Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

JigDFS for Implementing Secure Container Communities

Jiang Bian
Department of Computer Science
University of Arkansas at Little Rock
Little Rock, U.S.
jxbian@ualr.edu

Remzi Seker
Department of Computer Science
University of Arkansas at Little Rock
Little Rock, U.S.
rxseker@ualr.edu

Srini Ramaswamy
Department of Computer Science
University of Arkansas at Little Rock
Little Rock, U.S.
srini@ualr.edu

***Abstract*—The globalization of the economy has given increase to the number of shipping containers. The large number of shipping containers arriving at major ports makes inspecting every container impractical and therefore bears significant risks. This paper describes a solution that would monitor containers' integrity from the originating port to the destination port. Should there be an intrusion to the containers en route, the system will report the activity to its home station. The proposed system is envisioned to be secure and intrusion resistant. The proposed approach uses an implementation of Jigsaw Distributed File System (JigDFS) to further protect the container communities. This paper is the full version of extended abstract published at CSIIRW-09 [1].**

***Index Terms*—Cargo Container Security, Secure Distributed Systems**

I. Introduction

This paper is full version of the extended abstract published at the fifth Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW-09) [1]. The security of cargo containers used for international trade has received attention due to potential threats. Right after the terrorist events of September 11, 2001, the U.S. Transportation Security Administration (TSA) worked closely with Congress to significantly strengthen security in air cargo through the 9/11 Bill. In the bill, the TSA is to screen 50 percent of all cargo on passenger planes by February 3, 2009, and 100 percent within three years. Similarly, the U.S. Customs and Border Protection (CBP) developed the Container Security Initiative (CSI) that aims to "extend the zone of security outward so that American borders are the last line of defense, not the first." The CSI program aims to pre-screen containers at the port of origin to identify the high-risk containers. Although such measures are beneficial, they leave room for improvement. One such improvement is to assure integrity of containers en-route. The incidents of piracy that took place off the coast of Somalia recently were important in showing the criticality of ensuring containers' safety en-route. A container compromised en-route can contain an agent that is harmful to a nation's well-being. This paper proposes the Container Community Wireless Sensor Network (CC-WSN), in which the integrity of smart containers are monitored and these smart containers are wirelessly linked to form a neighborhood watch program.

Smart containers are equipped with necessary sensors that can detect any intrusion event. When an intrusion into a container is detected, the container will report this event to its neighbors. The containers in a container community exchange their status to increase the level of information redundancy. The resulting CC-WSN is attack hardened against attackers' need to cover their tracks.

An intrusion event message is first divided into n slices according to a chosen Information Dispersal Algorithm (IDA). Then each slice is sent to a different neighboring node. An investigator can reconstruct the original event message by decoding the IDA. The main characteristic of an IDA is that the original message can be recovered as long as k segments are still alive, where $k \ll n$, where n is the actual number of segments. To an attacker, each segment appears to be a chunk of meaningless random data. Even if an attacker manages to destroy (n - k - 1) segments of an event message, the alert will still be available to the investigator.

Wireless signals are prone to jamming attacks and being captured off air. The CC-WSN has an Active Status Polling (ASP) mechanism to address the scenario of disrupting communications and meanwhile intruding into containers. Since containers are stacked on top of one another and close to one another, an alternate communication channel can be considered as a fall-back channel in the presence of heavy wireless jamming. Each node in the CC-WSN will actively query its neighbors' security status and if no response or a false response is received, the querying node will report an abnormal event for the queried node. Authentication is important in ASP communications to prevent man-in-the-middle attacks. Section III discusses the man-in-the-middle attacks.

The Jigsaw Distributed File System (JigDFS) [2] is utilized to further harden the CC-WSN system against attacks. JigDFS is a secure distributed file system originally designed to provide plausible deniability to help protecting users' privacy. JigDFS also uses an IDA to achieve high redundancy which helps the system's fault tolerance. The file segments are encrypted recursively using keys generated by a hashed-key chain algorithm to make tracing of the origin of each file or file segment more difficult. In this version of CC-WSN, a table of containers with their status is stored distributively among container nodes using JigDFS. We will discuss the advantages of using JigDFS in CC-WSN further in Section III. In general, JigDFS helps concealing the path to each file slice and makes it difficult for an attacker to trace every slice of the information and wipe the intrusion event from the system thoroughly.

The rest of the paper is organized into four sections. Section II gives a brief overview of related background such as the Information Dispersal Algorithm, the concept of the JigDFS, etc. In Section III, we present the overall design of the proposed CC-WSN system. The last section presents the conclusions.

II. Related Work

A. Wireless Sensor Networks and Security

Wireless Sensor Networks (WSN) [3] [4] have wide range of applications [5] [6]. Sensor nodes are deployed in a region to track and monitor parameters of interest in the environment.

There is a wide spectrum of security threats for wireless sensor networks. These threats include eavesdropping, spoofing, impersonation, and denial-of-service (DOS) attacks. In general, a wireless sensor node is resource limited, especially in terms of battery life and computation power. Limited resources makes it impractical to use strong encryption and authentication at every node. Public-key infrastructure is normally considered resource intensive for WSN communications. However, Watro et. al. have developed a lightweight PKI system for WSNs named TinyPK [7], which can be utilized in the CC-WSN depending on the sensitivity of the cargo. Mostly, in both CC-WSN and JigDFS, symmetric encryption and one-way hash function are used to protect the information. Both of these procedures are computationally cheaper than assymmetric encryption. However, the ASP mechanism in CC-WSN utilizes pre-deployed public/private key pairs to generate and verify the digital signature of ASP responses so that the authenticity of messages can be ensured. Nevertheless, the key pairs are pre-computed and pre-deployed into each sensor node at the port of origin during pre-screening phase by using a computer with enough resources.

B. Information Dispersal Algorithm and Jigsaw Distributed File System (JigDFS)

Both the CC-WSN itself and the JigDFS use an Information Dispersal Algorithm (IDA) to not only increase the redundancy of the information, but also help assuring the secrecy of data. The IDA is a special use of *erasure codes* also referred as *forward error correction* (FEC) codes, first introduced by Michael O. Rabin [8] to design a fault-tolerant and transmission efficient information storage system. The basic idea of erasure codes is to add redundancy to the original data; this extra information allows the receiver or reader to detect and correct data errors without the need to ask the sender to resend the message. JigDFS uses an optimized Reed-Solomon (R-S) code [9] known as Cauchy Reed-Solomon code [10]. The encoder of an IDA splits the original file/data into n segments and adds error correction codes into each segment. While recovering, however, only k segments are needed to reconstruct the complete file. Both the n and k are set based on the configuration of the IDA, where $k \ll n$. Although the overall size of the original message increases due to the added redundancy, the length of each packet sent to individual node will be reduced. A smaller packet size in wireless communication means higher successful transmission rate. Moreover, each file segment is further encrypted and sliced by intermediate child nodes recursively. The file segment tree created on the fly based on the hashed-key chain algorithm. Such a design improves the anonymity of the file's origin.

In CC-WSN, each container corresponds to a node in JigDFS. A table of containers with their status is stored in the JigDFS structure and shared by all nodes. When an intrusion or an abnormal event is detected, the reporting node will not only report this event to nearby nodes, but also try to update the Container Status Table (CST) to indicate an change in its status.

C. Juxtapose (JXTA) P2P Framework and Distributed Hash Table (DHT)

JXTA is the most mature open source peer-to-peer protocol framework started by Sun Microsystems. JXTA allows a wide range of devices, from regular computers and cell phones to wireless handhelds, chained together to communicate and collaborate in a P2P manner [11] [12]. JXTA provides a set of communication protocols and services, which construct the underlying overlay network and infrastructure for both CC-WSN and JigDFS. The services provided by JXTA and utilized by JigDFS include:

1) discovering other peers and resources, provided by *Advertisement and Discovery Service*
2) exchanging linked peer lists with other peers, provided by the *Rendezvous Service*
3) communications between peers, provided by the *Pipes and Endpoints*

The Distributed Hash Table (DHT) algorithm is employed in JXTA [13] to index resources and peers because of its provable properties and excellent performance in a P2P network. Basically, a DHT performs the functions of a hash table (i.e. key-value pair table), but in a distributed manner. Just like a traditional hash table, one can lookup a value based on its key, however, the storage and lookups are distributed across the overlay network. The JXTA uses a loosely-consistent DHT walker mechanism to resolve inconsistency of the DHT within the dynamic P2P network, which avoids the use of a stable server or super-peer to ignite the communication.

III. Container Community Wireless Sensor Network

In CC-WSN, each container corresponds to a node in the WSN. Every node cooperatively works with others to tackle intrusion incidents. If an intrusion event is only logged locally on the compromised container, an attacker could cover his/her tracks by modifying the log or wipe the memory on the computing element inside the compromised container. Therefore, in CC-WSN, the Neighborhood Watch concept is adopted to form a neighborhood of containers where each container behaves as a responsible citizen of the community and stays alert to any unusual activity. An unusual activity such as intrusion into a container is reported by a node to its neighbors. The overall architecture of CC-WSN is shown in Figure 1.

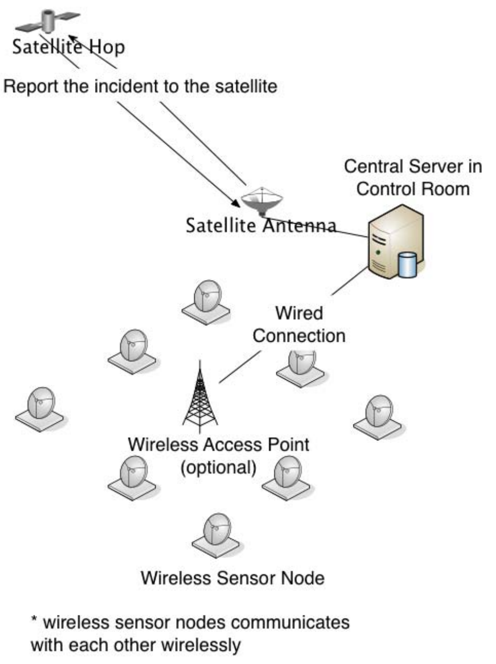
Satellite Hop
Report the incident to the satellite
Central Server in Control Room
Satellite Antenna
Wired Connection
Wireless Access Point (optional)
Wireless Sensor Node
* wireless sensor nodes communicates with each other wirelessly

Fig. 1. Overall Architecture of CC-WSN

In the CC-WSN implementation, a centralized server is located in the ship's control room with sufficient computing power. This server has the capability to gather real time information from the sensor nodes and display the status of containers to the ship's staff. The staff can then report any intrusion or abnormal activity to the port through a satellite connection, if equipped, so that the intrusions can be handled properly. The Wireless Access Point (WAP) in the system is optional. Existence of one or more WAP(s) helps to ease the burden of communication among nodes, but it also speeds up the communication channels between the sensor nodes and the centralized server, since the control room is wired with the WAP. Considering the scenario without a WAP, when a node reports an intrusion event, the message needs to be relayed by the middle nodes and then be transmitted to server. Even with a WAP, the intrusion alert should still be spread across the network.

A. Intrusion Reporting in CC-WSN

When an intrusion is detected, the intrusion event message will be split into n slices using an IDA and delivered to n different nearest neighbors along with the Message Identifier (MID) and Node Identifier (NID) information. The MID is the hash value (SHA-256) of the entire message, which includes Event Type, Event Timestamp, Event Node Identifier and Event Description. Since the combination of NID and timestamp should be unique across the entire network, the MID is unique. MID is used to identify whether a segment is a part of a specific event message, but it is also used to verify the integrity of the reconstructed message. The restored message is genuine if the hash value of the restored message equals the MID, otherwise, the message is corrupted. Moreover, like in JigDFS, we use GISP [14], an implementation of DHT in JXTA, to index message segments generated by the IDA. The following five basic elements captured in the DHT:

1) $[MID]$: Message Identifier is the hash value of the original message using SHA-256.
2) $[t]$: Timestamp of when the event took place.
3) $[SID]$: Segment Identifier is the hash value of the data segment using SHA-256.
4) $[NID]$: The identifier of reporting node, which should be unique across the whole community.
5) $[MSG\_SEG]$: The actual message segment.

Unlike in the traditional JigDFS implementation, each message segment is much smaller in length than a file slice. Therefore, in the CC-WSN implementation, the actual data is stored in the DHT rather than a link that points to the node where the particular file slice would be placed in the case of traditional JigDFS implementation. The DHT is not synchronously shared among all nodes, so that two or more nodes can exchange their local DHTs to share/synchronize information about the resources. As the actual data segment is stored in the DHT, the data segments are passively mirrored. By doing so, this implementation increases the difficulty for an attacker to wipe all the records in the system, since the message segments may have been duplicated and residing on different nodes.

During investigation, an investigator can easily reconstruct the original messages using the corresponding IDA decoder. Because of the characteristic of IDA, the original message can be recovered as long as there are k (i.e. based on the setting of the IDA) uncorrupted slices left.

Figure 2 shows how a message (m) in CC-WSN is split and delivered to its nearby nodes and later reconstructed at the investigator's computer at the destination port. Let us assume that the IDA parameters are set to be $n = 10$ and $k = 3$. These parameters mean that the IDA will split a message m into 10 slices and any 3 out of the 10 slices are needed to recover the original message m. The IDA first adds the extra erasure-code e and produces 10 unique message segments $d_1 ... d_{10}$; then node C_1 finds 10 nearest neighbors according to its peer list (contains a list of peers C_1 knows) and delivers a unique piece d_i ($1 \leq i \leq 10$) along with a timestamp t and the message unique id ($MID = SHA256(m)$) to each of them. The nodes which received a message segment will then record the segment (MSG_SEG), SID, MID, NID and T in a local database as well as in its local DHT. During investigation, the investigator will issue a message data request command REQ, NID (i.e. here NID is the node identifier of where the command is issued) from the server in the control room or any computer linked into the network. The request command will then be propagated across the whole network. Upon receiving the request command, the container node will

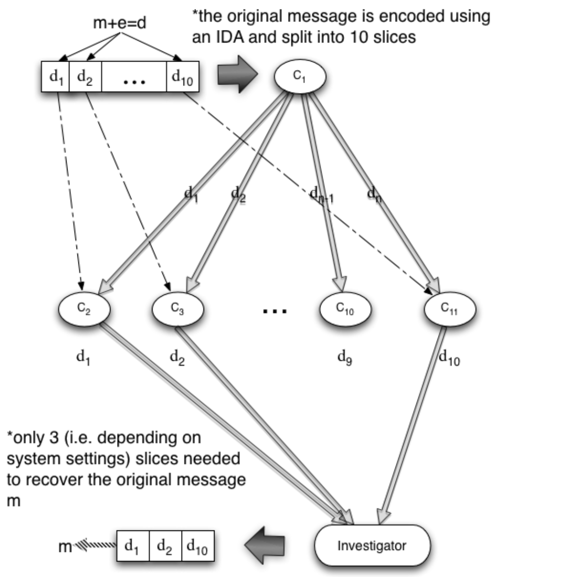
m+e=d
*the original message is encoded using an IDA and split into 10 slices
d1
d2
...
d10
C1
d1
d2
dn-1
dn
C2
C3
...
C10
C11
d1
d2
d9
d10
*only 3 (i.e. depending on system settings) slices needed to recover the original message m
Investigator
m
d1
d2
d10

Fig. 2. Split and reconstruct a message in the CC-WSN using an Information Dispersal Algorithm

respond with a message segment package as follows:

$$\{MID, t, SID, NID, MSG\_SEG\}$$

The message segment package will be transmitted through WAP, if exists, otherwise, the request will be passed by the middle nodes according to the routing algorithm implemented in the network (in our CC-WSN, routing depends on the implementation of JXTA framework). Eventually, the message segment package will be delivered to the requester. The message segments are grouped based on their MID. The original messages can then be reconstructed from any 3 segments in each group.

B. Active Status Polling (ASP)

It is very likely that the attacker may have the knowledge and equipment to block out one or more nodes' communication with others. If such an attack is sucessful, a node being compromised will not be able to report the intrusion incident even though it was detected. In CC-WSN, we introduce the Active Status Polling mechanism to detect such attacks. Each container node in CC-WSN will periodically and actively poll nearby nodes for their status. If there is no response or an invalid response is received from a polled node, the polling node will file an abnormal activity event for the polled node. Therefore, incidents in which the communications are being blocked can be quickly identified and reported to the investigator. The response messages are very simple and short, such as "ALIVE", "COMPROMISED", etc., along with the node identifier and a timestamp.

However, a man-in-the-middle attack can still be performed, if the authenticity of the response messages is not ensured. Therefore, in CC-WSN, all responses to ASP polling requests have to be signed and encrypted. Before starting the trip at the departure port, a public/private key pair is generated and deployed at every container. The private key is known to that specific container and authorized personnel such as an investigator, while the public key is public to all other nodes.

Each message is signed with the sender's private key to ensure the authenticity and integrity of the message. For example, let us assume that node C_j polled the status of node C_i and C_i is going to respond with message m. The private key of C_i is x_i while the public key is g^{x_i}. Also, the signing function is denoted as $sign()$, the encryption function is $enc()$, and $h()$ represents a one-way hash function. C_i first computes the hash value of message m as $h(m)$ and signs the message using its private key x_i and the signing function, $sign()$. The computed signature is

$$S_m = sign_{x_i}(h(m))$$

then, C_i sends the entire response package to C_j as:

$$\{m, S_m, i\}$$

where i represents the node identifier. Moreover, note that the message m itself also includes the NID and a timestamp.

When C_j receives the response from C_i, it can verify the message signature S_m since the C_i's public key g^{x_i} is known to the public. C_j will report an abnormal activity for node i if the signature is not valid or the message itself states that node i is in an abnormal state. The explained approach voids man-in-the-middle attacks, since the private key is unknown to the attacker.

C. Encryption on ASP Responses

Although the ASP response messages are authenticated by digital signatures, the messages themselves are sent in plain-text. Depending on the sensitivity of the contents in the containers, encryption may be used to further harden the CC-WSN against attacks. As mentioned previously, normally PKI infrastructure is considered computationally too expensive to be used in resource limited environments such wireless sensor networks. However, Watro et. al. have developed a lightweight PKI system for WSNs named TinyPK [7], which can be utilized in the CC-WSN. Moreover, depending on the level of security desired and considering the fact that large batteries can be placed in cargo containers, a device with more computational resources (if suitable, even a fully configured laptop) can be used in each container. If the sensor nodes are computationally powerful enough, then one can use a conventional PKI system to create a secure channel between nodes. If so, node C_i will encrypt the ASP request with node C_j's public key; and upon receiving the message, C_j should be able to decrypt the message without any difficulty. Then C_j will encrypt the response message with C_i's public key and C_i can decrypt it with its private key. This way, all communications between two peers can be secure and reliable. Even when the system gets attacked, there is no way for attackers to decrypt the communication among nodes without knowing the proper keys.

To be consistent with our previous notation, let us define the private key of node C_i as x_i and its public key as g^{x_i}; a PKI encryption algorithm $PK\_ENC_{g^{x_i}}(m)$ that transforms a plain-text message into its cipher-text M using private key is g^{x_i}; and there exists a corresponding decryption function $PK\_DEC_{x_i}(M)$ that can decrypt the secret M back to m. Assume that node C_j has been queried by node C_i, then it checks its own state and responds to C_i with a "ALIVE" message along with a timestamp t and the NID of C_j denoted as m. Therefore,

$$m = "ALIVE + t + NID"$$

then the queried node C_j computes the digital signature of m as follows:

$$S_m = sign_{x_j}(h(m))$$

Moreover, node C_j encrypts m with C_i's public key g^{x_i}:

$$M = PK\_ENC_{g^{x_i}}(m)$$

Finally, C_j sends the whole package $P_{h(m)}$ including the unique identifier of the message $h(m)$, the encrypted message M, the digital signature S_m, a timestamp t and node C_j's NID NID_j:

$$P_{h(m)} = \{h(m), M, S_m, t, NID_j\}$$

Upon receiving the message package $P_{h(m)}$ from node C_j, C_i first decrypts the message using its own private key as:

$$m' = PK\_DEC_{x_i}(M)$$

and then computes $h(m')$ and compares it with $h(m)$ to ensure the integrity of the message. If the message is authentic, node C_i then verifies the message's digital signature using node C_j's public key g^{x_j}. Finally, node C_i reads the plain-text response, and acts accordingly.

D. Container Status Table (CST) in JigDFS

Through Active Status Polling, a node can quickly learn the status of nearby nodes and detect whether they have been compromised or not. There is another mechanism used in CC-WSN to further protect the event logs generated by each container. The system keeps a CST, as has been previously mentioned, that contains a list of containers along with their status. Moreover, the database file is stored in the secure distributed file system JigDFS. JigDFS is designed to provide strong encryption and a certain level of plausible deniablity. Files in JigDFS are sliced into small segments using an IDA and distributed onto different nodes recursively to increase fault tolerance against node failures. Moreover, layered encryption is applied to each file with keys produced by a hashed-key chain algorithm, so that the data and keys reside on different nodes.

The plausible deniability of JigDFS may not seem to add value directly to our CC-WSN. However, the design of the deniability in JigDFS makes it hard to trace where a file was uploaded into the system and from which node it was retrieved. Moreover, the file segments can be moved around from node to node. Combining these features further lowers the attacker's chance to cover all nodes that contain segments of a specific file and wipe out the file entirely from the system.

Each container node is considered to be a node in JigDFS. The CST can be retrieved and updated on/by any node in the community. Therefore, when an intrusion is detected, not only will the compromised node report the intrusion event as usual, but it will also update the CST. The same process applies to the Active Status Polling; the polling node that detects an abnormal activity in the polled node will not only report the event but also update the CST. The status of one particular container node might be updated concurrently by two or more nodes with different status. Even worse, an attacker can use a compromised container to falsely update all other containers to normal status. Therefore, the CST not only contains the latest status of each container, but also keeps a history of previously reported statuses along with other information such as the NID of the node where the update is made, a timestamp, etc. An investigator can then easily analyze the CST to not only figure out which container is compromised, but also make a good guess of which container is falsely updating the CST, and therefore, has a good chance of being a compromised container.

IV. Conclusions

In this paper, we proposed to deploy sensors and computing elements into cargo containers to monitor and ensure the integrity of containers. Moreover, all container nodes are wirelessly linked to each other and work cooperatively to tackle attacks. When an intrusion is detected, the compromised container will report the event to its nearby neighbors. The warning message is split into n slices using an IDA and each message segment is sent to a different nearby node. The use of IDA not only increases information redundancy, but it also makes it more difficult for an attacker to cover all nodes to hide his/her tracks. The ASP mechanism is introduced to prevent the situation where the communications of the compromised container is blocked by the attacker. Digital signatures are used to ensure the authenticity and integrity of the responses to the status polling to prevent man-in-the-middle attacks. Furthermore, JigDFS, a secure distributed file system is used to store the table of containers along with their status. Not only the latest status of each container is stored in the CST, but also the history of previously reported status updates along with by whom and when the event was reported. Logging this way reduces the risk that a compromised container can be used by an attacker to falsely update the CST.

Acknowledgments

This work is based, in part, upon research supported by the National Science Foundation (under Grant Nos. CNS-0619069, EPS-0701890 and OISE 0729792). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

[1] J. Bian, R. Seker, and S. Ramaswamy, "Jigdfs in container communities for international cargo security," in *The fifth Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW-09)*. ACM, 2009.

[2] J. Bian and R. Seker, "Jigdfs: A secure distributed file system," in *Proceedings of 2009 IEEE Symposium on Computational Intelligence in Cyber Security*. IEEE, 2009.

[3] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, 2002.

[5] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1448–1453, 2002.

[6] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2004, pp. 129–143.

[7] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: securing sensor networks with public key technology," in *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2004, pp. 59–64.

[8] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[9] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[10] J. Blmer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman, "An xor-based erasure-resilient coding scheme," International Computer Science Institute, Tech. Rep., August 1995.

[11] S. Oaks and L. Gong, *Jxta in a Nutshell*. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2002.

[12] D. Brookshier, D. Govoni, N. Krishnan, and J. C. Soto, *JXTA: Java P2P Programming*. Indianapolis, IN, USA: Sams, 2002.

[13] B. Traversat, M. Abdelaziz, and E. Pouyoul, "Project jxta: A loosely-consistent dht rendezvous walker," jxta-dht.pdf, 2007. [Online]. Available: http://www.jxta.org/docs/

[14] D. Kato, "Gisp: Global information sharing protocol " a distributed index for peer-to-peer systems," in *P2P '02: Proceedings of the Second International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2002, p. 65.