A Secure Agent Architecture for Sensor Networks

Zhaoyu Liu, Yongge Wang
{zhliu, yonwang}@uncc.edu
Department of Software and Information Systems
University of North Carolina at Charlotte
9201 University City Boulevard
Charlotte, NC 28223

Abstract

Advances in sensor and wireless technologies will enable thousands, or even millions of small but smart sensors to be deployed for a wide range of monitoring applications. The sensor networks must be dynamically configurable and adaptive in order to response actively to events. Security must also be built into sensor networks at the very initial design stage to prevent any potential threats. This paper presents a design of a generic software architecture based on tiny active agents for sensor networks. The architecture has built-in security components. This architecture will significantly benefit the dynamic and secure deployment and configuration tasks of large sensor networks for various applications.

Keywords: adaptivity, security, tiny active agent, digital authenticator, sensor networks,

1 Introduction

Advances in sensor and wireless technologies will enable thousands, or even millions of small but smart sensors to be deployed for a wide range of monitoring applications [12]. Typical applications of sensor networks include, but are not limited to, natural event monitoring, emergency response information, energy management, medical monitoring, logistics and inventory management, and battlefield management. The sensor networks must be dynamically configurable and adaptive in order to response actively to events. Security must also be built into sensor networks at the very initial design stage to prevent any potential threats.

Flexibility and adaptivity are very important for sensor networks in practice. For example, in a natural event monitoring application, most sensor nodes are in hibernation state under normal situation in order to save energy. But under certain events, the sensors need to switch to an active state. Active responses are also needed during attacks or faulty cases, in order to contain the damages [7]. Agents have been applied extensively to wired infrastructure to achieve flexibility and adaptivity [15, 7, 1]. It is challenging to apply active agent technology to sensor networks, mainly due to the sensor node's resources constraints, such as limited processing power, storage (RAM, ROM, and etc.), bandwidth, and energy.

Security system must also be built into sensor networks to achieve secure communications. Secure communications have been extensively studied by researchers in the Internet scenario [3, 14]. Though these results could not be applied to sensor networks directly, some general approaches that have been exploited previously could be adapted to sensor networks. The major difference between sensor networks and other networks is that, by design, sensors are inexpensive, low-power devices. Only if cheap sensors could be built, a large deployment of sensor networks is possible.

In this paper, we present a generic software architecture based on tiny active agents for adaptive and secure sensor networks. Security mechanisms are built-in components of this software architecture. Survivable sensor networks can be achieved by using the software architecture. Our software architecture will facilitate the following sensor node functions or features (see, e.g., [2]) that are of critical importance to the success of sensor networks:

- to support uniformly a variety of network functions on different nodes (e.g., base stations, gateway, ordinary sensor node) through dynamic reconfiguration;
- to require only a minimal pre-configuration prior to deployment, and to add new processing functionalities dynamically and remotely;
- to support energy-efficient networking to exchange data locally and remotely;

- to support application-specific security services; and
- to provide an essential base of countermeasurements.

This paper is organized as follows. Section 2 presents some background information. Section 3 describes the design of our agent-based architecture in detail. Section 4 concludes the paper.

2 Background

A typical sensor network consists of nodes, which are small battery powered devices, and base stations. Sensor nodes communicate wirelessly with more powerful base stations, which in turn may be connected to an outside wired network. The numbers of base stations and sensors vary according to the purpose of sensor networks. For example, in a national natural event monitoring systems, there could be hundreds of base stations and thousands of or millions of sensors. On the other hand, for a small system such as the SmartDust sensor network [12], there is one or a few base stations and less than one hundred sensor nodes. Generally sensor nodes communicate over a wireless network and broadcast is the fundamental communication primitive. In addition to base stations and normal sensor nodes, some sensor networks could contain special sensor nodes with relatively stronger capabilities, such as having replaceable batteries. These special sensor nodes could be used for special purposes such as routing or data aggregation and forwarding. In particular, similar to normal sensor nodes, these special sensor nodes could also be built significantly cheap (compared to base stations), thus be deployed in a relatively large scale.

In this paper, we consider generic sensor networks in which nodes are geographically distributed in a large scale of unattended area. Base stations may not be able to communicate with sensor nodes directly and sensor nodes may also not be able to communicate with base station directly.

3 Software Architecture

This section describes the major components of our architecture and describes the security model that forms the basis of interaction among these components. Our architecture is based on executable *tiny active agents*. A tiny active agent carries concise dynamical configuration information, customized or tailored for a particular application or device. The information include security policies and mechanisms. The other components in our architecture include agent control and management, the software framework, the evaluation/enforcement engines, and the symmetric cipher based key distribution and management system. Agent control and management consists of a distributed network of agent administrators, software framework component repositories and agent servers. These management entities are trusted and reside at base stations. The agent administrator is responsible for verifying, validating and certifying the code inside the agent, and for authenticating the agent, based on the key distribution and management system.

The agent administrator can additionally manage the distribution of the agents using a secure channel. Trusted applications or sensors may be allowed to create their own agents. Each protection domain typically has one or more agent administrators that are collectively responsible for the integrity of the agents.

The agent uses a software framework for context. For example, a software framework may include a hierarchical structure of object-oriented classes of necessary security policies. Typically this framework is componentized and arranged so that the components themselves can be downloaded from the software component framework repository using a secure channel. Each node, typically a sensor device, acts as an evaluation/enforcement engine in its address space. This engine can also be customized according to the context and its components can be preconfigured in advance or be dynamically downloaded by using the secure channel. The agents are evaluated in the execution environment provided by this engine by instantiating the context of the software framework. The engine also enforces the result of the evaluation. We give a detailed description of these components next.

3.1 Agent Format

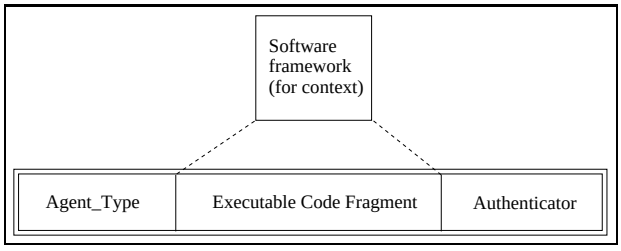

Figure 1: Generic Agent Content

The format of a generic agent is shown in Figure 1. The first field is similar to a header and contains information about the *type* of the agent. For instance, the agent type `ACCESS_CONTROL` indicates that the agent encodes an access control policy. The second field is the most important part of the agent. Ideally it can contain

an arbitrary piece of code, which specifies the policies, mechanisms, and/or other dynamic configuration information for a particular application. We use a software framework that can provide a context for the code field of the agent. The agent code can use the components of this software framework and impose additional constraints on their usage, leveraging the expressive power of the underlying framework. We describe one particular instance of this framework later. The framework implements an application platform and an access control policy in a modular and composable fashion. In addition, the agent can use the features provided by the underlying language and add conditional processing, based on timestamps, or accumulated credits, to specify timeouts and to impose limits on resource utilization, etc. The flexibility afforded by this approach is limited only by the language syntax. For efficient transmission purpose, this field needs to be optimized and to be as small as possible.

The last field is the digital authenticator. Typically the agent is created by an agent administrator or a trusted entity. This entity is responsible for the integrity of the agent, and attests to this by authenticating the agent. In our distributed architecture, each protection domain has one or a small number of replicated agent administrators. A public key infrastructure is not suitable for sensor networks. We propose a simple and efficient key distribution and management system that is based on symmetric cipher. Our architecture uses the system for the authenticator generation and verification.

The agent provides an interface that exports at least the following methods:

- `evaluate`
- `bind`

The `evaluate` method is called by the evaluation/enforcement engine. This method causes the executable code in the agent to be evaluated, and controls the configuration enforcement requested by the application. The `bind` method is used to bind agents to applications and aids in the retrieval of the context in the evaluation/enforcement engine. This list is not exhaustive and additional methods can be added to extend the functionality and create new agent types.

3.2 Software Framework

This section describes a particular example of a software framework that can be used to provide a context for the agents [4]. The framework is implemented in a composable and extensible object-oriented way. This framework has a GUI front-end that simplifies the process of specifying the policies and configuration parameters. The GUI functionalities reside at base station of sensor networks. This allows users to specify configurations tailored to their specific operational needs.

The software framework itself is a hierarchy of classes as shown in Figure 2. It is dynamically configurable and extensible. The classes at the bottom of the framework are mostly abstract, and are mainly used to represent mathematical concepts such as sets and mappings. These classes form the basis for a hierarchy of successively specialized classes representing concepts such as labels and access control lists. Finally, at the top of the framework are classes used to represent a policy form. This framework provides a platform for agent evaluation and security policy specifications. We use Role-Based Access Control (RBAC) policy [13] in our architecture for resource access control.

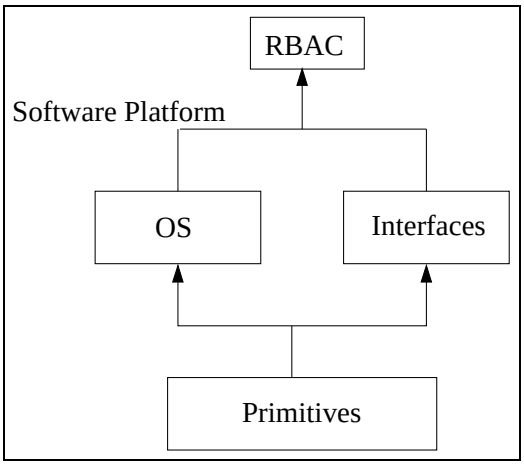

Figure 2: Component-level Map of the Software Framework

3.3 Evaluation/Enforcement Engine

This section describes the evaluation/enforcement engine component of our architecture. Figure 3 shows the representation of this component. The evaluation/enforcement engine consists of an agent cache, run-time resolvable references to customizable agent evaluation environment, and run-time resolvable references to a customizable, componentized software framework. The agent cache is used to cache agents that do not change very often and provides a fast processing path for commonly used or default mechanisms and policies. Different agent types require different contexts. By providing the run-time resolvable references, we can download the additional software components that form the context from a trusted repository, on the top of pre-configured ones. The pre-downloaded and pre-configured components would be sufficient for regular operations. An execution environment imposes

static and dynamic constraints on the code that runs inside it.

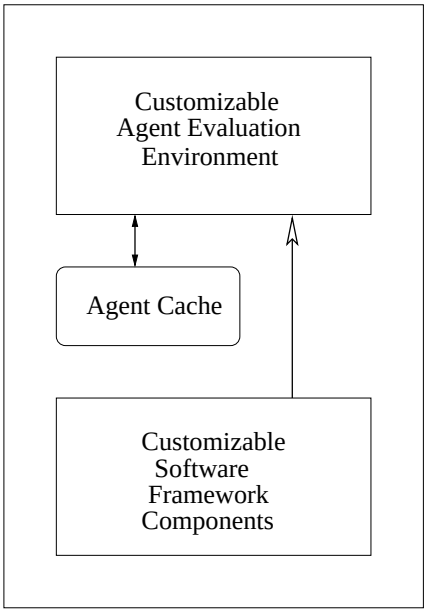

Figure 3: Evaluation and Enforcement Engine

The execution environment can also customize the evaluation of the agents or trusted applications. The established secure channel between a sensor node and base station can be used to obtain the agents and the downloadable software framework and execution environment components. The enforcement is done after evaluation. The evaluation/enforcement engine can subsume the concept of a traditional reference monitor. Typically when the application makes a call that accesses a specific resource or requires the use of a specific mechanism, the request is encapsulated and passed to the evaluation engine. The engine builds the context and evaluates the agent associated with the configurations requested by the application. Depending on the result of this evaluation, the engine can either grant or deny the application configuration request. The engine must either notify applications that the configuration request is denied or allocate the appropriate resources, thereby implementing and enforcing the configurations specified in the agent.

Since the evaluation/enforcement engine reside on sensor nodes, the efficiency would be the major challenging research issue. Optimization techniques need to be investigated with the consideration of resources constraints of sensor nodes.

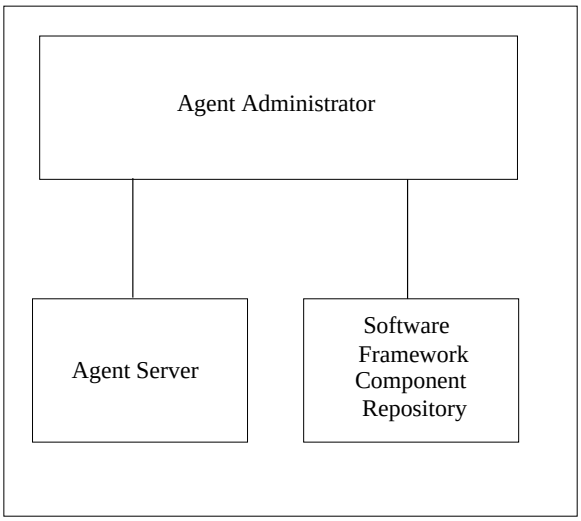

Figure 4: Agent Control and Management

3.4 Agent Control and Management

The agent control and management are handled by base stations. The agent administrator is equivalent to a trusted third party as in a traditional security model. It is responsible for validating and attesting to the integrity of the agents. For example, using our proposed key system described later, the agent administrator can authenticate agents, and other entities, like applications and evaluation/enforcement engines, can perform authenticator verification. Typically the administrator is also responsible for the creation of the agents using the interface provided by the software framework. Although agents can carry arbitrary code, the creation interface provided by the framework can restrict the agents to well-formed optimized expressions and can perform static type checking and verification to make sure that the code in the agent cannot compromise the security of the underlying system. In addition, the communication channel between the administrator and the evaluation/enforcement engine needs to be secure. The agent administrator itself may be implemented as one centralized entity in a base station, or its functionality can be distributed throughout several base stations in one protection domain. There may be multiple instances of the agent administrator to achieve load balancing, scalability and fault tolerance. The agent server acts as a front-end to an agent administrator's agent repository. This server may be a part of the administrator or may be another entity, closely coupled with the functionality of the agent administrator.

3.5 Security Services

The aims of security component in the software architecture are:

- confidentiality: only authorized parties have access to the content of the messages;
- integrity: unauthorized parties should not be able to modify the messages in transit;
- authentication: only authorized parties could initiate mobile agents for a specific task or command;
- access control: a flexible role based access control mechanisms to control the participation in the sensor network management and communications; and
- ease of use: most components are transparent and the systems should not be difficult to use.

In order to build a security-aware software architecture for sensor networks, it is necessary:

- to understand the essential security threats in a distributed sensor network;
- to design enabling security technologies for extremely resource constrained devices (e.g., sensor nodes);
- to integrate these enabling security technologies into the design of software architecture so that they are the essential components of the architecture; and
- to design upper layer security mechanisms so that these mechanisms could be provided as application services above the software architecture.

The primary challenge in developing security mechanisms for sensor networks is the extremely constrained capability of sensor nodes. Public key cryptography is too expensive for sensor networks since it generally requires extensive computation and high energy consumption at sensor nodes. Even symmetric key encryption and authentication mechanisms for conventional networks are not suitable for sensor networks since they may require large overhead per packet. Larger message overhead per packet is generally not acceptable in sensor networks. For example, in a typical Supervisory Control & Data Acquisition System (SCADA) network for power grids which uses "poll-response" model, poll messages should typically be small (e.g., 16 bytes) and responses should be ready in a very short time period. This kind of requirement excludes the possibility of using computation intensive symmetric ciphers. The security components of our software architecture should be flexible and their design should take all kinds of requirements into consideration. In particular, efficient stream ciphers (such as RC4) and hash function-based one time authentication scheme (such as [6]) will be used to design data authentication and broadcast authentication schemes. Indeed, Lamport's one-way hash function based authentication scheme [6] has been used in design for network broadcast schemes (e.g., TESLA [9, 10] and μTESLA [11]). In addition, key management mechanisms will be essential in the security components of the architecture.

In our model, we assume that each sensor has a built-in master key stored in a tamper-resistant component. The master key of each sensor can initially be transferred to the base station manually when sensor nodes are deployed. Our software architecture will support Kerberos-like key distribution schemes, where base stations can serve as Kerberos-like authentication servers. The term of "Kerberos-like" means that key distribution scheme for sensor networks is not the Kerberos scheme. Indeed, it is impossible to deploy the Kerberos scheme on sensor networks due to the constrained capabilities (note that Kerberos is based on Needham-Schroeder protocol [8]). Energy and computation efficient provable secure key distribution and authentication schemes for sensor networks will be designed and embedded into the software architecture.

The key establishment protocols and approaches for distributed sensor networks must satisfy several security and functional requirements. The key protocol must establish a shared key (or several shared keys) that can be used by two or more sensor nodes to provide confidentiality and group-level authentication of application data. The protocol must establish a key between all sensor nodes that need exchange application data securely. A single key may protect data over a large portion of the network, or just a pair of nodes.

Each agent carries a digital authenticator so that the host (either a sensor node or a base station) can check the authenticity (including integrity) of the agent. In our scheme, symmetric key based authentication schemes will be used to generate authenticators for agents. For example, when a base station wants to send an agent to a specific sensor node, the symmetric key shared between the base station and the sensor node or a random session key (protected by the long term shared key) can be used to generate the authenticator. When a sensor node wants to send an agent to another sensor node, a session key needs to be agreed first, either by the key distribution scheme or by the key agreement scheme discussed later. Then the sensor node can use this agreed session key to generate the authenticator. If a base station wants to broadcast an agent or to send an agent to carry out a task over several sensor nodes, then efficient one-way hash

function based authentication schemes will be used to generate a single authenticator (similar authentication schemes has been used in design for network broadcast schemes and sensor broadcast schemes, such as TESLA [9, 10] and μTESLA [11]).

Although each sensor node has a built-in master key that is shared between the sensor node and the base station, it is impractical for each pair of sensor nodes to share a secret in a large sensor network that could have millions of sensor nodes. Thus in order for two sensors to communicate securely, they have to contact base stations first to get a session key for them to communicate. In certain environments or applications, this may be difficult to achieve or may not be necessary (note that in our sensor network model, sensors may not be able to reach base station directly and base stations may not be able to reach a specific sensor node directly).

We will study the fundamental technologies for two sensor nodes to share a secret without contacting a trusted third party and without using public key technologies that is too expensive for sensor networks to implement. Our technologies will be based on our previous research results [3], which are for conventional networks without the consideration of constrained device capabilities.

3.6 Application Examples

Secure routing is critical for sensor networks [5]. Our software architecture allows tiny active agents to route themselves among sensor nodes towards a node of interest, to configure dynamically the routing algorithms in use, or to change the routing information contained in host sensor nodes. All these operations are performed with proper security configurations. Other critical services that will be built upon our software architecture include active responses. When some intrusions are detected, authenticated mobile agents could be initiated to dynamically change the configurations (including security parameters) of the sensor network and to repair the damages that have been caught by the intrusions.

4 Conclusion

Achieving adaptive and secure communications in sensor networks is becoming one of the most important problems in designing sensor networks. Active agents and generic software architectures based on them have been extensively studied and designed for wired computing systems and networks. Although these techniques have been developed to demonstrate the advantages of the active agents and software architectures, it is challenging to apply these techniques to sensor devices beneficially due to the inherent resources constraints of such devices. This paper presents a design of a generic software architecture based on tiny active agents for sensor networks and identify the key research issues for optimizing the architecture for the constrained environment in sensor networks. Security mechanisms are built-in components of the software architecture. Using the software architecture we could develop survivable sensor networks with different security features.

References

[1] Roy H. Campbell, Zhaoyu Liu, M. Dennis Mickunas, Prasad Naldurg, and Seung Yi. Seraphim: dynamic interoperable security architecture for active networks. In *IEEE OPENARCH 2000*, Tel-Aviv, Israel, March 26–27, 2000.

[2] D. Carman, P. Kruus, and B. Matt. Constraints and Approaches for Distributed Sensor Network Security (final), DARPA Project report, (Cryptographic Technologies Group, Trusted Information System, NAI Labs), September 1, 2000

[3] Y. Desmedt and Y. Wang. Perfectly secure message transmission revisited. In: *Proc. Eurocrypt '02*, pages 502–517, Lecture Notes in Computer Science 2332, Springer-Verlag, 2002.

[4] Tim Fraser. An object-oriented framework for security policy representation. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, December 1996.

[5] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *Proceedings of the first International Workshop on Sensor Network Protocols and Applications*, November, 2003.

[6] L. Lamport. Password authentication with insecure communication. *Communication of the ACM*, **24**(11):770–772, 1981.

[7] Zhaoyu Liu, Roy H. Campbell, and M. Dennis Mickunas. Active security support for active networks. *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Technologies that promote computational intelligence, openness and programmability in networks and Internet services*. Accepted.

[8] R. Needham, and M. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, **21**(12):993–999, December 1978.

[9] A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium (NDSS)*, 2001.

[10] A. Perrig, R. Canetti, J. Tygar, and D. Song Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, 2000.

[11] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler. SPINS: Security protocols for sensor networks. *Wireless Networks* **8**:521–534, 2002.

[12] K. Pister, J. Kahn, and B. Boser. SmartDust: wireless networks of millimeter-scale sensor nodes. *Highlight Article in 1999 Electronics Research Laboratory Research Summary*, 1999.

[13] R. S. Sandhu and E. J. Coyne. Role-based access control models. *IEEE Computer*, 29(2), February 1996.

[14] Y. Wang, Y. Desmedt, and M. Burmester. Models for dependable computation with multiple inputs and some hardness results. *Fundamenta Informaticae*, 42(1):61–73, 2000.

[15] D. Wetherall, J. Guttag, and D. Tennenhouse. ANTS: a toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH'98*, San Francisco, CA, April 1998.