slot for transmission is a pseudo-random function of the node ID so that each node knows the scheduled transmission slot for any node. Our goal is to design a routing strategy to route messages from the source to the sink with average message delay under the constraints of the controlled transmission schedule. Our algorithms are centralized, assuming that network topology is known to the node who calculates the routing assignment.

5.6.1 Problem

For easy exposition, we assume the data messages are generated at the same time in a bursty fashion. Our algorithm can be easily extended to the case that messages are generated at a certain rate. Our goal is to distribute those messages to the sensors in proximity so that the total delay that those messages go through is minimized. Suppose the source is labeled as "0" and the sink is labeled as "n". Strictly speaking, the source is not a sensor, instead it is a conceptual node for easy explanation. The source node connects to the sensors that are in its proximity and can monitor the source for data generation. Since it is a dummy node, we assume the source can send data to the nearby sensors without capacity or rate constraints.

Assume every sensor sends one message per T time units. Let $t_i = f(i)$ be the schedule transmission slot of node i, where $t_i \in [0, T)$ and f is a pseudo-random function. Node i will send a message at time t if $t \equiv t_i (\text{mod } T)$. We define d_{ij} as the delay at j if i sends a message to j directly,

$$d_{ij} = (t_j + T - t_i) \text{ mod } T.$$

The network is modeled as a graph $G(V,E)$, where each edge ($e_{ij} \in E$) connects two nodes ($i,j \in V$) within the communication range. We assign d_{ij} as the weight to edge e_{ij}. Let 0 and n be the labels of the source and destination of the messages respectively.

$$\begin{aligned} d_{0j} &= (t_j + T - t_{start}) \bmod T, \text{for any edge } e_{0j}, and \\ d_{in} &= 0, \text{for any edge } e_{in} \text{ connected to the sink,} \end{aligned}$$

where t_{start} is the starting time for the source to generate data messages. Our goal is to find routing paths that deliver messages from the source to the sink with the minimum average delay, i.e., the total delay of all messages. It is evident that sending one message with the minimum delay is equivalent to finding the shortest path from the source to the destination in the weighted graph G. In the following, we investigate how to route multiple messages.

5.6.2 Multiple Messages

If we have $k > 1$ messages to send, one solution is to send all of them through the shortest path. However, due to the schedule constraint, every message arrives at the destination T time later than the previous message. There may exist a more efficient solution, which uses multiple paths, instead of repeatedly using the shortest path, to relay the k messages. A solution S of this problem consists of a set of paths $P = \{p_1, p_2, \cdots, p_m, m \leq k\}$ and the corresponding message loads on the paths $M = \{M_1, M_2, \cdots, M_m\}$. In order to avoid message collision, the paths in our solution are node disjoint. Our objective is to minimize the average/total delay of all messages. In other words, our goal is to find a set of disjoint paths and assign message loads to each of them, such that the total delay can be minimized.

Our algorithm is shown in Algorithm 5. We aim to find L node-disjoint routes to transmit

messages. During every time slot of T, we inject one message to each of these L routes. Let l_i be the length of route p_i. The total delay of this strategy can be expressed as

$$\begin{aligned} & \sum_{i=1}^{L} l_i + (\sum_{i=1}^{L} l_i + L \cdot T) + (\sum_{i=1}^{L} l_i + L \cdot 2T) + \cdots \\ = \; & \frac{k}{L}(\sum l_i + \frac{kT}{2}) - \frac{kT}{2}, \end{aligned}$$

where $\sum l_i$ is the length summary of all selected paths. Let SN be the set of nodes within communication range of the source, $SN = \{j | e_{0j} \in E\}$. In Algorithm 5, we enumerate all of the possible number of routes in the outer loop, which is upper-bounded by $|SN|$. For each value of L, we find a set of L node disjoint paths, such that $\sum l_i$ is minimized. This problem is equivalent to the minimum k node-disjoint paths problem in graph theory. The existing algorithms, e.g., [86, 87, 6, 85], can be applied to our problem. After checking all possible values of L, we finally obtain a solution with the minimum total delay, which is stored in variable opt.

Algorithm 5 Find the optimal solution

for $L = 1$ to $|SN|$ **do**

  Find L node disjoint paths that the total length is minimized

  $min =$ total length of L paths

  if $\frac{min + \frac{kT}{2}}{L} < opt$ **then**

    $opt = \frac{min + \frac{kT}{2}}{L}$

    $L' = L$

  end if

end for

$opt = opt \cdot k - \frac{kT}{2}$

In the following, we show the performance of the approximate algorithm. We use $\{\bar{P}, \bar{M}\}$ to

represent our solution, where the route set $\bar{P}$ is obtained by the k node disjoint path algorithm and the message load on each route is the same, i.e., $\bar{M_i} = \frac{k}{L'}$, where L' records the value of L yielding the optimal solution. We use a function $D(P,M)$ to denote the total delay of solution $\{P,M\}$. In our algorithm, $opt = D(\bar{P},\bar{M})$. Let $\{P^*,M^*\}$ be the optimal solution. In the following, we compare our solution with the optimal one and show opt is very close to $D(P^*,M^*)$.

Let $P^* = \{p_1, p_2, \cdots, p_{L_{opt}}\}$ and $M^* = \{M_1, M_2, \cdots, M_{L_{opt}}\}$, where L_{opt} is the number of routes used in the optimal solution. Let l_i be the length of p_i. The total delay of $\{P^*,M^*\}$ is

$$\begin{aligned} D(P^*,M^*) &= \sum_{i=1}^{L_{opt}} \sum_{j=1}^{M_i} l_i + (j-1)T \\ &= \sum_i (M_i l_i + \frac{M_i(M_i-1)}{2} T) \\ &= \sum_i M_i l_i + \frac{T}{2} \sum_i M_i^2 - \frac{kT}{2} \end{aligned} \tag{5.21}$$

For each path p_i, the delay of the last message is $l_i + T(M_i - 1)$. We can prove the following lemma.

Lemma 5.1 *For any two distinct paths $p_i \in P^*$ and $p_j \in P^*$,*

$$|(l_i + T(M_i - 1)) - (l_j + T(M_j - 1))| \leq T.$$

Proof: Proof is omitted due to page limits. ■

Corollary 1 *For any two distinct paths $p_i \in P^*$ and $p_j \in P^*$,*

$$\frac{l_j - l_i}{T} - 1 \leq M_i - M_j \leq 1 - \frac{l_j - l_i}{T}.$$

Let p_{min} be the path with the minimum length, i.e.,

$$l_{min} \leq l_i, i \neq min.$$

Accordingly, the message load on p_{min} is denoted as M_{min}. We can prove the following lemma

Lemma 5.2 *M_{min} is the maximum among the optimal message loads of all paths,*

$$M_{min} \geq M_i, i \neq min.$$

Proof: Proof is omitted due to page limits. ■

Now, let us consider another solution, where the route set is the same as P^*, but the message load on each route is the same. We use M' to indicate this message distribution, i.e., $M'_i = \frac{k}{L_{opt}}$. The following lemma shows the performance of this solution.

Lemma 5.3 *When k is large,*

$$D(P^*, M^*) > (1 - \frac{L_{opt}(l_{max} - l_{min})}{kT}) D(P^*, M'),$$

where l_{max} and l_{min} are the longest and shortest path in P^* respectively.

Proof: According to Corollary 1, $M_i \leq M_{min} - \frac{l_i - l_{min}}{T} - 1$. Recall Eq.(5.21), the first term is

$$\begin{aligned} \sum_i M_i l_i &\geq \sum (M_{min} - 1 - \frac{l_i - l_{min}}{T}) l_i \\ &= (M_{min} - 1) \sum l_i - \sum \frac{l_i - l_{min}}{T} l_i. \end{aligned}$$

Since M_{min} is the maximum message load, it must be greater than the average load $\frac{k}{L_{opt}}$. Therefore,

$$\sum_i M_i l_i \geq \frac{k}{L_{opt}} \sum l_i - \frac{l_{max} - l_{min}}{T} \cdot \sum l_i,$$

where l_{max} is the longest path among the path set. Thus,

$$\begin{aligned} D(P^*,M^*) &\geq \frac{k}{L_{opt}}\sum l_i - \frac{l_{max}-l_{min}}{T}\sum l_i \\ &\quad + \frac{T}{2}\sum_i M_i^2 - \frac{kT}{2}. \end{aligned}$$

Since $\sum M_i = k$, we know

$$\sum_i M_i^2 \geq \sum(\frac{k}{L_{opt}})^2 = \frac{k^2}{L_{opt}}.$$

Therefore, we have

$$\begin{aligned} & D(P^*,M^*) \\ \geq\ & \frac{k}{L_{opt}}\sum l_i + \frac{T}{2}\sum_i(\frac{k}{L_{opt}})^2 - \frac{kT}{2} \\ & - \frac{l_{max}-l_{min}}{T}\sum l_i \\ =\ & D(P^*,M') - \frac{l_{max}-l_{min}}{T}\sum l_i \\ =\ & D(P^*,M') \\ & - \frac{l_{max}-l_{min}}{T}\frac{L_{opt}}{k}(D(P^*,M') + \frac{kT}{2} - \frac{k^2T}{2L_{opt}}) \\ =\ & D(P^*,M') - \frac{l_{max}-l_{min}}{T}\frac{L_{opt}}{k}D(P^*,M') \\ & + \frac{l_{max}-l_{min}}{2}(k-L_{opt}) \\ >\ & (1 - \frac{l_{max}-l_{min}}{T}\frac{L_{opt}}{k})D(P^*,M'). \end{aligned}$$

■

Recall

$$\begin{aligned} D(P^*,M') &= \sum M_i' l_i + \frac{T}{2}\sum(M_i')^2 - \frac{kT}{2} \\ &= \frac{k}{L_{opt}}(\sum l_i + \frac{kT}{2}) - \frac{kT}{2}, \end{aligned}$$

the value of the total delay only depends on $\sum l_i$ and L_{opt}. In Algorithm 5, we enumerate all possible values of L, which include L_{opt}, and try to minimize $\sum l_i$. Thus, opt in Algorithm 5 must be no more than $D(P^*, M')$, i.e.,

$$\begin{aligned} opt &\leq D(P^*, M') \\ &< \frac{kT}{kT - L_{opt}(l_{max} - l_{min})} D(P^*, M^*). \end{aligned}$$

Therefore, when k is large, our solution is very close to the optimal solution in terms of total message delay.

5.6.3 Evaluation

To defend against traffic monitoring by the adversary sensor network, all sensors have to transmit messages periodically (in T time units) as long as there is a message to be delivered to the sink. As a result, message delivery becomes a very energy consuming task. Therefore, we want to keep the message delivery time as short as possible. In this subsection, we examine the efficiency of our proposed L-disjoint path message delivery solution through simulation.

We set up a rectangular sensor network similar to that presented in Fig. 5.11, with a length of 800 m and a width of 200 m. Once the sensor network is deployed, the sink can calculate the optimal routing solution as we proposed for each sensor node. We assume each sensor node receives the routing provisioning from the sink, so that there is no processing delay while routing the message from a specific source node to the sink (the routing path is predetermined).

In the simulation, we measure the total amount of time for the source node to successfully deliver various numbers of messages to the sink. Note that the message delivery time here is different from the total delay we discussed in the previous subsection (which is solely for simplifying the

analysis). Here, the time is the real world time delay for the source node to deliver the messages to the sink.

We randomly and uniformly deploy 10,000 sensor nodes in the rectangular sensor field. We run the algorithm presented in the previous subsection and find a total of $k = 16$ paths. The length of the 16 paths is shown in Table 5.1.

Path	1	2	3	4	5	6	7	8
Hops	87	88	89	89	89	89	89	89
Path	9	10	11	12	13	14	15	16
Hops	90	90	90	90	91	91	92	93

Table 5.1: Length of shortest 16 paths between the source and the sink.

Given the 16 routing paths, we estimate the time delay for the source node to deliver various numbers of messages to the sink. For simplicity, we set T to one second. Each sensor node is allowed to transmit either a real message or a dummy message in one second. For example, as shown in Table 5.1, the shortest path between the source and the sink is 87 hops. It thus takes 87 s for the source node to transmit one message to the sink. Now, we compare the message delivery time given a different number (k) of paths, and plot the results in Fig. 5.20.

As we can see, when there is only one message to be sent, the message delivery time is the same for different k. As the number of messages increases, however, we start to notice a difference in time delay. Considering that we have 10 messages to deliver, if $k = 1$, all messages have to be sent through the only path; it therefore takes 9 extra time cycles to delivery 10 messages, for a total of 96 s. If $k = 2$, the source node sends five messages to one of two paths, so the total delay is 92 s. We can get the results for other three cases in a similar fashion.

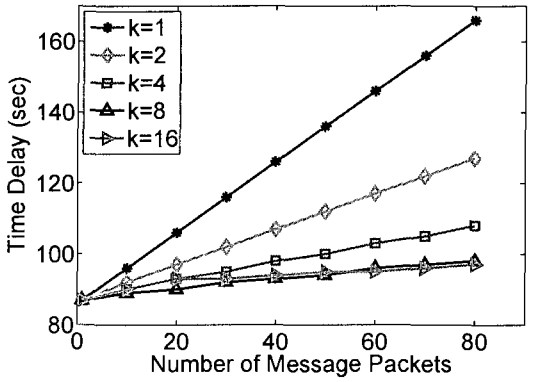


Figure 5.20: Time delay for delivering various number of messages from the source node to the sink, given a different number of paths.

Interestingly, we notice that the time delay for $k = 16$ is larger than that of $k = 8$ when the number of messages is less than 50. The reason is that, as we can find in Table 5.1, the longest path length of 16 paths is 93 hops, while the longest path length of 8 paths is only 89 hops. As we discussed in the previous section, our algorithm assigns the same message load to each path, so that the longest path in $k = 16$ takes an extra 4 cycles to deliver a message compared to the longest path in $k = 8$. As a result, the time delay for $k = 16$ is larger when the number of messages is small. The advantage of $k = 16$ starts to show when the number of messages is more than 60.

Overall, multi-pathing does help to reduce the message delivery time, which in turn reduces the energy consumption of the sensor network. However, it does not mean more paths will bring more benefits. If k becomes larger, the longest path length may be very long, which could increase the message delivery time. As shown in Fig. 5.20, the message delivery time for 8 paths and 16 paths is very close. 16 paths do not bring significant benefit over 8 paths.

5.7 Conclusion

In this chapter, we focus on the location privacy problem in sensor networks. We formulate the problem as an optimization problem in terms of the average traceback time and minimal traceback time for the adversary to reach the message source starting from the sink. We show that the traceback time is related to the number of sensor nodes involved in routing. We give routing strategies to maximize the average and minimal traceback time for a set of fixed routes. Based on it, we propose the WRS, a privacy-aware routing protocol. Our simulation results show that WRS significantly hampers the adversary's traceback progress compared with the Random Walk scheme. We also extend the adversary model to a more powerful one in which an adversary sensor network is deployed to monitor our sensor network communication activities. We show an approximation algorithm to route messages with minimal average delay.

Chapter 6

A Search Engine for the Physical World

Wireless sensors have grown beyond motes scattered off a plane to collect environmental data. Sensors today can be found on diverse objects such as buildings, cars [47], and even clothing [35]. As sensors become more ubiquitous in our environment, their roles can extend beyond environmental sensing, to become an electronic representation of different objects. A sensor attached to a folder for example, can contain a short summary of the contents of the folder. Information once scribbled onto post-it notes and stuck to the folder can also be stored directly onto the sensor itself. These sensors, which are considered as representatives of the physical objects they are attached to, naturally form a database of the physical world. The new techniques for searching information about a particular object become necessary.

Information retrieval (IR) has been widely used to search for information[1] within databases. People can use search engines like Google, to easily find the remote web pages. However, since the physical objects are disconnected from the cyberspace, searching for information in the physical world is more difficult. For example, a college student can easily search and view the Shakespeare

[1]Although information may be in many different types, this dissertation only focuses on the most popular textual information search.

manuscript on the Web in several mouse clicks, but may have spend hours to find his notebook for an exam. The above observation motivate us to develop an information retrieval system for the physical world.

A straightforward system design is to maintain a central database, and let each sensor return its location and data to this database. The user will query the database to find a particular sensor. However, since the data in an object can change, frequent updates to the database are needed. This poses a scalability issue when the number of sensors increase, since database will be unable to support large numbers of simultaneous sensor updates. The alternative design of broadcasting a user query to all the sensors instead of maintaining a database can eliminate the cost of frequent updates. Upon receiving a query, each sensor will determine whether its data matches the query and then decide whether to respond. The user will collect the responses to determine the most suitable answer. However, the communication costs of delivering the query to all sensors is very high when there are large number of sensors. Furthermore, a sensor is unaware of the answers provided by its peers, and hence cannot accurately determine whether its own answer is best suited for the query. As a result, the user has to sift through a large number of responses to determine a suitable answer.

We present Snoogle, an information retrieval system built on low-cost wireless sensor networks. In the pervasive computing environment, Snoogle serves as the search engine and helps people to search physical objects at their vicinity.

6.0.1 Challenges

While IR in Internet is well-established, adopting IR within a sensor network poses several unique challenges. First, a sensor network has to limit communication to conserve power. Internet search

engines can have spiders that continuously crawl the Internet for data. Large amounts of data can be collected and stored in a depository for further processing. Sensor networks do not have this luxury so that the data collection techniques in the search engines for WWW cannot be applied directly. Novel data storage and collection techniques are necessary to overcome such limitations. Second, when we consider sensors being attached to physical objects, these sensors can be mobile and the stored data can change rapidly. Most web page locations on the other hand are comparatively more static even though the content could be very dynamic. This makes maintaining up-to-date information in a sensor network more challenging. Third, security and privacy are bigger challenges in sensor network search than Internet search. People may choose to not have a web page, or not to update it frequently. However, since sensors attach to physical objects like clothes, a user may have several sensors they are not aware of. Furthermore, sensors will always have less resources compared to web servers, making implementation of security even more challenging. In this chapter, we focus on reducing the communication cost, and addressing security and privacy concerns.

6.1 System Design

6.1.1 System Components

Snoogle consists of three main components: object sensors, Index Points (*IP*s) and Key Index Points (*KeyIP*s). An object sensor is a mote attached to a physical object, and contains a textual description of the physical object. The object sensor can be either static or mobile, depending on whether the attached physical object is stationary or moving. Snoogle does not require object sensors to be homogeneous. Object sensors can be as powerful as an iMote [70] or MICAz mote

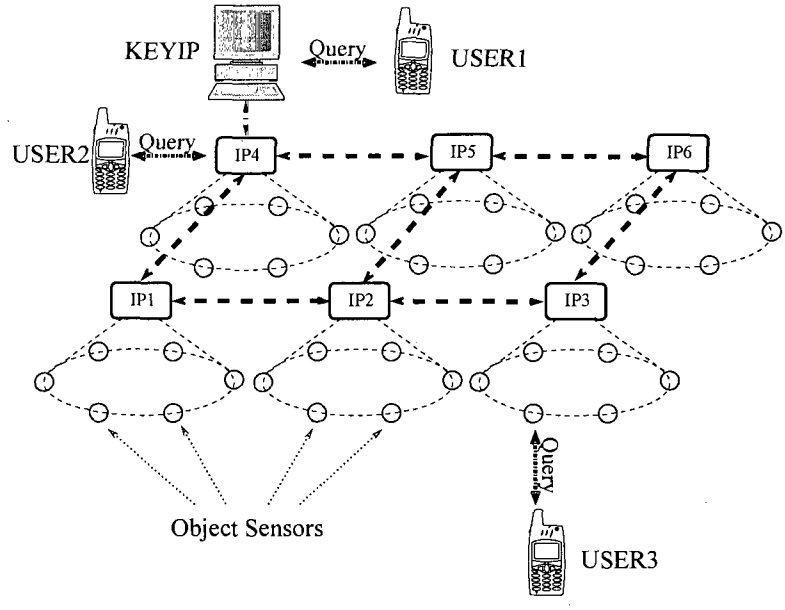


Figure 6.1: Overview of Sensors, *IP*s and *KeyIP* Architecture

[48], or as weak as an active RFID tag. Snoogle only requires all object sensors to communicate under the same radio frequency.

An *IP* is a static sensor-device that is associated with a physical location, for example, a particular room in an office building. *IP*s are responsible for collecting and maintaining the data from the object sensors in their vicinity. A typical *IP* is battery powered, and equipped with a micro controller, radio module and a large amount of flash memory. A collection of *IP*s forms a homogeneous mesh sensor network.

The *KeyIP* collects data from different *IP*s in the network. The *KeyIP* is assumed to have access to a constant power source, powerful processing capacity, and possess considerable storage and processing capacity.

6.1.2 System Architecture

Snoogle adopts a two-tier hierarchical architecture shown in Fig. 6.1. The lower tier involves object sensors and *IP*s. Each *IP* manages a certain area within its transmission range. Object sensors register themselves and transmit the object description metadata to the specific *IP*. *IP*s are responsible for building the inverted indexes for local search. We assume the object description data are either pre-loaded or incrementally uploaded by the object owner. For example, before a novel is place on the counter for sale, the store staff attaches a sensor loaded with the book's introduction to the item, just like putting a price tag on the book. Later, the store staff can upload some reader reviews to the sensor attached so that the potential buyers can directly search the reviews from the book instead of via a remote website like *amazon.com*.

On the upper tier, *IP*s have dual roles. First, *IP*s forward the aggregated object information to the *KeyIP* so that the *KeyIP* can return a list of *IP*s that are most relevant to a certain user query. Second, *IP*s also provide the message routing for the traffic between *IP*s, *KeyIP*, and users. The *KeyIP*, considered as the sink of the network, holds the global object aggregation information reported by each *IP*. Snoogle does not restrict the number of *KeyIP*s. For the simplicity, we only consider a single *KeyIP* setup in this dissertation.

Users query Snoogle using a portable device such as a cell phone or PDA. Snoogle provides two different kinds of queries, a local query and a distributed query. A local query is performed when a user directs his query to a specific *IP*. This type of query occurs when a user wishes to find objects at a specific location. A user performs a distributed query when he queries the *KeyIP*. The distributed query capability allows Snoogle to scale since users do not need to flood every *IP* to find a particular object.

6.1.3 Data Processing in Object Sensors

Each object sensor contains two types of data, *payload data* and *metadata*. Payload is the short description about the particular physical object. Metadata is a representation of the payload data. For example, consider an object sensor attached to a folder. The payload data could be a short note describing the contents of the folder. The metadata is a set of tuples, $\{term_1 : freq_1 : id\} \cdots \{term_n : freq_n : id\}$, where term is a single word describing the payload data, and $freq$ indicates the importance of this term in describing the payload data. A user storing information into an object sensor will create both the payload data and metadata. To minimize the data transmission cost, the data in the object sensor can also be pre-compressed using compression schemes described in the next section.

6.1.4 Data Processing and Storage at *IP*s

*IP*s in Snoogle have two data processing roles. First, *IP*s collect data from object sensors within their range and organize the data into an inverted index. Due the reliability and space concern, the inverted index table in the *IP* is stored in sensor on-board flash memory rather than RAM. Second, *IP*s have to periodically send aggregated information to the *KeyIP* so that the *KeyIP* can maintain its inverted index of *IP*s. *IP*s perform the following three data operations.

Insert: This operation is executed when a new object comes into the *IP*'s region and sends the metadata to the *IP*. The *IP* stores the new metadata with associated term frequency and object id into its inverted table.

Delete: When a physical object leaves the vicinity of a particular *IP*, e.g. a user moves a book from one office to another, the corresponding object sensor is no longer associated with the previous *IP*. The *IP* then performs a "delete" operation to remove all the metadata of the leaving

object from its inverted table.

Modify: This operation is performed when there is a change in the object sensor's data. When this happens, the object sensor sends a modification request to the *IP*. Since the *IP* inverted table is stored in the flash memory, which does not support random write, the "modify" operation is achieved by the combination of a "delete" and an "insert".

We let the *IP* to only store the metadata of the objects, instead of the entire payload data, in the general storage media to conserve storage space. We take advantage of the small granularity write capability of the NOR flash (TelosB on-board flash memory) and allow *IP*s to be able to append the object metadata sequentially in the flash memory.

We also implement a "delete" function that efficiently invalidates the metadata associated with an object sensor. We perform the "delete" by zeroing out the necessary bytes in the flash memory, avoiding the expensive *read and write* method used in general flash storage system. That same memory location is not overwritten until there is a sector delete during garbage collection.

After the sensor sends its id and metadata to the *IP*, the information is first stored in a buffer in RAM. Fig. 6.2 illustrates the *IP* storage architecture. Once the buffer is full, a hash function is applied to every term in the buffer. The hash results are used as the indices that map to the lookup table entries. We maintain the lookup table (INDEX in Fig. 6.2) in RAM to store the address pointing to the flash page. Each flash page has the size of 256 bytes. Those flash pages which are associated to the same lookup entry are organized in a chained structure, very similar to the structure of the linked list in data structure. The value of the lookup table entry always points to the head of the flash page chain. The most populated terms that are mapped to the same lookup table entry are flushed to the flash memory, and the flash address is returned to the lookup table entry. This flushing operation continues until there are enough empty buffer slots to hold the incoming

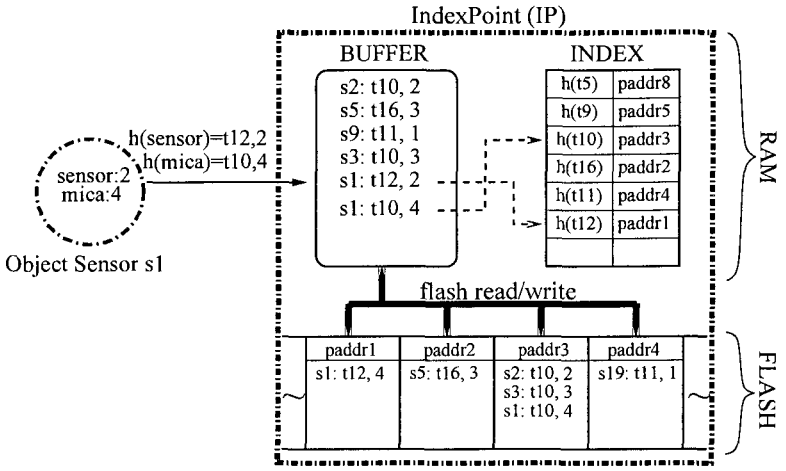


Figure 6.2: Sensor S1 sending data to *IP*

object terms. The lookup table manages the flash addresses in a chained structure that multiple flash pages can be assigned to the same table entry.

When the *IP* receives a query, it applies the hash function to the query to map each query term to a lookup table entry, and obtains the flash address. This address stores a location of the flash page chain head which contains that particular term. Next, each flash page in the chain is sequentially read to the RAM, and scanned for the matching elements. Eventually, a list of matching terms with associated sensor IDs is obtained, then a ranked list of sensor IDs that best match the query is derived using an IR algorithm elaborated in the next section.

Finally each *IP* will periodically send the updated metadata terms and sensors, which reflect the object dynamics in the region, to the *KeyIP*. The *KeyIP* stores the data and checks for inconsistency. This inconsistency arises when sensors moved from one *IP* to another before the *IP*s have a chance to update their information. Since all sensors have a unique id, this inconsistency is easily detected by the *KeyIP*. The *KeyIP* then informs the involved *IP*s verify the sensor data. For example, both IP_1 and IP_2 report having sensor s_1. Each *IP* will send a message directed to s_1. If s_1 is no longer in the range of IP_1, then only IP_2 will receive a reply. IP_1 will flag s_1 as no longer

present and inform *KeyIP*. The same holds if s_1 is no longer in the range of IP_2. If s_1 falls in the intersection of both *IP*s, s_1 will reply to both and *KeyIP* is not updated.

6.1.5 Additional Discussion

When an object sensor lies within the vicinity of multiple *IP*s, the object sensor has to determine which *IP* to select to transmit its data. Ideally, the nearest *IP* in the physical distance could be a good criteria. However, it may not be practical because the *IP* deployment may be restricted at certain locations due to the physical limits. In this dissertation, we use a pre-determined mapping table to identify the *IP* that the object sensor should select. The lookup table maps the RSSI pattern to a specific *IP*. In this scheme, the sensor will sample the RSSI values from multiple *IP*s, and query the nearby *IP*s for the designated *IP*s given the RSSI readings. For example, a first-aid kit placed in the cabinet should select the room *IP* mounted on the other side of the wall rather than a closer *IP* that is mounted in the next room. This lookup table can be precomputed ahead of time, and can be stored in *IP* flash memory.

The use of RSSI for localization is widely studied, and Snoogle can be modified to use more advanced localization algorithms [11, 76, 10, 80, 67] to achieve more accurate localization.

While we do not specify the maximum number of objects sensors that can associate with an *IP*, we assume that there should not be more than about 200 objects that lie within the range of a single *IP*. The reason is that even though object sensors are small, the sensors are attached to larger physical objects like laptops and coffee mugs. For a smaller space like a office cubical, the number of tagged things are unlikely to be in the hundreds or thousands, thus the *IP* never has to index so many items. Furthermore, since the *IP* will delete data from object sensors that have left its vicinity, there will no accumulation of data.

Larger spaces such as a warehouse storage area may contain thousands of object sensors. In this situation, multiple *IP*s can be installed to index the data from the sensors. As mentioned earlier, a sensor facing a choice of multiple *IP*s will send the RSSI values to the *IP*s which will then assign an *IP* for that sensor to associate with. This way, the object sensors can be associate with an appropriate *IP* to facilitate searching, and will not be concentrated into a single *IP*.

6.2 Communication Compression

A Bloom filter [7] is used in Snoogle to compress groups of terms together. A Bloom filter with a m-bit array and k independent hash functions is used for every n words. The m-bit array is first initialized to "0". Then, for each word, the hash function maps the input to a value between 0 and $m-1$, corresponding to the bit position in the bit array, and that bit is then set to "1". After n words are inserted, the resulting value of the array becomes the summary of the n words. The collection of the arrays becomes the summary of the document. To check whether or not a word is in the document, we apply the k hash functions to the word and check if the resulting bit positions are all "1"s in any of the array collection. Any "0" indicates there is no match. However, a result of all matching "1"s only indicates there is a certain probability that there is a real match. The uncertainty is due to false positive (or collision, we use false positive and collision interchangeably in our description). If a Bloom filter has m bits, k functions, and holds n words, the probability of having a collision (incurs the false positive) with another word is

$$(1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{-kn/m})^k. \tag{6.1}$$

When m and n are fixed, the optimal false positive rate can be achievable when [30]

$$k = ln2 \cdot \frac{m}{n} \tag{6.2}$$

Bloom filters can be further compressed to achieve better transmission efficiency [65]. This is based on the observation that a m-bit string may be transmitted by a less number of bits without any information loss. We denote z as the number of bits after compression. Note the compression only works ($z < m$) when there are less "1"s than "0"s (or in reversed case). Mitzenmacher [65] indicated that each bit of the Bloom filter has roughly 1/2 probability to be "1" or "0" when a Bloom filter is tuned to have optimal false positive rate. This tells us an optimal Bloom filter almost cannot be compressed. It also means there is trade-off between false positive and compression ratio. To gain transmission efficiency, we have to sacrifice the false positive rate. Mitzenmacher [65] continued to point out that the procedure of compressing a Bloom filter is actually equivalent to hash each term into a z/n bit string. Therefore, instead of doing complicated bit operations, we simply hash each term to a z/n bit string, and concatenate the n hash results together to generate an array. Suppose the hash function is perfect, the probability of having a collision with another word for each z/n bit string is roughly $(\frac{1}{2})^{z/n}$.

Selecting the correct compression method is crucial for Snoogle system. The optimal bloom filter achieves the lowest false positive rate, while the compressed bloom filter scores better compression ratio [58] so that it can achieve better transmission efficiency and lower processing overhead. We believe that the low transmission cost and processing overhead are more desirable for extremely resource constrained sensor nodes. Therefore, weighing the pros and cons of the two compression schemes, we use the compressed Bloom filter for our Snoogle system. Actually, with carefully chosen parameters, we can lower the false positive rate to an acceptable level. As we

will describe in the evaluation section, given the data set with 1512 words, and compressed Bloom filter size of 16 bits, the false positive rate is only about 2.3%.

6.3 Performing Query

In this section, we present the details and theoretical discussion of the user query process and top-k query schemes.

6.3.1 Query Process

There are two ways of querying Snoogle. The first is to query an *IP* directly, the second is to query the *KeyIP* first, and then to perform the distributed query given a list of most relevant *IP*s returned by the *KeyIP*.

The first query method is used when a user is only interested in finding the sensor in some specific area, or if the user has an approximate idea where the sensor might be found. For example, a user wants to find a particular magazine, but only if it is within a short distance from where he is currently at. Thus, he only queries the *IP* near him by sending a few terms that describe this magazine. The *IP* evaluates the answers to the user. Each answer is the id of a sensor that best matches the user query. The user can then query the sensor directly, or physically find the sensor and hence the object.

The second query method is used when a user wishes to find an object regardless of where it is, or has no idea which *IP* to start querying. The user first queries the *KeyIP* with several terms describing the target object. The *KeyIP* then returns a ranked list of m *IP*s that contain objects that best match the query, where m is a system parameter. The user then perform the distributed top-k query from the returned m *IP*s and find the satisfied answers.

6.3.2 Improving Query Accuracy

When a user queries an *IP*, he receives a ranked list of sensor IDs that best match his query from the *IP* as his answer. This ranking is derived from a score for each sensor contained within that *IP* based on the query terms. For example, the user issues a query with two query terms, (t_x, t_y) to an *IP* with three sensors, (s_1, s_2, s_3). The score for s_1 is the sum of the weight of t_x in s_1 and weight of t_y in s_1. The score for s_2 and s_3 are determined in a similar fashion.

The weight of a term in a sensor is determined using the TF/IDF weighing algorithm from IR research. The intuition behind TF/IDF is that the importance of a term in describing a sensor is based on two considerations. The first is the number of times that term appears in that sensor, the TF. The more often a term appears in the sensor, the more relevant that term is in describing that sensor. In our system, the TF value is given as part of the metadata of the sensor.

The second consideration is how important that term is among the *collection* of all sensors in a particular *IP*. The *IDF* is determined as

$$IDF = \log\left(\frac{\text{Total number of sensors}}{\text{Number of sensors containing the term}}\right).$$

The idea here is that if a term appears in many sensors found within an *IP*'s neighborhood, it is less important. Consider the extreme case where a term appears in every sensor under an *IP*. Then, any one of the sensors returned will contain that term, making that term not descriptive of any one sensor at all. To get the *IDF* value, we need the total number of sensors and the number of sensors containing the term. The first one is easy to get since an *IP* knows all sensors in its neighborhood. The second value is acquired while processing the query at an *IP*. Given a query term, an *IP* counts the number of the matches with stored terms in its flash memory.

Putting it all together, the weight for a term t_x in a sensor s_1 is

$$\text{Weight of } t_x = (TF_{t_x} \text{ in } s_1) \cdot (IDF_{t_x} \text{ in } s_1).$$

The above TF/IDF scoring methods can also be used to evaluate the weight of the IP in the distributed query.

We initially considered CORI weighing algorithm [33] when a user queries the *KeyIP*, but there was no noticeable improvement. Thus we use a simple TF/IDF algorithm throughout this chapter.

The use of the IDF allows the appropriate answer to be derived when comparing different IPs. Consider for instance two IPs, IP_a and IP_b, where IP_a is placed at a music CD store, and IP_b is placed inside a student's dorm room. When the student wants to query for his **own** CD, he would like to obtain an answer from IP_b rather than IP_a. However, since IP_a is placed in a store with a lot of music albums, IP_a will contain more terms associated with music albums, even though the appropriate answer should be from IP_b.

If our scoring algorithm only used the TF, then the answer from IP_a will be better than IP_b, since there are more sensors in IP_a that contain album terms. However, using IDF means that the scores from IP_a will be smaller since the album terms appear in almost all the sensors in IP_a, resulting in a much lower overall score. This behavior allows IP_b to be returned to the student as the most likely location for his own CD album.

6.3.3 Performing Top-k Query

While Snoogle is capable of returning a ranked list of all relevant objects matching a query to a user, a user will usually want to limit the number of replies due to limited device display or battery

power. Snoogle allows the user to specify a top-k query which returns the k best matches to a user query. The k is a user specified value.

For a local query, returning the top-k query is straightforward since an IP needs to only return the top k answers to the user. For a distributed query, a naive top-k query scheme is for the user to perform a top-k query for each of the m IPs returned by $KeyIP$. By collecting the $m \cdot k$ answers the user can then obtain the top k objects. However, the message complexity of $O(mk)$ is too expensive for the energy constrained system.

Our distributed top-k query algorithm is shown in Algorithm. 6. The intuition is as follows. Upon receiving a list of m ranked IPs, the $KeyIP$ queries each IP for the most relevant object, denoted as $ta_i, 1 \leq i \leq m$. The $KeyIP$ stores the m objects in an array a such that $a[i].obj = ta_i, a[i].weight = weight(ta_i), a[i].ip = IP_i$, where $weight(ta_i)$ returns the weight score determined by TF and IDF as we discussed previously. After collecting the top weighing objects from all m IPs, the $KeyIP$ sorts the objects in the descending order of the object weight, and obtains a new array that $a[1].weight \geq a[2].weight \geq \cdots \geq a[m].weight$. The first top-$k$ answer, $a[1].obj$, is immediately available. The $KeyIP$ sets the threshold value as $a[2].weight$, and queries $a[1].ip$ for the objects (excluding $a[1].obj$) that weights more than the threshold value. Note that among all the m IPs, it is possible for IP $a[1].ip$ to solely hold objects with weights larger than $a[2].weight$, so there is no reason to firstly query other IPs. Ignoring objects that are designated as top-k objects, each IP has a new top weighing object, and the same process continues till all top-k objects are found. The algorithm stops any time when k top objects are retrieved, and the $KeyIP$ returns the answer to the user.

An important issue in the above algorithm is the accuracy of the merged object ranking. Note that the object weights reported by each IP are local scores, which are determined by local IDFs.

However, the local weights cannot be compared directly [12]. We consider following two solutions: 1. normalizing the local scores; 2. calculate the global scores. As indicated in the study [12] by Callan *et al.*, the common normalization scheme requires the exchange of object statistics among *IP*s, which may incur large amount of communications between *IP*s. On the other hand, the heuristics used in the proposed normalized score estimation are tied to the specific database and therefore cannot be used in Snoogle. In this work, we choose to calculate the global *IDF*. Upon receipt of a user query, the *KeyIP* first query m *IP*s for the local *DF* value of each term. After collecting all local *DF*s, *KeyIP* immediately computes the global *IDF*s and sends them back to *IP*s. From now on, the weights computed at each *IP* becomes global scores and then can be compared with each other. Although this approach requires an extra round of communication between *KeyIP* and *IP*s, the actual cost is bounded by $2m$, where m is the number of *IP*s.

To bound the number of messages transmitted in the process, we make the following observations. First, each *IP* transmits at most one object that will not appear in the top-k list. Therefore, the number of messages sent by all the *IP*s is at most $m+k$ including the top-k objects and other objects that will not appear in the top-k list. Second, for each query sent out to the *IP*, we will get back at least one object (which may appear or not appear in the final top-k objects). Thus, the number of queries sent out to all the *IP*s is bounded by the number of received objects, which is at most $m+k$. From the two observations, the number of messages in this process is at most $2(m+k)$, which is more efficient than $m \cdot k$ in naive scheme.

6.4 Mobility and Security Support

Here we present the system enhancement for supporting the mobile objects, and describe a flexible and resilient security mechanism for private objects.

6.4.1 Supporting Mobile Objects

An *IP* needs to keep up-to-date information about the object sensors in its neighborhood. However, since objects can be mobile, there will inevitably be object sensors moving in and out of an *IP*'s neighborhood. Snoogle uses a combination of *beacon* and *timer* methods to maintain updated information.

In the *beacon* method, the *IP* will periodically broadcast a beacon that identifies itself. An object sensor in the neighborhood that receive this beacon will compare it against the previous beacon. If both beacons match, this indicates that the physical object's metadata has already been sent to that *IP*, and the sensor does nothing. Otherwise, it indicates that the physical object has moved to a new location, and sensor will send the metadata and id to the new *IP*.

In the *timer* method, the communication is initiated by each individual sensor. Each object sensor periodically broadcasts a "keepalive" message. At the same time, the *IP* maintains a timer. If the *IP* does not receive any "keepalive" message from a certain associated object before the timer expires, the *IP* considers the object is gone, and then deletes the all data of the object sensor from its storage. The *beacon* and *timer* methods can vividly regarded as "pull" and "push". In the *beacon* method, *IP*s pull the status information from the object sensors. In the timer method, object sensors push their status to *IP*s.

The *beacon* scheme consumes less energy than the *timer* method. The object sensors only need to wake up in the duty cycle to listen the beacons. They do not need to transmit any message

as long as there is no movement. The *timer* method, however, offers better reliability. When an object moves to another *IP* neighborhood, the previous *IP* can notice an object missing through the timer, and the new *IP* also can also be notified by the timer message sent by the moving object. In short, the *beacon* method is more suitable for static objects, while the *timer* method works better for mobile ones. In practice, the two methods can be properly combined depending on the system requirement.

6.4.2 Providing Security and Privacy

Since Snoogle is built on sensors, it shares all common security threats with other applications in sensor networks. Furthermore, Snoogle also poses unique security and privacy requirements in searching. The concern is that Snoogle may violate personal privacy by revealing object information to others. For example, a user may not want his private object (i.e., DVD movie) to be searchable by strangers, but only his friends and himself.

Based on the above concerns, Snoogle must have a security mechanism to prevent objects from being searched by unauthorized users. We adopt the public key cryptography rather than the symmetric key scheme to have a clean user interface and a simple key management. Recent research [41, 59] have demonstrated that public key schemes are feasible for sensor nodes. We developed an Elliptic Curve Cryptography (ECC) public key scheme for Snoogle. The reason we choose ECC over more popular RSA is that ECC can be more efficiently implemented in resource constrained sensors. On TelosB sensor motes, it takes 1.4s to generate a public key. To the best of our knowledge, this is the best ECC performance achieved among the academic implementations. In Snoogle, the access control is performed at the *IP* instead of at *KeyIP* in a distributed fashion.

We provide security for Snoogle by adding a security tag field to the object sensor. The security

tag has an *OwnerID* field and a *GroupMask* field. The *OwnerID* refers to the owner identification. The *GroupMask* determines which group of users has the privilege to access the object. The ECC-based user authentication is very similar to RSA. If a user wants to search private objects, he first sends the query and the certificate, where the certificate is issued by Certification Authority (can be Snoogle administration). The *IP* first verifies the user certificate and then makes sure the corresponding *OwnerID* and *GroupMask* matching with the object tag. In the next step, the *IP* uses the derived user public key (from the certificate) to encrypt a randomly chosen secret key, and sends the ciphertext to the user. If the user can successfully decrypt the key, it proves that the user is the legitimate owner of the certificate. Finally, the key is used to establish the secure channel between the *IP* and the user. This key can also be used to achieve the user privacy, since the user can simply encrypt his query terms by using the key so that no one can learn the query content.

6.5 Prototype Experience

6.5.1 System Setup and Parameters

We implement a prototype of Snoogle system, including object sensors, *IP*s, *KeyIP*, and user module, on TelosB motes, a research platform developed by Berkeley. TelosB hardware features a lower-power TI MSP430 16-bit micro-controller with 10KB RAM and 48KB ROM. The on-board IEEE 802.15.4/ZigBee compliant radio transceiver facilitates the wireless communication with other IEEE 802.15.4 compliant devices. TelosB also has an on-board flash memory with 1MB space, which enables our prototype *IP* to store as many as 262,144 terms and the associated object IDs and term frequency. The low-power feature ($5.1\mu A$ current draw in sleep mode) of TelosB motes allows object sensors to stay alive for long time. We use an HP iPAQ for the user module.

The HP iPAQ features a 522MHz ARM920T PXA270 processor, 64MB RAM and 128MB flash memory. The software of *IP*s, object sensors and user module are written by NesC language on TinyOS version 1.1.15. Table 6.1 illustrates the summary of the implementation.

KeyIP	Laptop Computer
Index Point	TelosB mote Code size (binary): 31.7K bytes Data size: 3475 bytes Index table: 16 entries Buffer size: 64 metadata Flash page size: 256 bytes
Object Sensor	TelosB mote Code size (binary): 19.3K bytes Data size: 6054 bytes
User Module	TelosB + iPAQ Code size (Java class):10.8KB

Table 6.1: The summary of Snoogle implementation.

We adopt the RC5 block cipher as the cryptographic hash function to implement the compressed Bloom filter. We choose 16-bit as the Bloom filter size. Given our data set with 1455 unique terms, the false positive rate is only 2.3%.

We set up a Snoogle network in our computer science building. The floorplan and the deployment of object sensors, *IP*s and *KeyIP* are shown in figure 6.3. Our experiment consists of two *IP*s in two wings of the building, in rooms 101 and 107. We attach a laptop to each *IP* to collect timing

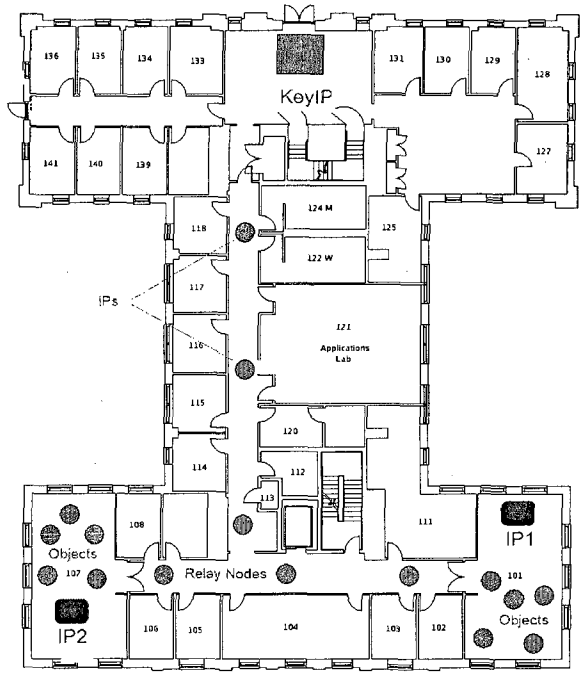


Figure 6.3: Floorplan for testbed. The *KeyIP* in rectangular shape is placed in the lobby. The two *IP*s, *IP*1 and *IP*2, also in rectangular shape, are located in two wings of the building, in rooms 101 and 107. There are 5 object sensors (squares) in the neighborhood of each *IP*. To route the messages between *IP*s and *KeyIP*, we also deploy 6 IPs (round dots) in the hall way.

information. The laptop plays no role in processing any of the data sent from sensor to *IP*. Each *IP* has in its neighborhood 5 sensors which simulate sensors attached to different objects. Each sensor contains data from one complete conference paper from the workload discussed earlier. The *KeyIP* is placed in the lobby, and consists of a laptop using a TelosB sensor as a communications module. Since the *KeyIP* in our scheme can be a server, all data processing is performed by the laptop. We deploy several additional *IP*s to route data to the *KeyIP*

We assume that a user will query Snoogle with a PDA like iPAQ. Since iPAQ does not directly support sensor communications, we use a TelosB mote to attach to the iPAQ through a USB adaptor as the front end radio communication module. The iPAQ is running Windows Mobile 5.0 and Mysaifu JVM [69]. Fig. 6.4 shows a picture of our *PDA* querying device.

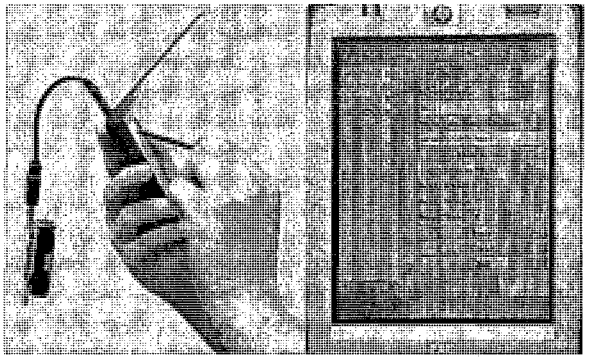


Figure 6.4: Picture of a *PDA* Query Device

6.5.2 Prototype Test

We use the prototype test to demonstrate the validity of the Snoogle architecture in a real world environment. The evaluation of specific Snoogle components are left to the next section. We consider the following two tests. First, can a user successfully query the *KeyIP* to get a list of *IP*s, and can he query an *IP* to obtain a list of sensors? Second, can the *IP* and *KeyIP* effectively and accurately detect and manage mobile sensors?

Our first test emulates an user's query experience. We are particularly interested in the time duration for a user to get the object he searches for. A graduate student wants to search an academic publication. He first enters the lobby of Computer Science building, and uses his iPAQ to query the *KeyIP* with the paper key words. The *KeyIP* immediately replies him with the list of *IP*s that carries the record as well as the associated term frequency information. Given the information replied from the *KeyIP*, the student picks the *IP* which most probably contains the paper he is looking for, which is *IP*1 in our experiment setup. He then immediately queries *IP*1 again. This time, IP1 gives more detailed answers which finally help the student to find the paper. The time duration for the whole procedure is 1 minute and 45 seconds. The most time is spent on walking across the hallway and operating iPAQ. Actually the *KeyIP* and *IP* query response time only has

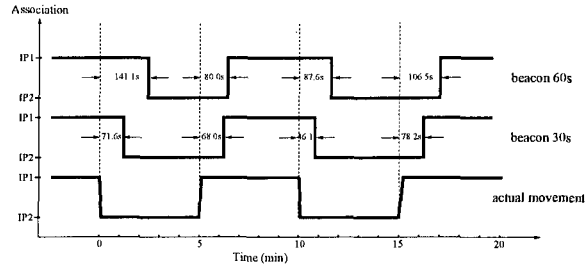


Figure 6.5: Mobility test with beacon method

a very tiny portion of the total time consumption, it only takes $41ms$ and $55ms$ for *KeyIP* and *IP* to reply the user query, respectively.

Our second experiment tests whether or not the *KeyIP* and *IP*s are able to give a correct answer to the query for a mobile object. We implement both the beacon and timer methods in the prototype and test them separately. In each test, we set the cycle period with 30 seconds and 60 seconds. The mobile object starts at the neighborhood of $IP1$. When the experiments begin, one of our group members takes the mobile object and walks cross the hallway to room 107. He stays in room 107 for 5 minutes and then carries the object back to room 101. During this period, we keep track of the object status in *KeyIP*.

The test results of the beacon method is shown in Fig. 6.5. In the test with 30s beacon cycle, it takes 71.6s for the *KeyIP* to get the object update report from $IP2$. Once the update arrives, the *KeyIP* detects the object was originally associated with $IP1$. The *KeyIP* therefore immediately issues a notification to $IP1$. Note $IP1$ has no way to notice the missing object in the beacon method. After 5 minutes, the object returns to room 101, it takes 68.0s for the *KeyIP* to receive the update from $IP1$. Similarly, *KeyIP* issues a notification to $IP2$. When the beacon cycle is extended from 30s to 60s, it takes longer time to detect the associate change.

Fig. 6.6 plots the mobile object association changes at the *KeyIP* in the timer method. Comparing the two methods, the timer method is more reliable than the beacon method. The timer

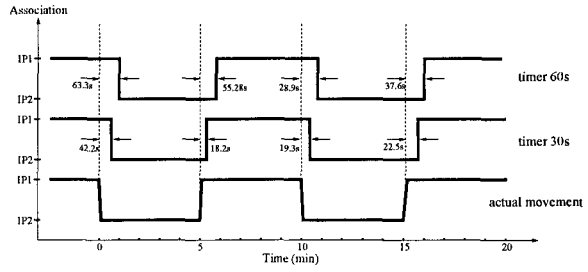


Figure 6.6: Mobility test with timer method

method allows both the *IP* and the object to detect the movement while the *IP* cannot detect the leaving object in the beacon method. That explains why it takes more time for the beacon method to detect the object movement once the object itself misses a couple of beacons, as shown in Fig. 6.5. However, the timer method requires more energy consumption for the object sensors because they have to keep sending "keepalive" message. This is a disadvantage for extremely energy constrained sensors. In practice, the two methods may be combined together to achieve the tracking performance to the energy consumption ratio.

6.6 Performance Evaluation

To better discern the performance of the system, we break the search system down into individual components and evaluate each separately. We mainly focus on object sensor and *IP* interaction because both the sensor and *IP* are power constrained and computationally challenged devices, while the *KeyIP* is a resource-rich device. This makes the performance of the object sensors and *IP*s crucial for the validity of the system.

6.6.1 Workload Design

We use data from an academic conference to create our dataset. The title, authors and affiliations of each entry become the metadata terms in each sensor. We use the IR definition of TF to obtain the weight of each metadata term. This yields a workload of 1455 terms, which are sufficient for about 80 sensors, each of which has about 15 to 25 unique words on the average. The average term size is between 7 characters to 8 characters. We further divide the 80 object sensors into eight IPs as the testbed for the distributed query.

6.6.2 Data Input and Maintenance at IPs

The startup phase for our search system occurs when the IP is first initialized and contains no object data at all. This is a costly activity since the IP has to identify all the sensors within its range, and obtain their metadata. Fortunately, this initialization phase occurs rarely since the IP utilizes persistent flash memory for data storage to protect against data loss. The main metric we use to evaluate this portion is the time latency needed for an IP to obtain necessary data from object sensors and update the collected data for the future changes to give accurate answers for queries.

To reduce the transmission cost and improve the storage efficiency, Snoogle adopts the idea of compressed Bloom filter to compress the metadata terms. In particular, a hash function residing in the object sensor convert each plaintext metadata term into a 2-byte digest before transmitting the data over to the IP. We perform a comparison test to learn the benefit of the data compression. Fig. 6.7 shows the time taken to transmit hashed data to the IP compared to the plaintext method. As we can see, the transmission time grows linearly as the number of terms increases when the plaintext data is used, while it takes much less time for the IP to collect the same amount of data

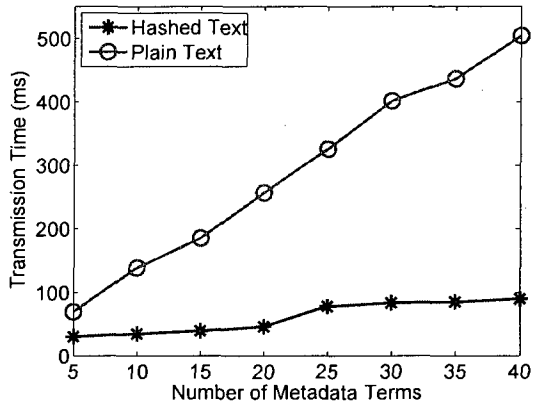


Figure 6.7: Time taken to transmit metadata to *IP*

in the compressed form. It only takes 90*ms* to collect 40 compressed terms. However, it requires more than 5 times of amount of time to transfer 40 uncompressed terms.

Note that the time taken to transmit hash terms is not proportional to the number of terms. For instance, the transmission time for five hashed terms is 30ms, while the transmission time for 40 hashed terms is 90ms. It only takes triple the amount of time to transmit eight times the data. The reason is due to TinyOS message overhead. For the communications between each object and an *IP*, we set the message payload size up to 60 bytes. Besides the 60-byte payload, each messages has 8-byte message header and 10-byte TinyOS header. We need three bytes to transmit one hashed text: one for the term frequency and other two for hashed value. Given a 60-byte payload, one message can carry up to 20 terms. Due to the above reason, even though the text size of 40 terms is 8 times of that for 5 terms, it only takes one more message to transmit 40 terms, compared to that for sending 5 terms. Therefore, the transmission time for 40 terms should not be 8 times of that for 5 terms.

Next, we show how the buffer helps to further improve the data collection efficiency. An *IP* has limited RAM and uses flash memory to store the sensor metadata. The flash memory

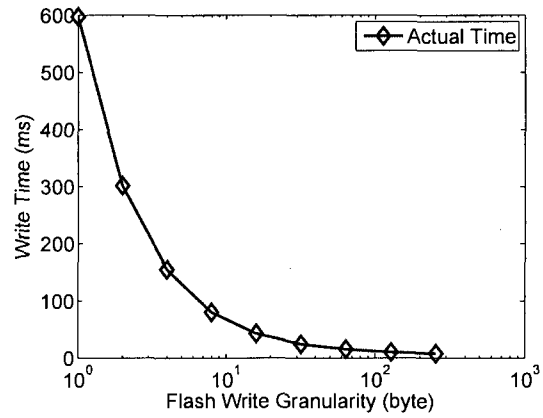


Figure 6.8: The time delay to write 256 byte of data with different write granularity.

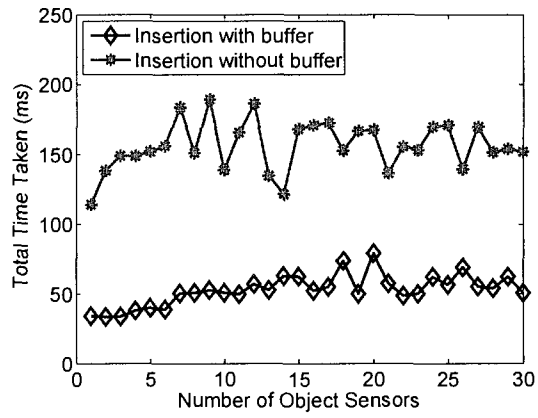


Figure 6.9: Insertion performance with buffer and without buffer at *IP*

operation principle determines that the write in flash memory, specially in small granularity, is slow. We have performed a simple test to show the write efficiency at different write granularity. Fig. 6.8 plots the experiment result. As we can see, it consumes almost 0.6s to write 256 bytes in flash with 1 byte write granularity, compared to 8ms to write the same amount of data with 256 byte granularity. A common way to amortize the memory write overhead is to use a buffer. In Snoogle, each *IP* maintains a small buffer in RAM of 256 bytes, to buffer sensor data before flushing to flash. Even though the NOR flash in TelosB supports random writes, we adopt the buffering approach to improve the efficiency. When there are multiple sensors wanting to send

data to an *IP*, the *IP* will have to periodically halt transmission to flush the coming data into flash. The *IP* does not need to invoke the expensive flash flushing routine as long as there is enough buffer space to hold the coming object terms, and picks a spare time later to flush the buffered terms into the flash.

Again, we conduct the comparison test to compare the two schemes. For the test case with an *IP* buffer, we choose the buffer size with 256 bytes, equivalent to the page size of the flash memory setup. Since each object term requires 4 byte memory space, including 2 byte digest, 1 byte term frequency and 1 byte for the object id, a 256-byte buffer can hold at most 64 object terms. In the both experiments, 30 object sensors, each having 10 terms, sequentially transmit the data to the *IP*. We record the average waiting time of each object sensor and present the results in Fig. 6.9. It clearly shows that each object sensor waits significantly less amount of time when the *IP* uses the buffer.

We further notice that the variation of the object sensor waiting time without an *IP* buffer is much larger. Our investigation reveals that the variation is determined by the amount of time taken to flush the data to the flash. Since each compressed term is further hashed by the *IP* (as previously described in Section 4) to an index table, different terms can be mapped to different positions of index entries. The number of entries can be any value between 1 and the number of terms. The bigger the number is, the longer time is required because the *IP* has to flush more flash pages. As the comparison, this variation is much smaller with a buffer enabled *IP*. The reason is that, the *IP* buffer keeps track of the index entry position of each term. When the number of buffer empty slots is not enough to hold the coming data, the buffer first flushes the most populated terms that hashed to the same index position, and stops flushing if there are enough space. As the result, with a high probability the number of pages required to be flushed is less than that in a bufferless *IP*.

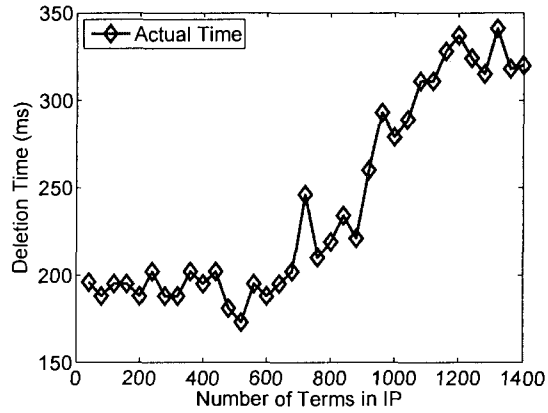


Figure 6.10: The amount of time to delete an object with 10 terms.

When an object is removed from its original location, the *IP* has to update its inverted index table to reflect such change. As described previously, the delete operation requires the *IP* to scan the entire valid flash storage area and tag the deleted object terms to be invalid. It is not difficult to suggest that delete performance is determined by the size of stored flash data. Our experiment results, as shown in Fig. 6.10, exactly show this trend. The experiment is conducted in the following way. We select a specific object sensor with 10 terms, and perform deletion with different amounts of data loaded in the *IP*, ranging from 0 to 1600 terms. Initially, the deletion time does not vary much when the number of loaded terms increases. The reason is that the *IP* has to scan at least one flash page for each index entry, no matter how many terms have already been stored in the flash. When the term number continues to grow, some index entries require more flash page to store the metadata terms. Therefore, the deletion operation has to scan more flash pages. As the result, the time consumption increases accordingly.

Note that deletion does not have to be done each time a sensor leaves an *IP*'s neighborhood. A simple list can be kept by the *IP* that records the IDs of sensors that have left. Then, before the *IP* replies to a query, it removes the sensors found in the list from the answer. This way, the user

will still have the correct answer. The *IP* can then perform the deletion in the background when there are no other pending query requests.

6.6.3 Local Query

To evaluate the local query performance, we focus on two main areas: query latency and query accuracy. We first test the performance of the query latency of Snoogle. Then, we demonstrate the Snoogle query efficiency by a comparison test that compares the latency performance between Snoogle and a flat structured network. Finally, we evaluate the query accuracy.

6.6.3.1 Query Latency

Query latency is the time taken for a user querying an *IP* to receive a reply. This includes the time to transmit, process and reply to a query. To better evaluate our search system, we measure the query latency using common web search characteristics. From [51], the average number of query terms per search is less than 3. We then determine the average time taken to complete a user query comprising of one to four terms. Fig. 6.11 shows the results. We see that the query response time increases as the number of query terms increase. As mentioned in section 4.1, multiple flash pages may have to be read from flash memory to determine the *IDF* of each query term. This accounts for the increase in query response time.

The above experiment has shown an example of the local query performance given a typical system setting with 80 objects. To evaluate the scalability performance of the local query, we design the following test. In the test, the IP term index table is configured with 16 entries. As we discussed in section 6.1 , each object term (after being compressed) is hashed again to get its index in the index table that stores the address pointing to the flash memory. Given the ideal

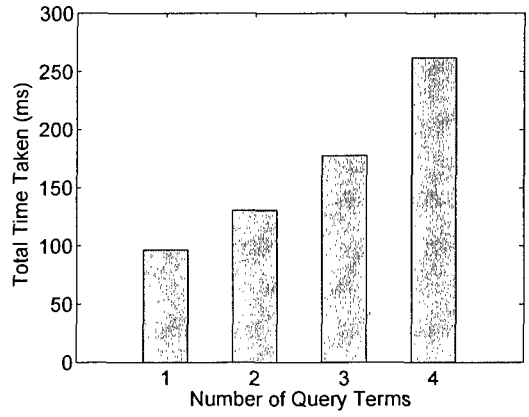


Figure 6.11: Time taken for *IP* to respond to a query

hash function as we assume, each term is equally likely hashed to one of 16 indices. Thus, the 16 page chains, pointed by 16 indices, can be considered to have the same length. Based on the above consideration, we generate a number of synthetic object terms and directly load them into the flash memory of an *IP*. We perform the local query performance as the flash memory starts from empty to full. Suppose on average each object has 30 terms, an IP with 1MB flash memory can index up to 262,144 terms.

The experiment results are plotted in Fig. 6.12. We find the local query time is affected by two parameters: 1. the number of query terms; 2. the number of objects. For each query term, the IP needs to scan the flash memory pages that are pointed by a certain entry in the indexing table. Our test shows that it takes around 7ms to read a page of flash memory to RAM by a TelosB sensor. Multiple terms then require the multiple scans and thus produce the corresponding delay. Note the page scanning delay dominates the total query response time. The similar trend is observed that more query time is used when the number of object increases as more terms are stored in flash memory. Overall, the query time is efficient. It takes 1.2s for an *IP* to respond a 4-term query when there are 256 objects.

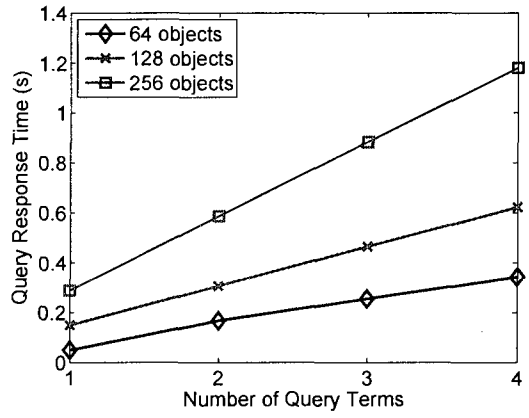


Figure 6.12: Time taken for *IP* to respond to a query.

6.6.3.2 Compare to searching without *IP*s

An alternative searching method is to have users query the sensors sequentially, and then collect the replied data to find the desired information. This method gets rid of the *IP*. To evaluate we implement this alternative searching scheme and compared the performance against our Snoogle system. The alternative searching scheme is implemented as follows. A group of sensors are organized to a chained structure. The user always queries the chain head sensor, the queried sensor searches the query term in its memory and puts the results at the pre-assigned position in the message packet, and then forward the query to the next sensor in the chain. The 2nd sensor repeats the above searching and puts the results in its pre-assigned position. This procedure repeats until the last sensor finishes the query processing. The last sensor directly replies to the user. We believe this is the most efficient way that a general searching scheme can achieve because it requires lowest amount of the message transmission. We select 10 sensors for the both experiment setups. Each sensor is pre-loaded with the metadata of one conference paper. The user performs a single term query to the both systems. We measure the user query response time with the number of object sensors changes from 1 to 10. In Fig. 6.13, we show the difference in query response time

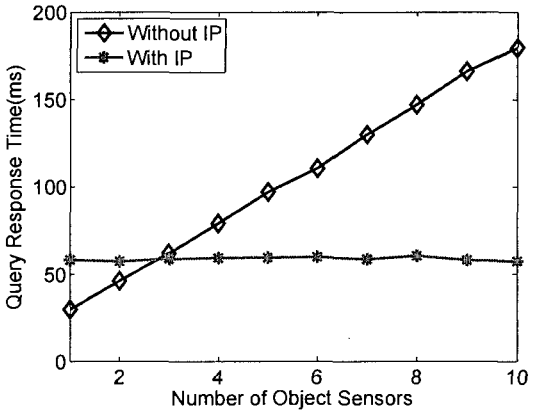


Figure 6.13: Query latency with and without *IP*s

in two different searching systems. We see that the query response time in Snoogle system remains relatively constant. The time taken in general searching system, however, increases linearly with the number of objects increases. This proves Snoogle achieves much better scalability than any general searching scheme.

6.6.3.3 Query Accuracy

Query accuracy in traditional IR uses *precision verses recall* to evaluate the effectiveness of a search system. However, Shah and Croft [81] pointed out that using the mean reciprocal rank (MRR) metric from question answering (QA) was more suitable when performing IR on power constrained or bandwidth limited devices. In QA, the emphasis is to return a single or a very small group of answers in response to a query, and not the return as many relevant answers as possible. In other words, the QA metric places more emphasis on the accuracy of the *ranking* of answers. This is apt for our search system built on sensors. The MRR is defined as

$$MRR = \frac{1}{\text{rank of first correct response}}$$

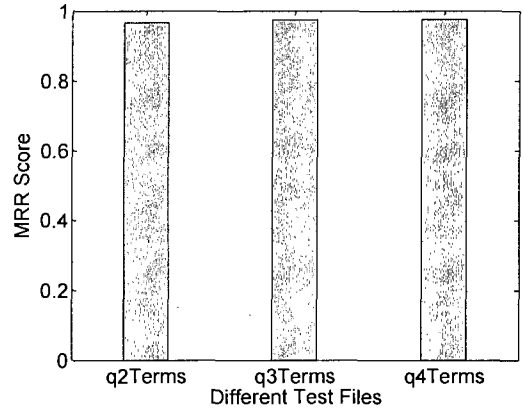


Figure 6.14: Accuracy of query answer

For example, let the search system return a ranked answer (A, C, B) where the model answer is B. In other words, the correct highest ranked response should be B. The MRR for this query is thus $\frac{1}{3} = 0.33$ since the correct answer is three spots off. An answer that matches the model answer will have a MRR of 1.

We first generate three different test files, *q2Term*, *q3Term*, *q4Term*, each file containing a collection of two, three and four query terms respectively. Each collection consists of 20 different questions and model answers. These questions are designed to contain some degree of ambiguity. For example, in our collection, there are two papers with DHT in their title, four papers with Frans Kaashoek listed as an author, and two papers with Jeremy Stribling as an author. Thus, a query "Kaashoek DHT Stribling" will have the model answer the paper titled "Bandwidth-efficient Management of DHT Routing Tables". In other words, the ranked list of answers returned should have this paper as the top ranked result. Fig. 6.14 shows the results. We see our search system has a high MRR for different number of query terms. We do not test for only one query term since to derive a model answer, the query term needs to be unique, which is equivalent to a simple grep match.

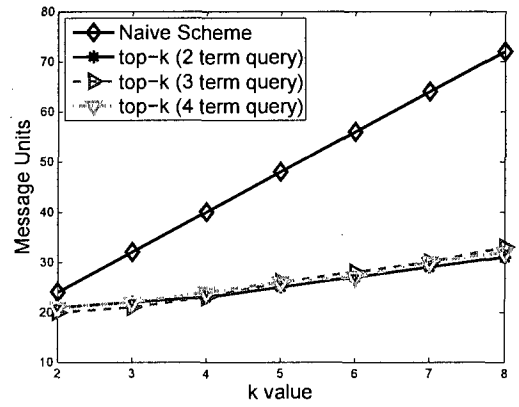


Figure 6.15: Message complexity of distributed top-k query.

6.6.4 Distributed Top-k Query

As we discussed in Section 6.3.3, the message complexity is the major concern in the distributed top-k query. To evaluate the performance of our top-k query scheme, we first perform a simulation based on our own data set to study the message complexity. Then, we estimate the query response time on a larger network setting by using the packet transmission time collected from our experiments. In the both studies, we compare the performance of the proposed top-k query with that of the naive scheme.

6.6.4.1 Message Complexity

In the simulation, we use the same dataset which is composed of 80 objects. We evenly and randomly distribute these objects into eight *IP*s (each *IP* has 10 objects). In this way, we create a testbed for the distributed query with eight *IP*s, which are returned from the *KeyIP* for the user query (note $m = 8$). In the next step, the user performs the distributed top-k query.

We implement our distributed top-k query scheme on our simulator since our interest is the message complexity only. The rule of determining the message complexity is explained as follows.

1. A single user query to a certain *IP* is counted as one message unit. 2. The answer with k objects from a certain *IP* is counted as k message units since the message length grows as k increases. We run the simulations for three different queries with two, three and four query terms, respectively. We first randomly distribute the objects into eight *IP*s, then run the query and count the message numbers. We repeat this procedure for 100 times for each simulation and calculate the average message count values. For the comparison purpose, we also implement the naive top-k query scheme. Note there is no change in message complexity of naive scheme given variant object distribution and query term numbers.

The simulation results are shown in Fig. 6.15. As we can see, the performance of naive scheme is significantly worse than that of our distributed top-k query scheme. When k increases by one, the naive scheme needs m more messages (here $m = 8$). Comparatively, the number of extra messages required for our top-k query is much less than m. As the result, when k increases to eight, the naive scheme costs 72 messages, while our top-k query only needs 32 messages on average. The figure also shows that the number of query terms has no significant impact on the performance of the distributed top-k query, the performance of two, three and four term query is very close to each other.

6.6.4.2 Query Response Time

While the above simulation studies the performance in the number of message units, the query response time is more concerned for a real world application, specially in a large scale network. Next, we estimate the distributed query time. Since the *KeyIP* is a resource-rich computer, we ignore the processing time at the *KeyIP*. The message transmission time thus dominates the query delay. Our experiment shows the average transmission time T_p for a packet with 68 byte payload

is 11.4ms.

The message complexity of the distributed top-k query is $2(m+k)$, where m is the number of *IP*s having searched terms, and k is the number of top answers the user is looking for. Without loss of generality, we let k be 10 so that the user always wants the top-10 answers. To determine the value of m, we consider the following two scenarios. We let m be 20 as the search items are popular, and let m be 5 as the items are non-popular. We further denote D as the average hop distance between *IP*s and the *KeyIP*. A larger D indicates a larger network.

As we described in Alg. 6, the *KeyIP* first queries m *IP*s for their top-ranked answers. After collecting the responses from m *IP*s, the *KeyIP* sequentially queries up to k *IP*s (that have higher scores than the rest $m-k$) until the top-k answers are found. In the worst case, the message complexity is $2(m+k)$. Combining all components, our estimation of the query time for the distributed top-k query is:

$$T_{query} = m \cdot T_P + 2 \cdot D \cdot T_p + 2 \cdot D \cdot T_p \cdot k. \tag{6.3}$$

The first term is the time to transmit m query messages by the *KeyIP*. Note the *KeyIP* can continuously send out the messages without waiting for the reply for the specific *IP*. The second term indicates the time duration of the average response time for the query send by the *KeyIP*. The third term is the time taken for the *KeyIP* to query up to k *IP*s.

For the naive top-k query scheme, each *IP* has to replies k messages, each of which carries an answer. The total query time can be expressed as:

$$T_{query} = m \cdot T_p + D \cdot T_p + m \cdot k \cdot T_p \cdot D. \tag{6.4}$$

The first term is the same as in Eq. 6.3. The second term indicates the time taken for the query to

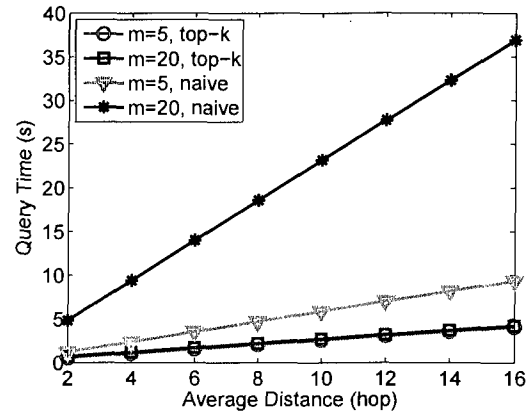


Figure 6.16: Query response time of distributed top-k query.

reach IPs. The third term is the transmission time for the IPs to send k messages over D hops.

The estimated query response time for the both schemes is shown in Fig. 6.16. We find that our distributed top-k query scheme is much more efficient than the naive scheme. When the average hop distance increases from 2 to 16, the query response time of our distributed top-k query grows linearly from 0.7s to 4.2s. The response time of the naive scheme, on the other hand, grows from 4.8s to more than 36s. We also find the value of m has very little affect toward the query response time of our distributed top-k scheme. However, the m value has a large impact on the response time of the naive scheme because every IP out of m members has to respond the query from $KeyIP$ with k answers.

6.6.4.3 Impact of IDF on Query Accuracy

As we discussed in Section 6.3.3, we choose to use the global IDF to get accurate merged object rankings. Now, we are interested in the accuracy comparison between the ranking using the global IDF and that using the local IDF. We perform the similar simulation as we described in Section 6.6.3.3. We still use the previous data set with 80 objects that form the testbed with eight

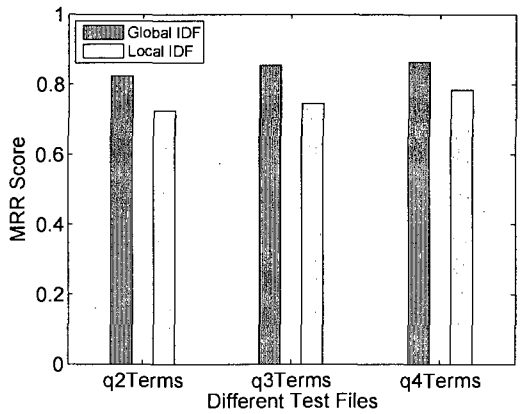


Figure 6.17: Query accuracy of distributed top-*k* query.

*IP*s (each *IP* has 10 objects). We also use MRR as the metric, and generate three different test files, *q2Term*, *q3Term* and *q4Term*, each of which contains 20 questions with certain degree of ambiguity.

The test results are illustrated in Fig. 6.17. We find the MRR scores of the rankings by the global *IDF* are consistently higher than those of the rankings by the local *IDF*s. The reason can be explained as follows. Once the data set (80 objects) are fixed, the global *IDF*s are determined and will not be affected by the object distribution in *IP*s. Therefore, the merged rankings by using the global *IDF* are also determined. However, the local *IDF* values depend on the local document frequency (*DF*), which is affected by the distribution of the objects. Thus, the model answer may not top the merged rankings by using the local *IDF* if the matching query term has a small local *IDF* value (due to a large local *DF*) so that the matching term cannot contribute much weight in the overall score.

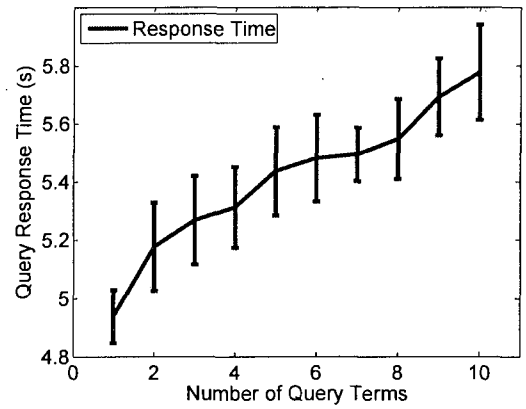


Figure 6.18: User perceived private object query response time.

6.6.5 Security Overhead for User Query

Finally, we add the authentication module to the *IP* and test the performance of private object query. We used an ECC public key cryptosystem designed for TelosB motes. Our extensive optimization allows TelosB mote to efficiently perform ECC public key operation. Our experiment shows it only takes 1.4s to do a point multiplication. To the best of our knowledge, this is the best ECC performance achieved on TelosB motes by academic implementations. When the user queries the private objects, the user's identity and access privilege have to be verified. The 160-bit ECC based authentication is performed for the verification purpose. The user query response time is presented in Fig. 6.18. To query a private object, the user waits around 4.9s to pass the authentication check. Obviously, the authentication time dominates the overall response time. This is because that the ECC based authentication scheme requires 3 ECC point multiplications, which contribute more than 90% of the overall delay.

6.7 System Limitations

Communication Reliability. From our experience building the prototype experiments, we notice that message dropping is a major concern. Not only does it happen when *IP*s report to the *KeyIP* for record updates, message drop is also found during *IP* beacon sending and replying, which may cause object missing while moving. That is, the former *IP* has already deleted a leaving object (due to the timeout), but the new *IP* does not catch it due to packet loss during beacon sending and reply. Therefore, it is desirable and imperative to implement a reliable communication mechanism in the application layer or, if possible, in the transport layer, such as acknowledge and retransmission.

System Scalability. Our Snoogle system design does not limit the number of *IP*s and *KeyIP*s. When the number *IP*s is getting larger (e.g., the network spans several buildings in a district), we can certainly deploy multiple *KeyIP*s, each of which can serve a number of *IP*s in a certain area as we discussed previously. Since *KeyIP*s are resource-rich devices, the information sharing and exchanging among *KeyIP*s is beyond the scope of this dissertation.

Given the limited hardware resources, an individual *IP* domain has its capacity limit. As we have discussed in the previous section, each object term takes four-byte flash space. Considering 20 terms for each objects on average and 1MB flash memory space, each *IP* in Snoogle can support more than 10,000 objects in its neighborhood. Given the limited sensor transmission range (normally 100ft), we believe the scale of 10,000 objects can support most applications. When the amounts of object are getting larger (within a certain *IP*), however, the query accuracy may be affected for a query with multiple terms. The reason is that the RAM space in *IP* is very limited in our prototype system and cannot hold all intermediate results while calculating the score of relevancy given a multi-term query. One possible solution is to select a more powerful *IP* device

with a larger RAM size for the object populated areas.

Mobility Support. While Snoogle supports the search for a mobile object, it does not track a moving object in real time. Due to the power constraints in both *IP*s and object sensors, Snoogle cannot afford very frequent beacon or timer mechanism so that an *IP* may not immediately detect a moving object in its neighborhood. Therefore, a snapshot of the system view does not necessarily give accurate moving object locations. However, once the object stops at a certain place for a certain amount of time (e.g., a beacon cycle), the *IP* at that location will capture the object and update *KeyIP* with the new indexed items. Obviously, a large number of moving objects will trigger many index updates from *IP*s to the *KeyIP*, which may cause much battery drain and could be a concern of the *IP* life-cycle. We currently assume there are limited moving objects in the system and reserve the *IP* power management in our future work.

6.8 Conclusion

In this chapter, we presented Snoogle, an information retrieval system built on sensor networks. Our system reduces communication costs by employing compressed Bloom filter on sensor data, while maintaining low rates of false positive. We also introduced a flexible security method using public key cryptography that protects user privacy. Our current implementation incurs a five second latency. Currently we are working on different techniques to further reduce the latency for security.

Algorithm 6 Distributed Top-k Query Algorithm

1: Input: k IPs: $IP_1, IP_2, \cdots, IP_m$

2: Output: top-k answers: $Obj_1, Obj_2, \cdots, Obj_k$

3: Each IP sorts its objects in descending order of the weights

4: **for** from $i = 1$ to $i = m$ **do**

5: query IP_i for the top answer; each IP removes the first object from the sorted list and sends it to user

6: store the top answer ta_i and its associated weight in an array: $a[i].obj = ta_i, a[i].weight = weight(ta_i), a[i].ip = IP_i$

7: **end for**

8: set the number of committed objects, num_commit=1

9: **while** num_commit $< k$ **do**

10: sort the array in descending order of weight so that $a[1].weight \geq a[2].weight \geq \cdots a[m].weight$

11: send $a[2].weight$ and num_commit to IP $a[1].ip$

12: IP $a[1].ip$ removes from its sorted list a list of objects (say l of them) such that the last object has the highest weight less than $a[2].weight$, say w

13: IP $a[1].ip$ sends the first min(l, $k-$ num_commit)

14: commit all retrieved objects with weight greater than $a[1].weight$, change the value of num_commit, set $a[1].weight = w$

15: **end while**

16: return all the committed objects $Obj_1, Obj_2, \cdots, Obj_k$

Chapter 7

Conclusion

This dissertation explores the privacy issues in Cyber-Physical Systems (CPS). CPS is a new Computer Science realm in that a CPS system composes of myriads of interacting low cost computing elements that are deeply embedded in the physical world. While CPS brings the great system performance enhancement in terms of response time, adaptability, reliability, safety, efficiency, and so on, privacy becomes a big concern as these intelligent computing elements become deeply pervasive in human societies. This dissertation proposes a privacy preserving framework to preserve the privacy in CPS applications. The privacy preserving framework composes of two privacy solutions: data privacy solution and location privacy solution.

The data privacy solution not only prevents the unauthorized access to the sensitive data, also protects the network by filtering out the unauthorized messages. The building block of the data privacy solution is WM-ECC, an efficient ECC implementation for resource constrained devices. WM-ECC tops the performance in running time among the public available ECC implementations. In particular, the ECC signature generation is only 0.77s on Tmote Sky sensors and 1.12s on MICAz sensors. WM-ECC makes computing intensive public key exponentiations become

practical on extremely resource constrained low-power devices.

Based on WM-ECC, this dissertation proposes a number of security schemes to protect the data content and prevents DoS attacks in the network. In particular, the pairwise key establishment scheme enable neighboring sensors to efficiently negotiate secret keys. The extensive evaluation shows that the PKC-based schemes achieve better performance in memory overhead, message complexity and group key establishing time than the symmetric-key based schemes proposed in prior work. The user access control schemes allow a user to query a network node either locally or remotely. Unlike the local access control scheme based on symmetric-key schemes, the PKC-based solution is resilient to user collusion attacks. Further, the remote access control leverages the threshold cryptography to defend against the adversary's DoS attacks. While the above schemes provide the end-to-end message security in the network, the proposed PDF scheme protects the network by blocking the unauthorized messages (forged by the adversary). PDF leverages the threshold cryptography and arranges a number of nodes to jointly generate a system signature so that any intermediate node can easily verify. All above schemes are implemented on commodity sensor motes. The extensive experiments show the ECC-based security schemes are practical for real world applications.

To address the location privacy issues, this dissertation presents privacy-ware routing schemes under a complete adversary model. When the adversary has limited radio detection capability and can only monitor the traffic within a small network area, we design the routing schemes that distribute messages to a geographic area with certain energy constraint so that the adversary's traceback time can be maximized. An optimization problem is formulated to study the adversary's maximum traceback time under the routing energy constraints. When the adversary has the global monitoring capability, a different routing strategy is designed to prevent the adversary

from profiling the traffic. To save the energy, a random message transmission schedule scheme is presented to limit the message transmission period to minimum.

Finally, the proposed privacy preserving frame is applied to Snoogle, a sensor-based searching system for the physical world. On top of the framework, a versatile security and privacy scheme is integrated in the system that enables flexible user access control in searching for private items. In addition, Snoogle also incorporates a number of techniques including communication compression, distributed top-k query and information retrieval. The combination of above techniques with the pervasively embedded and resource constrained devices makes Snoogle a practical and powerful search engine for the physical world, which is validated by the combination of real world experiments and scalability simulations.

Bibliography

[1] JALAL AL-MUHTADI, ROY CAMPBELL, APU KAPADIA, M. DENNIS MICKUNAS, AND SEUNG YI. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *ICDCS*, pages 65–74, Vienna, Austria, Jul 2002.

[2] A.D. AMIS, R. PRAKASH, T.H.P. VUONG, AND D.T. HUYNH. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In *INFOCOM*, 2000.

[3] S. BANDYOPADHYAY AND E. COYLE. An Energy-efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks. In *INFOCOM*, 2003.

[4] S. BANNERJEE AND S. KHULLER. A Clustering Scheme for Hierarchical Control in Multihop Wireless Networks. In *INFOCOM*, 2001.

[5] S. BASAGNI. Distributed Clustering Algorithm for Ad-hoc Networks. In *I-SPAN*, 1999.

[6] RAMESH BHANDARI. Optimal Physical Diversity Algorithms and Survivable Networks. In *ISCC*, Washington, DC, USA, 1997. IEEE.

[7] BURTON BLOOM. Space/time Trade-offs in Hash Coding with Allowable Errors. *Communications of ACM*, 13(7):422–426, July 1970.

[8] BLA BOLLOBS. *Random Graphs.* Acadamic Press Inc, 1985.

[9] D. BONEH AND M. FRANLIN. Identity-Based Encryption from the Weil Pairing. In *CRYPTO*, pages 213–229, Springer-Verlag, 2001.

[10] NIRUPAMA BULUSU, DEBORAH ESTRIN, LEWIS GIROD, AND JOHN HEIDEMANN. Scalable coordination for wireless sensor networks: Self-configuring localization systems. In *Proceedings of the 6th IEEE International Symposium on Communication Theory and Application*, July 2001.

[11] NIRUPAMA BULUSU, JOHN HEIDEMANN, AND DEBORAH ESTRIN. Gps-less low cost outdoor localization for very small devices, October 2000.

[12] JAMIE CALLAN. Distributed information retrieval. In *In: Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, 2000.

[13] D. CARMAN, B. MATT, P. KRUUS, D. BALENSON, AND D. BRANSTAD. Key Management in Ditributed Sensor Networks. In *DARPA Sensor IT Workshop*, 2000.

[14] CERTICOM. Code and Cipher. *Certicom's Bulletin of Security and Cryptography*, 1(3):1–5, 2004.

[15] H. CHAN AND A. PERRIG. PIKE: Peer Intermediaries for Key Establishment in Sensor Networks. In *INFOCOM*, Miami, FL, March 2005.

[16] H. CHAN, A. PERRIG, AND D. SONG. Random Key Predistribution Schemes for Sensor Networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, Berkeley, California, May 2003.

[17] M. CHATTERJEE, S. K. DAS, AND D. TURGUT. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. In *Cluster Computing*, 2002.

[18] D. CHAUM. Untraceable electronic mail, return addresses and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[19] D. CHAUM. The dining cryptographers problem: Unconditional sender and receipient untraceability. *J. Cryptol.*, 1(1):67–75, 1988.

[20] ROGER CLARKE. Introduction to dataveillance and information privacy, and definitions of terms. August 1997.

[21] H. COHEN, A. MIYAJI, AND T. ONO. Efficient elliptic curve exponentiation. In *Advances in Crytology-Proceedings of ICICS'97, Lecture Notes in Computer Science*, pages 282–290, Springer-Verlag, 1997.

[22] H. COHEN, A. MIYAJI, AND T. ONO. Efficient elliptic curve exponentiation using mixed coordinates. In *ASIACRYPT: Advances in Cryptology*, 1998.

[23] N. CORRELL, A. BOLGER, M. BOLLINI, B. CHARROW, A. CLAYTON, F. DOMINGUEZ, K. DONAHUE, S. DYAR, L. JOHNSON, H. LIU, A. PATRIKALAKIS, J. SMITH, M. TANNER, L. WHITE, AND D. RUS. Building a Distributed Robot Garden. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St Louis, MO, Oct 2009.

[24] HUI DAI, MICHAEL NEUFELD, AND RICHARD HAN. ELF: an efficient log-structured flash file system for micro sensor nodes. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 176–187, New York, NY, USA, 2004. ACM Press.

[25] J. DENG, R. HAN, AND S. MISHRA. A Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks. In *IPSN*, pages 349–364, Palo Alto, California, 2003.

[26] W. DU AND J. DENG. A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In *ACM CCS*, 2003.

[27] X. DU. Detection of Compromised Sensor Nodes in Heterogeneous Sensor Networks. In *ICC*, pages 1446–1450, Beijing, China, 2008.

[28] L. ESCHENAUER AND V.D. GLIGOR. A Key-management Scheme for Distributed Sensor Networks. In *ACM CCS*, November 2002.

[29] RONALD FAGIN, AMNON LOTEM, AND MONI NAOR. Optimal aggregation algorithms for middleware. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 102–113, 2001.

[30] L. FAN, P. CAO, J. ALMEIDA, AND A. BRODER. Summary cache: A scalable wide-area web cache sharing protocol. In *SIGCOMM*, 1998.

[31] A. C. FERREIRA, M. A. VILAA, L. B. OLIVEIRA, H. C. WONG, AND A. A. LOUREIRO. . In *Networking-ICN*, pages 449–458, 2005.

[32] ARMANDO FOX AND STEVEN D. GRIBBLE. Security on the Move: Indirect Authentication Using Kerberos. In *Mobicom*, New York, November 1996.

[33] JAMES C. FRENCH, ALLISON L. POWELL, JAMES P. CALLAN, CHARLES L. VILES, TRAVIS EMMITT, KEVIN J. PREY, AND YUN MOU. Comparing the performance of database selection algorithms. In *ACM SIGIR on Research and Development in Information Retrieval*, 1999.

[34] XINWEN FU, YE ZHU, BRYAN GRAHAM, RICCARDO BETTATI, AND WEI ZHAO. On Flow Marking Attacks in Wireless Anonymous Communication Networks. In *ICDCS*, pages 493–503, 2005.

[35] RAGHU K. GANTI, PRAVEEN JAYACHANDRAN, TAREK F. ABDELZAHER, AND JOHN A. STANKOVIC. SATIRE: a software architecture for smart attire. In *MobiSys*, 2006.

[36] IAN GOLDBERG, DAVID WAGNER, AND ERIC A. BREWER. Privacy-enhancing technologies for the Internet. In *IEEE COMPCON*, Feb 1997.

[37] J. GROBCHADL. The Chinese Remainder Theorem and its application in a high-speed RSA crypto chip. In *ACSAC*, page 384, 2000.

[38] MARCO GRUTESER, GRAHAM SCHELLE, ASHISH JAIN, RICK HAN, AND DIRK GRUNWALD. Privacy-Aware Location Sensor Networks. In *HotOS IX*, 2003.

[39] L. GU AND J. STANKOVIC. t-kernel: Providing Reliable OS Support to Wireless Sensor Networks. In *ACM SenSys*, Boulder, CO, Nov. 2006.

[40] V. GUPTA, MILLARD M, S. FUNG, Y. ZHU, N. GURA, H. EBERLE, AND S. C. SHANTZ. Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet. In *PERCOM*, Kauai, Hawaii, Mar 2005.

[41] NILS GURA, ARUN PATEL, ARVINDERPAL WANDER, HANS EBERLE, AND SHEUELING CHANG SHANTZ. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *CHES*, Boston, Aug 2004.

[42] ALLAN R. HAMBLEY. *Electrical Engineering : Principles & Applications*. Prentice Hall, 2004. 3rd Edition.

[43] D. HANKERSON, A. MENEZENS, AND S. VANSTONE. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[44] QUN LI HAODONG WANG, CHIU C. TAN. Snoogle: A Search Engine for the Physical World. In *IEEE International Conference on Computer Communications (INFOCOM)*, Phoenix, AZ, April 2008.

[45] W.R. HEINZELMAN, A. CHANDRAKASAN, AND H. BALADRISHNAN. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transaction on Wireless Communication*, 1(4):660–670, October 2002.

[46] HOPPYTRON.COM. Doppler Direction Finder Kit.

[47] BRET HULL, VLADIMIR BYCHKOVSKY, KEVIN CHEN, MICHEL GORACZKO, ALLEN MIU, EUGENE SHIH, YANG ZHANG, HARI BALAKRISHNAN, AND SAMUEL MADDEN. Cartel: A distributed mobile sensor computing system. In *SenSys*, 2006.

[48] CROSSBOW TECHNOLOGY INC. Wireless Sensor Networks. *http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.*

[49] C. INTANAGONWIWAT, R. GOVINDAN, AND D. ESTRIN. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *MOBICOM*, Boston, MA, August 2000.

[50] M. JACOBSSON. Flash Mixing. In *Proceedings of Symposium on Principles of Distributed Computing*, May 1999.

[51] BERNARD J. JANSEN, AMANDA SPINK, JUDY BATEMAN, AND TEFKO SARACEVIC. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 1998.

[52] SHU JIANG, NITIN H. VAIDYA, AND WEI ZHAO. Routing In Packet Radio Networks to Prevent Traffic Analysis. In *Proceedings of the IEEE Information Assurance and Security Workshop*, West Point, NY, July 2000.

[53] P. KAMAT, Y. ZHANG, W. TRAPPE, AND C. OZTURK. Enhencing Source-Location Privacy in Sensor Network Routing. In *ICDCS*, Columbus, Ohio, June 2005.

[54] C. KARLOF, N. SASTRY, AND D. WAGNER. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *SENSYS*, Baltimore, MD, Nov 2004.

[55] B. KARP AND H. KUNG. Greedy Perimeter Stateless Routing. In *MOBICOM*, 2000.

[56] C. K. KOC. High-Speed RSA Implementation. In *RSA Laboratories TR201*, Nov 1994.

[57] EDWARD LEE. Cyber Physical Systems: Design Challenges. In *University of California, Berkeley Technical Report No. UCB/EECS-2008-8*, January 2008.

[58] JINYANG LI, BOON THAU LOO, JOSEPH M. HELLERSTEIN, M. FRANS KAASHOEK, DAVID KARGER, AND ROBERT MORRIS. On the feasibility of peer-to-peer web indexing and search. In *IPTPS*, 2003.

[59] AN LIU AND PENG NING. Tinyecc: Elliptic curve cryptography for sensor networks. Sept 15 2005.

[60] D. LIU AND P. NING. Establishing Pairwise Keys in Distributed Sensor Networks. In *ACM CCS*, Washington, DC, October 2003.

[61] D. LIU AND P. NING. Improving Key Pre-Distribution with Deployment Knowledge in Static Sensor Networks. *ACM Transaction on Sensor Networks*, 1(2):204–239, November 2005.

[62] D.J. MALAN, M. WELSH, AND M.D. SMITH. A Public-key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography. In *The First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA, October 2004.

[63] GAURAV MATHUR, PETER DESNOYERS, DEEPAK GANESAN, AND PRASHANT SHENOY. Ultra-low Power Data Storage for Sensor Networks. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 374–381, New York, NY, USA, 2006. ACM Press.

[64] KIRAN METHA, DONGGANG LIU, AND MATTHEW WRIGHT. Location Privacy in Sensor Networks Against a Global Eavesdropper. In *ICNP*, Beijing, China, Oct 2007.

[65] M. MITZENMACHER. Compressed bloom filters. In *Proc. of the 20th Annual ACM Symposium on Principles of Distributed Computing*, 2001.

[66] P. MONTGOMERY. Modular Multiplication Without Trial Division. *Mathematics of Communication*, 44(170):519–521, April 1985.

[67] D. MOORE, J. LEONARD, D. RUS, AND S. TELLER. Robust distributed network localization with noisy range measurements, 2004.

[68] F. MORAIN AND J. OLIVOS. Speeding up the computations on an elliptic curve using addition-subtraction chains. *Theoretical Informatics and Applications*, 24:531–543, 1990.

[69] MYSAIFU. *www2s.biglobe.ne.jp/ dat/java/project/jvm/*.

[70] LAMA NACHMAN, RALPH KLING, ROBERT ADLER, JONATHAN HUANG, AND VINCENT HUMMEL. The Intel®mote platform: a bluetooth-based sensor network for industrial monitoring. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.

[71] B. CLIFFORD NEUMAN AND THEODORE TS'O. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9):33–38, September 1994.

[72] J. NEWSOME, E. SHI, D. SONG, AND A. PERRIG. The Sybil Attack in Sensor Networks: Analysis and Defenses. In *IPSN*, 2004.

[73] NIST. Key Management Guideline. In *Workshop Document (DRAFT)*, October 2001.

[74] CELAL OZTURK, YANYONG ZHANG, AND WADE TRAPPE. Source-Location Privacy for Networks of Energy-Constrained Sensors. In *WSTFEUS*, 2004.

[75] A. PERRIG, R. SZEWCZYK, V. WEN, D. CULLER, AND D. TYGAR. SPINS: Security Protocols for Sensor Networks. *ACM/Kluwer Wireless Networks Journal (WINET)*, September 2002.

[76] N. B. PRIYANTHA, A. CHAKRABORTY, AND H. BALAKRISHNAN. The cricket location-support system. In *the 6th annual international conference on Mobile computing and networking*, 2000.

[77] M. REED, P. SYVERSON, AND D. GOLDSCHLAG. Anonymous Connections and Onion Routing. In *IEEE JSAC Copyright and Privacy Protection*, 1998.

[78] MICHAEL REITER AND AVIEL RUBIN. Crowds: Anonymity for Web Transaction. In *ACM Transaction on Information and System Security 1(1)*, June 1998.

[79] KUI REN AND WENJING LOU. Privacy enhanced access control in pervasive computing environments. In *Proceedings of BroadNet05*, October 2005.

[80] A. SAVVIDES, C. HAN, AND M. STRIVASTAVA. Dynamic fine-grained localization in ad-hoc networks of sensors. In *the 7th annual international conference on Mobile computing and networking*, 2001.

[81] CHIRAG SHAH AND W. BRUCE CROFT. Evaluating high accuracy retrieval techniques. In *SIGIR*, 2004.

[82] A. SHAMIR. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[83] S. CHANG SHANTZ. From Euclid's GCD to Montgomery Multiplication to the Great Divide. In *Technical report, Sun Microsystems Laboratories TR-2001-95*, June 2001.

[84] M. SHAO, Y. YANG, S. ZHU, AND G. CAO. Towards Statistically Strong Source Anonymity for Sensor Networks. In *IEEE INFOCOM*, Phoenix, AZ, Apr 2008.

[85] ANAND SRINIVAS AND EYTAN MODIANO. Finding Minimum Energy Disjoint Paths in Wireless Ad-Hoc Networks. *Wireless Network*, 11(4):401–417, 2005.

[86] J.W. SUURBALLE. Disjoint Paths in a Network. *Network*, 4(2):125–145, 1974.

[87] J.W. SUURBALLE AND R.E. TARJAN. A Quick Method for Finding Shortest Pairs of Disjoint Paths. *Network*, 14(2):325–336, 1984.

[88] TINYOS. TinyOS 1.1.15. *http://www.tinyos.net*, 2006.

[89] P. TRAYNOR, H. CHOI, G. CAO, S. ZHU, AND T.L. PORTA. Establishing Pair-wise Keys in Heterogeneous Sensor Networks. In *INFOCOM*, Barcelona, Spain, April 2006.

[90] P. TRAYNOR, R. KUMAR, H. B. SAAD, G. CAO, AND T. L. PORTA. LIGER: Implementing Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks. In *MOBISYS*, Uppsala, Sweden, June 2006.

[91] HAODONG WANG AND QUN LI. Distributed User Access Control in Sensor Networks. In *the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, San Francisco, CA, June 2006.

[92] HAODONG WANG AND QUN LI. Efficient Implementation of Public Key Cryptosystems on MICAz Motes. In *the 8th International Conference on Information and Communications Security (ICICS, short paper)*, Raleigh, NC, Nov 2006.

[93] HAODONG WANG AND QUN LI. Efficient Implementation of Public Key Cryptosystems on Mote Sensors (Short Paper). In *International Conference on Information and Communication Security (ICICS), LNCS 4307*, pages 519–528, Raleigh, NC, Dec. 2006.

[94] HAODONG WANG AND QUN LI. Achieving robust message authentication in sensor networks: A public-key based approach. *ACM Journal of Wireless Networks*, 2009.

[95] HAODONG WANG, BO SHENG, AND QUN LI. Efficient Implementation of Public Key Cryptosystems on MICAz and TelosB motes. In *Technical Report WMCS-2006-7*, College of William and Mary, 2006.

[96] HAODONG WANG, BO SHENG, AND QUN LI. Elliptic curve cryptography based access control in sensor networks. *International Journal on Security and Networks*, 1(2), 2006.

[97] HAODONG WANG, BO SHENG, AND QUN LI. Privacy-aware routing in sensor networks. *Computer Networks*, 53(9):1512–1529, 2009.

[98] HAODONG WANG, BO SHENG, CHIU C. TAN, AND QUN LI. WM-ECC: an Elliptic Curve Cryptography Suite on Sensor Motes. In *Technical Report WMCS-2007-11*, College of William and Mary, 2007.

[99] HAODONG WANG, BO SHENG, CHIU C. TAN, AND QUN LI. Comparing Symmetric-key and Public-key based Schemes in Sensor Networks: A Case Study for User Access Control. In *the 28th International Conference on Distributed Computing Systems (ICDCS)*, Beijing, June 2008.

[100] M. WRIGHT, M. ADLER, B. LEVINE, AND C. SHIELDS. An Analysis of the Degradation of Anonymous Protocols. In *Proc. ISOC Symposium Network and Distributed System Security (NDSS)*, pages 38–50, February 2002. Outstanding Paper Award.

[101] H. YANG, F. YE, Y. YUAN, S. LU, AND W. ARBAUGH. Toward Resilient Security in Wireless Sensor Networks. In *MOBIHOC*, Urbana-Champaign, IL, May 2005.

[102] F. YE, A. CHEN, S. LU, AND L. ZHANG. A scalable solution to minimum cost forwarding in large sensor networks. In *Tenth International Conference on Computer Communications and Networks*, pages 304–309, 2001.

[103] F. YE, S. LU, AND L. ZHANG. Gradient broadcast: a robust, long-live large sensor network. In *Tech. Report, Computer Science Department, UCLA*, 2001.

[104] F. YE, H. LUO, S. LU, AND L. ZHANG. Statistical En-Route Filtering of Injected False Data in Sensor Networks. In *INFOCOM*, 2004.

[105] O. YOUNIS AND S. FAHMY. Distributed Clustering in Ad-hoc Sensor Networks. In *INFOCOM*, 2004.

[106] ZHEN YU AND YONG GUAN. A Dynamic En-route Scheme for Filtering False Data in Wireless Sensor Networks. In *INFOCOM*, Spain, April 2006.

[107] DEMETRIOS ZEINALIPOUR-YAZTI, SONG LIN, VANA KALOGERAKI, DIMITRIOS GUNOPULOS, AND WALID A. NAJJAR. MicroHash: An Efficient Index Structure for Flash-Based Sensor Devices. In *FAST*, 2005.

[108] Q. ZHANG, T. YU, AND P. NING. A framework for identifying compromised nodes in wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(3):1–37, 2008.

[109] W. ZHANG, H. SONG, S. ZHU, AND G. CAO. Least Privilege and Privilege Deprivation: Towards Tolerating Mobile Sink Compromises in Wireless Sensor Networks. In *MOBIHOC*, Chicago, IL, May 2005.

[110] Y. ZHANG, W. LIU, W. LOU, AND Y. FANG. Location-based Compromise-tolerant Security Mechanisms for Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications (Special Issue on Security in Wireless Ad Hoc Networks)*, 24(2):247–260, Feb 2006.

[111] Y. ZHANG, Y. YANG, L. JIN, AND W. LI. Locating Compromised Sensor Nodes through Incremental Hashing Authentication. In *DCOSS*, San Francisco, CA, 2006.

[112] YANCHAO ZHANG, WEI LIU, WENJING LOU, AND YUGUANG FANG. Securing Sensor Networks with Location-based Keys. In *WCNC*, New Orleans, Louisiana, March 2005.

[113] YANCHAO ZHANG, WEI LIU, WENJING LOU, AND YUGUANG FANG. MASK: Anonymous on-demand Routing in Mobile ad hoc Networks. *IEEE Transactions on Wireless Communications*, 5(9), 2006.

[114] S. ZHU, S. SETIA, AND S. JAJODIA. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In *ACM CCS*, Washington D.C., October 2003.

[115] S. ZHU, S. SETIA, S. JAJODIA, AND P. NING. An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

VITA

Haodong Wang

Haodong Wang received his Bachelor of Engineering degree in Electronic Engineering from Tsinghua University, China, and his Master of Science degree in Electric Engineering from Penn State University, University Park, PA. He started his PhD study in Computer Science Department at the College of William and Mary in 2003. He has become a PhD candidate since 2004. His research interests include security and privacy support for wireless embedded devices and systems, efficient data management, storage, search and retrieval in networked embedded systems, efficient 802.11 data communication in vehicular environments.