Distributed Device Networks with Security Constraints

Xu, Y.; Song, Ronggong; Korba, Larry; Wang, L.; Shen, W.; Lang, S.

*Distributed Device Networks with Security Constraints**

Xu, Y., Song, R., Korba, L., Wang, L., Shen, W., and Lang, S.
November 2005

* published in IEEE Transactions on Industrial Informatics. Vol. 1, No. 4. November 2005. pp. 217-225. NRC 48222.

Distributed Device Networks With Security Constraints

Yuefei Xu, Ronggong Song, Larry Korba, Lihui Wang, Weiming Shen, *Senior Member, IEEE*, and Sherman Lang

***Abstract*—In today's globalized business world, outsourcing, joint ventures, mobile and cross-border collaborations have led to work environments distributed across multiple organizational and geographical boundaries. The new requirements of portability, configurability and interoperability of distributed device networks put forward new challenges and security risks to the system's design and implementation. There are critical demands on highly secured collaborative control environments and security enhancing mechanisms for distributed device control, configuration, monitoring, and interoperation. This paper addresses the collaborative control issues of distributed device networks under open and dynamic environments. The security challenges of authenticity, integrity, confidentiality, and execution safety are considered as primary design constraints. By adopting policy-based network security technologies and XML processing technologies, two new modules of Secure Device Control Gateway and Security Agent are introduced into regular distributed device control networks to provide security and safety enhancing mechanisms. The core architectures, applied mechanisms, and implementation considerations are presented in detail in this paper.**

***Index Terms*—Distributed device network, distributed device control, industrial control systems, collaborative control, network security, computer supported cooperative work.**

I. Introduction

WITH the growing globalization and decentralization of businesses, the boundary between what is "inside" and what is "outside" of an organization is blurring. Businesses and interactions are now happening across traditional physical boundaries. The decentralization of organizations has become a major impact on the traditional business models. Services and resources are distributed everywhere and sourced anywhere through global supply chains. For example, in the area of e-Manufacturing, product design, process planning, scheduling, and manufacturing have shifted rapidly from simply occurring within one enterprise to being spread across global networks. To cope with this trend, a collaborative environment with interactive design, planning, scheduling, monitoring, and control capabilities is essential for any manufacturing enterprise to increase its competitiveness and profitability.

Recently, there are strong demands in industry to add portability, interoperability, configurability, and other collaborative features to existing industrial control systems. These provide the following advantages.

- Control tasks or components can be designed and exchanged between different vendors.
- Different devices can be operated and monitored by each other or by human operators, within or outside the organization.
- Different devices can be reconfigured remotely to respond unanticipated events.

How to keep all the above activities under control in an open and dynamic environment is still a very challenging question, especially when one considers the distributed control features of low-level control services, machines, devices and processes. For example, even for modern factories with programmable logical controllers (PLCs), their status and processes are kept in closed environments and are separated from outside networks. Meanwhile, different control systems may use different devices from different vendors. Software and tools by different suppliers cannot recognize each other. The status and operations of distributed devices are difficult to predict and control from a remote site. Such situations have created barriers to forming new collaborations with changing global supply chains and other activities.

In recent years, a number of research projects have been proposed to address the collaborative control problem of distributed device systems in open environments. Significant projects include *NIIIP* [1]. The goal of this project was to develop a series of open industry software protocols that can make software interoperation possible between manufacturers and their suppliers. More recently, *Cimplicity* [2] from GE Fanuc Automation (USA) was developed to allow users to view their factory's operational processes through an XML-based *WebView* screen. In order to bring legacy machine tools on-line and make machine tools become servers of information, e-Manufacturing Network Inc. (Canada) introduced its *ION Universal Interface* and *CORTEX Gateway* [3]. Hitachi Seiki (Japan) introduced *Seiki FlexLink Open CNC/PC Network Connectivity* [4] to its turning and machining centers, making possible to connect with a hand held personal digital assistant (PDA) or a laptop computer and do in-process gauging, machine monitoring, and cycle-time analysis. Since 1998, Mazak (Japan) has operated its high-tech *Cyber Factory* concept at its headquarters in Oguchi, Japan. The fully networkable *Mazatrol Fusion controls* allow Mazak machines to communicate over wireless networks for applications including real-time machine tool monitoring and diagnostics. *MetaMorph II* [5] was proposed as a hybrid, agent-based, mediator-centric architecture to integrate partners, suppliers, and customers in a dynamic manufacturing environment.

Manuscript received March 19, 2004; revised September 19, 2004.

Y. Xu, R. Song, and L. Korba are with the Institute for Information Technology, National Research Council of Canada, Ottawa, ON, Canada (e-mail: Yuefei.Xu@nrc.gc.ca; Ronggong.Song@nrc.gc.ca; Larry.Korba@nrc.gc.ca).

L. Wang, W. Shen, and S. Lang are with the Integrated Manufacturing Technologies Institute, National Research Council Canada, London, ON, Canada (e-mail: Lihui.Wang@nrc.gc.ca; Weiming.Shen@nrc.gc.ca; Sherman.Lang@nrc.gc.ca).

Digital Object Identifier 10.1109/TII.2005.843826

However, despite all these accomplishments, most of the above systems are either for off-line simulation or for monitoring only. Most of these systems require a specific application to be installed and configured instead of using standard interfaces. The requirement of a specific application has limited the system portability, interoperability, and configurability. Advanced system design, planning, scheduling, control and execution remains isolated from the collaborative processes. To be more competitive, users are now demanding flexible and adaptable solutions for their requirements.

On the other hand, the Internet is becoming widely used as an open medium for information communication, and is expanding to industrial control and device control areas as well. With system control and processing continuing to move toward open and dynamic environments, like the Internet, existing simple security protection methods, such as user name-password approach, tend to be insufficient to face all the risks. More seriously, the requirements of portability, configurability, interoperability, and other collaborative features bring new security risks and challenges. The corresponding mechanisms and solutions that are fully competitive with the demands of open and dynamic environments are critical for the safety and the functionality of collaborative systems.

This paper explores the design issues of distributed device networks and the corresponding secure collaborative control solutions under open and dynamic environments. The design of distributed device control systems its primary implementation challenges of portability, configurability, interoperability are discussed in Section II. The security constraints and execution safety requirements are then analyzed in Section III. In Section IV, a distributed device control model with security features is proposed to address how distributed devices can be controlled, not only remotely, but also securely. The concept and architecture of Secure Device Control Gateways are introduced in Section V. The design of the Security Agent (SA) and the applied security mechanisms are presented in Section VI. Implementation considerations are discussed in Section VII. The future work and conclusions are given out in Section VIII.

II. Design of Distributed Device Control Systems

Within the efforts toward increasing the interoperability, portability, configurability and other collaborative features of distributed device control systems, International Electrotechnical Commission's Function Block specification (IEC61499) [6] is one of the most significant efforts. IEC61499 provides a set of systemic approaches for the design of distributed industrial process measurement and control systems (IPMCS). It specifies a series of reference models and covers the whole life cycle of a control system, including the phases of system planning, design, implementation, validation, operation, and maintenance.

According to IEC61499, a distributed control system is defined as a collection of interconnected devices communicating with each other by means of one or more communication networks, as shown in Fig. 1. A function performed by the control system is modeled as an application which may reside in a single device, such as the application C in Fig. 1, or may be distributed among several devices, such as the applications A and B in Fig. 1. For instance, an application may consist of one or more control loops in which input sampling is performed in one device, while control processing in another device, and output conversion in a third.

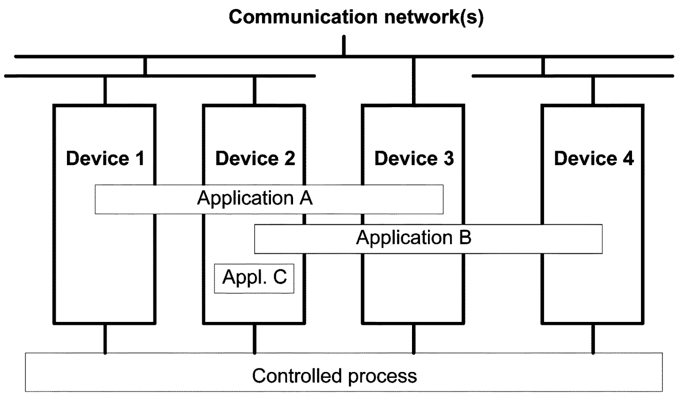

Fig. 1. IEC61499 Distributed Control Systems Model (Source: [6]).

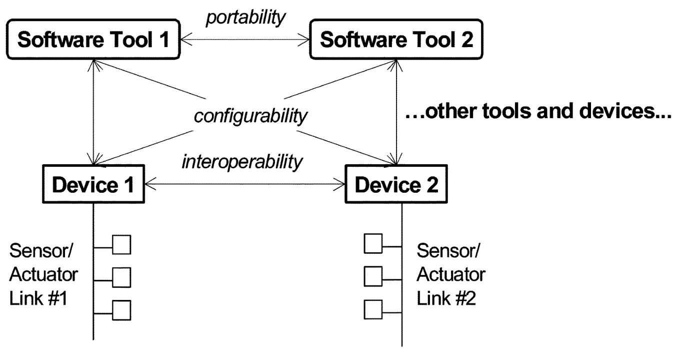

Fig. 2. Implemented features of IEC 61 499-based distributed control systems (Source: [7]).

IEC 61 499-4 specifies 4 primary implemented features in its compliance profile [6]. These features, as illustrated in Fig. 2, are briefly introduced as follows.

- *Portability:* the capability to exchange control components or codes between different tools and suppliers.
- *Configurability:* devices from multiple vendors can be configured by different software tools from different suppliers.
- *Interoperability:* devices from different vendors can cooperate with each other.

While the IEC 61 499 architecture described in the standard focuses on reference models that allow distributed applications to be developed using function blocks (FBs), it provides a well-suited modeling architecture for the design of distributed device control systems. More importantly, IEC61499 considers XML encoding as a standard format for device control commands and responses. IEC61499's specification of IEC/PAS 61 499-2 gives precise XML document type definitions (DTDs) of FB. So FB elements can be expressed in XML and stored as FB libraries. FB codes can also be easily made portable among different commercial software tools and devices only if such tools and devices can correctly parse and interpret all elements of XML elements. As a result, IEC61499 architecture is particularly suitable for collaborative control in an environment that is concurrent, asynchronous, and distributed.

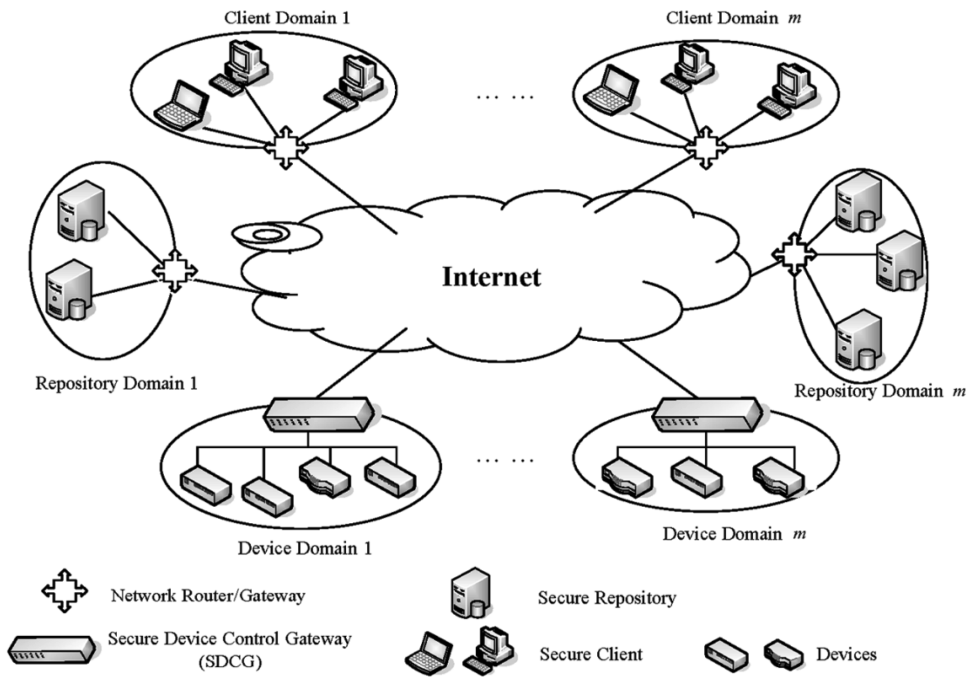

Fig. 3. High-level model of secure collaborative distributed device control systems.

Demonstration of IEC61499 and FB related concepts for distributed industrial and device control systems have been demonstrated in a number of projects in the literature [8]–[10]. For example, Brennan *et al.* [8] proposed a model to support runtime reconfiguration of distributed control systems that is built upon FB models. FBDK [9] was developed as an experimental development kit to enable the building and testing of FB-based control and applications. However, IEC61499 specification does not address how FB components are stored, retrieved and protected in a network environment. In another words, there is no discussion concerning how such a distributed device control system could run under open and dynamic environments. It is also hard to find such discussion in available research literature. In order to actually implement collaborative control under open and dynamic environments, further research and developments are required.

III. Security Challenges Under Open and Dynamic Environments

The open and dynamic environment brings many new challenges to the collaborative control of distributed device systems. In such an environment, for example, device access and information exchange have to cross multiple corporate networks or even over the Internet. Critical security challenges may arise in the processes of command transferring, function modifications, remote diagnosis, and maintenance. Any control operations in open and dynamic environments may result in potentially hazardous conditions. Most significantly, security risks may come from the network, data storage, operating platforms, and application modules. Compared with traditional control systems, generally running in closed, trusted environments, this open collaboration scenario is accompanied by the following significant security challenges.

a) *Identification:* Distributed devices, software tools, and human operators need to identify themselves within the open and dynamic environment. This is the means by which devices, tools, and users may use to claim their identities to the device control system.
b) *Entity Authentication:* This is a process to demonstrate the evidence of the identity of a device or operator.
c) *Data Authentication:* This is a process to demonstrate the evidence of the identity of the original source of exchanged commands, control codes, or device feedback.
d) *Authorization:* This is a process by which an entity is granted to access another entity's processes or resources. It determines the extent of system rights that an entity holds.
e) *Integrity:* Command and control codes must be protected to ensure that no part is changed while in storage, or in transmission across networks.
f) *Confidentiality:* Valuable codes and data, perhaps representing confidential manufacturing processes, must be protected from malicious attackers or competitors.
g) *Nonrepudiation:* An entity (a device, an operator, and so on) cannot deny the authenticity of its signature on a command or a communication that it originates.
h) *Execution Safety:* The acceptance and execution of outside commands and control codes must be safe to the device and the control system. Each device may have independent local task execution and protection requirements.

IV. Design of Distributed Device Control Systems With Security Features

This section presents the design of a secure collaborative distributed device control system. The high-level model of this system is illustrated in Fig. 3.

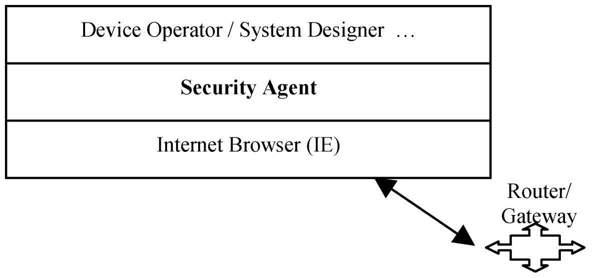

Fig. 4. Functional modules in a SC.

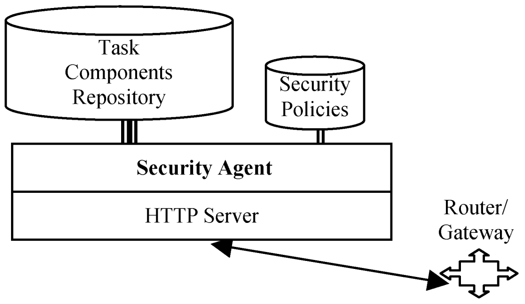

Fig. 5. Functional modules of SR.

There are three logical domains in this model: client domain, repository domain, and device domain. Entities in different domains may be geographically distributed and connected through networks (LAN or WAN). The collaboration and control operations may run across any open and unsecured network, like the Internet.

Each client in this model is a secure client (SC), which is protected by a module called the SA. Fig. 4 illustrates the basic functional modules of SC. The SA is playing important roles, which implements all the security enhancing operations including security policy negotiation, entity authentication, data confidentiality, nonrepudiation, and integrity. The architecture and the working mechanisms of SA are to be presented in Section VI in details.

On the client side, an operator or designer holds one or more credentials, which certify that s/he has some granted rights to request some services under limited conditions. For example, before an operator wants to query the status of a device, the operator must provide a proof (credential) signed by the device's administrator or others who have the delegation authority. Depending on an operator's priorities and responsibilities, s/he may request different services, like programming a new control application, device reconfiguration, device operation, or device monitoring.

On the repository side, a series of control components, expressed as IEC61499 FB Codes are stored in different repositories. These components are different control functions and applications (e.g., PID Control) that may come from different suppliers. These repositories are distributed across several physical locations and may belong to different organizations. There is a SA residing in each of the repositories. The SA is responsible for all the security-related operations such as authentication, data encryption, integrity verification, audit and so on. Each repository SA also maintains a series of security-related policies, which specify the detailed security requirements and constraints. Only requests compatible with these policies will be served by SA. Fig. 5 illustrates the basic functional modules of a secure repository (SR).

On the device side, there are a series of secure device control gateways (SDCGs) with local devices beneath them. By default, each SDCG and its associated devices are considered to be within one trust boundary. This means that there is no security risk between devices in one trust boundary. More details of the SDCG will be discussed in Section V.

In our model, the above three domains are independent from each other. Each client is an operating platform which provides user interactions. A repository is responsible for the storing of

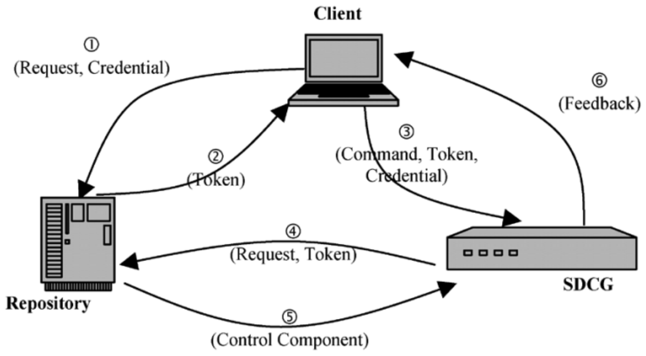

Fig. 6. Typical secure processing among Client, Repository, and SDCG.

control components and codes. A device is responsible for the execution of control tasks. While all the control components could be logically designed or reassembled by software tools at the client side, all the codes are stored in one or more repositories and executable by one or more devices. Therefore, control components and codes are securely stored and retrievable by authorized users or devices from anywhere, and an authorized operator can use any client to control distributed devices remotely.

Here is one example to demonstrate the security working mechanisms of a secure distributed device control process. Suppose an operator wishes to reconfigure some application running on a device, the following activities would occur in progression. First, the operator defines the new control function by browsing available control components and codes, e.g., PID module, from a repository. Then she/he sends commands to that device. The device executes the commands. The operator gets feedback from the device as necessary. In this case, the typical security operations among the involved three parties (client, repository, and SDCG) are illustrated in Fig. 6.

1) First, the SA at the client side sends a message containing its Request, Credential to the repository. The Request indicates that the *Operator* wants to retrieve some available control components from a repository. The credential indicates the operator's authority to do that request.
2) The SA at the repository side checks the credential against the security policy to see if the request is authorized. If the request and provided credentials satisfy the policies, the *Repository* SA sends a *Token* to the *Operator*. This *Token* is an authorization, which means that a holder of the *Token* has the right to retrieve some specified control components and codes, from the repository.

3) After getting the Token, the operator sends the reconfiguration command, the token and her/his reconfiguration credentials to the corresponding SDCG for the device in its realm.
4) The SDCG then checks the request against SDCG's security policy. If the credential cannot satisfy the security policy, the request will be rejected. It is possible that the operator turns to seek more credentials to support his/her request, e.g., by coordinating with the device administrator. If the policy is satisfied, the SDCG asks to retrieve the specified control components from the corresponding repository by sending a Request, Token pair to that repository.
5) The repository verifies the Request and the Token against its security policy. If satisfied, the repository sends the control components and codes to the SDCG.
6) The SDCG then installs the control components on the specified device according to the operator's request. Accordingly, SDCG may send a (*Feedback*) like task-finished message to the operator.

Above is a simplified process of device configuration. There could be more complicated security processes between Client/Repository, Client/SDCG, or SDCG/Repository. For example, besides browsing available control components, an operator may design new components as necessary and store the new designs in a repository. In order to store the new design, the client will require a corresponding "storing" token from the repository and then send the (Request, Control Component, Token) to that repository to complete the storage process.

V. Secure Device Control Gateway

In our proposed model, the SDCG plays important roles by mediating outside requests and internal control actions in the Device Domain. It guarantees the security and safety of the device control. The architecture of this SDCG is illustrated in Fig. 7, which includes the following.

- ***Security Agent:*** For incoming dataflow, the SA checks the authenticity and integrity of the data. Then it decrypts dataflow, which may contain commands or control components in the form of XML. For outgoing dataflow, the SA signs and encrypts outgoing XML data and delivery results for their destinations. The architecture of the SA and the applied mechanisms are presented in Section VI.
- ***XML-Binder Agent:*** This agent is responsible for marshaling/unmarshaling XML data to/from Java Objects. For incoming XML data, it unmarshals XML data to runtime objects and generates OS-supported schedulable tasks (i.e., threads). The generated OS tasks are sent to the Admission Agent. For outgoing information, e.g., when the Execution Agent has feedback corresponding to a device, the respective feedback objects are marshaled to XML data and sent to the SA. The SA then signs, encrypts, and sends it out. It is worthy to note that because only those tasks compatible with predefined FB XML DTD schema can be unmarshaled or marshale. This module contributes to the execution safety partially. In other words, requests which are not compatible with the FB XML DTD schema will be blocked out of the SDCG.

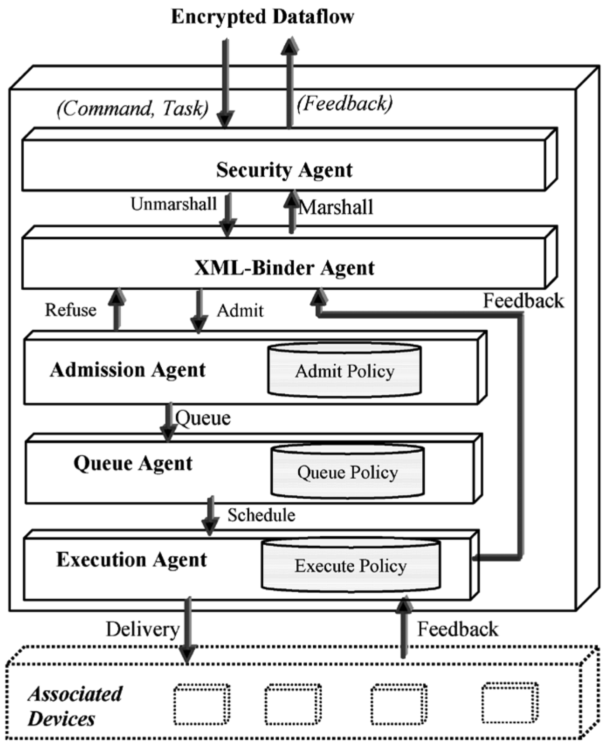

Fig. 7. Architecture of SDCG.

- ***Admission Agent:*** The Admission Agent checks whether a newly requested task is available to be executed according to the admission policy. For example, an admission policy specifies how much of a device's processing capacity is reserved for real-time tasks. If there is insufficient processor bandwidth available, the new Task will be rejected for registry in the Task Queue. This module is an important part of the Gateway to guarantee execution safety of the distributed low-level control.
- ***Queue Agent:*** When a new Task passes the admission test, a Task ID will be assigned to it. The task will then be put into the Task Queue. The first task in the queue is always waiting for execution in the next scheduling period.

  The Queue Agent maintains the queuing policy for ordering queried tasks. Specified rules can be set, for example, the rule of "earliest deadline first" is used as the most appropriate ordering rule for real-time device control. In addition, using different rules can constitute different task queues that are suitable for different types of devices and task control requirements. For example, the task queuing policy may be set according to the following factors.
  - Priority of each task.
  - Task entry time.
  - Fairness-guarantee that each waiting task will have a 'fair' opportunity to run.
  - Desired completion time of each task.
- ***Execution Agent:*** The Execution Agent is responsible for the execution of a task. It gets the first task from the task

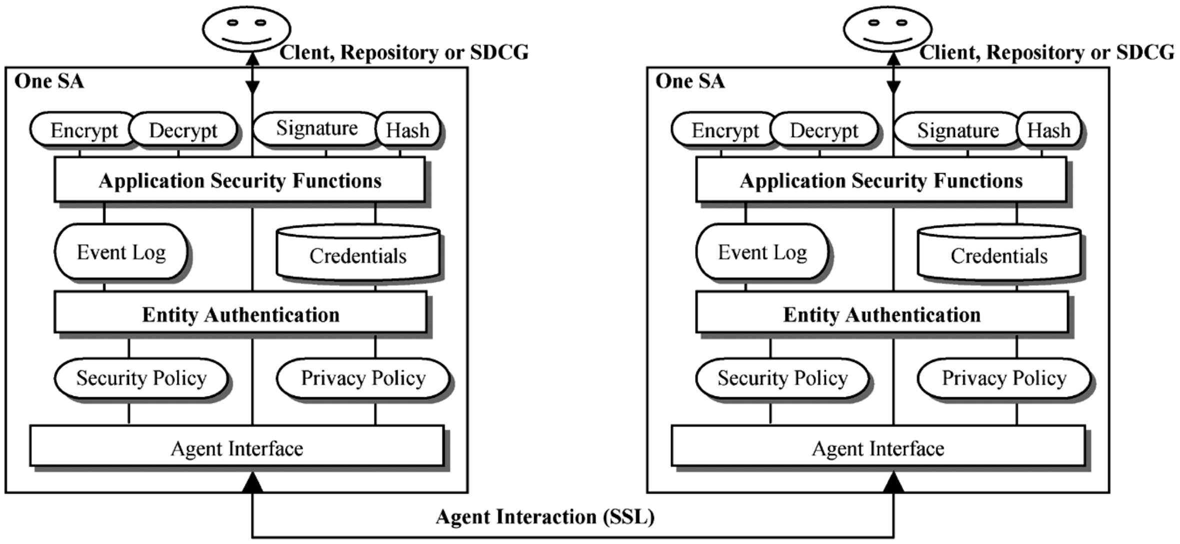

Fig. 8. Structures and interaction of two SAs.

queue and uses the corresponding device API to begin the execution of that task on the specified device.

VI. Security Agent and Its Security Mechanisms

In our proposed model, the SA is a key module in each of the clients, repositories or SDCGs to support the security control. It addresses the security challenges of authenticity, integrity, confidentiality and so on. Fig. 8 illustrates the structures and interaction relationships of two SAs. These two SAs may belong to any two of the client, repositories, or SDCG.

Between two interactive agents, one agent sends a request to another agent through agent interfaces. Before a request is serviced, the receiver always verifies whether or not the owner of the request has the rights to access the services using the entity authentication mechanisms in the application layer. If it is successful, the request is signed and encrypted by the sender for data confidentiality and nonrepudiation with the application security functions. The involved security mechanisms are discussed below.

A. Entity Authentication Mechanisms

There are two schemes, A and B, for entity authentication. It is the SA that has the responsibility to deduce, negotiate, and adjust which scheme should be applied for an interaction.

Comparatively, scheme A is a weak authentication mechanism that is based on password. It is called "weak" because the password chosen is often short or obvious to allow easy memorization. These sorts of passwords may be easily broken by brute-force attacks [12]. However, the advantage of using a weak authentication mechanism is that it suits the lightweight systems, especially in the environment where clients do not use a public-key infrastructure. Fig. 9(a) illustrates the processing of the scheme A—weak entity authentication protocol.

The protocol is described as follows.

- The claimant agent computes a hashing value with its password and timestamp as input. It then sends its identity (ID), timestamp (T), and hashing result $(h(\mathrm{Password}, T))$ to the verifier agent.

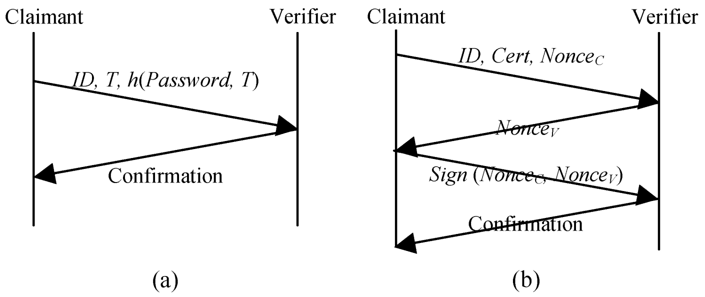

Fig. 9. Weak and strong entity authentication protocol. (a) Scheme A. (b) Scheme B.

- After receiving the above message, the verifier agent verifies whether the timestamp is acceptable, and whether the received hashing value is same as the hashing value over the password and timestamp computed with verifier agent. If the verification is successful, the verifier agent sends a successful confirmation message to the claimant agent. Otherwise, it sends a failed message to the claimant agent.

Scheme B is a strong authentication mechanism. It uses a challenge-response authentication protocol. In order to make a strong authentication scheme work, each of the clients, repositories, and SDCGs will be assigned a unified public-key certificate, which consists of a key pair consisting of a public key and a private key provided by a designated certificate authority (CA). The public key is used as the identification of the key holder, and for data verification, and data encryption functions. The private key is used to form a signature on the credential and request, and to provide other cryptographic functions such as decryption and session key exchange. The public key is published on the public-key tree so that every entity can obtain it. Of course, it is essential that the individual entities keep the private keys secret.

Fig. 9(b) illustrates the processing of scheme B. The protocol is described as follows:

- The claimant agent first sends its identity (ID), certificate (Cert) and a random number (Nonce_C) to the verifier agent.

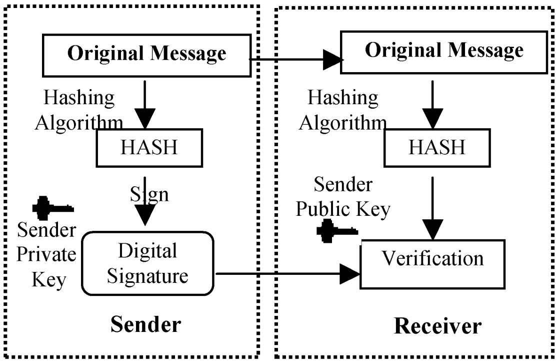

Fig. 10. Scheme 1-Authenticity, Non-repudiation, and Integrity.

- Upon the reception, the verifier agent verifies that the certificate is correct, and responds with another random number (Nonce_V) to the claimant agent.
- After receiving the Nonce message from the verifier agent, the claimant agent signs the two Nonces together using his private key, and sends the signature value $(\text{Sign}(\text{Nonce}_C, \text{Nonce}_V))$ to the verifier agent.
- Upon the reception, the verifier agent verifies whether the signature value is correct using the public key of the claimant agent, and then sends a confirmation message to inform the claimant agent the verification is successful or not.

B. Application Security Mechanisms

Three schemes are used to provide the application layer's data authentication, confidentiality, nonrepudiation, and integrity functions. They are described as follows.

1) Scheme 1—Data Authentication, Non-Repudiation, and Integrity: Fig. 10 illustrates the process of Scheme 1, including signature and verification processes. It provides authenticity, nonrepudiation, and integrity but no confidentiality. The hash value of the original message is formed by a hashing algorithm, such as MD5 (a one-way transformation of a string of characters into usually a shorter, fixed-length value). Only the hash value of the message is signed in this scheme, avoiding the time-consuming process of signing large messages. The original message is sent with the signature. The receiver verifies the signature using the hash value and the sender's public key.

The basic protocol is described as follows.

- The sender first creates the hash value of the original message.
- The hash value then is signed with sender's private key.
- The message and the signed hash value are sent to the receiver.
- The receiver recomputes the hash value, and verifies the signature with the sender's public key and the new hash value.

2) Scheme 2—Data Confidentiality: Scheme 2 provides data confidentiality. In this protocol, the sender first randomly creates a session key, and then encrypts the message using the session key. Finally, the sender encrypts the session key using the receiver's public key, puts the two ciphertexts together, and sends them to the receiver.

3) Scheme 3—Data Authenticity, Confidentiality, Non-repudiation, and Integrity: Scheme 3 is a combination of Scheme 1 and Scheme 2. It provides full protection of the application data, including authentication, confidentiality, nonrepudiation, and integrity. The process is illustrated in Fig. 11.

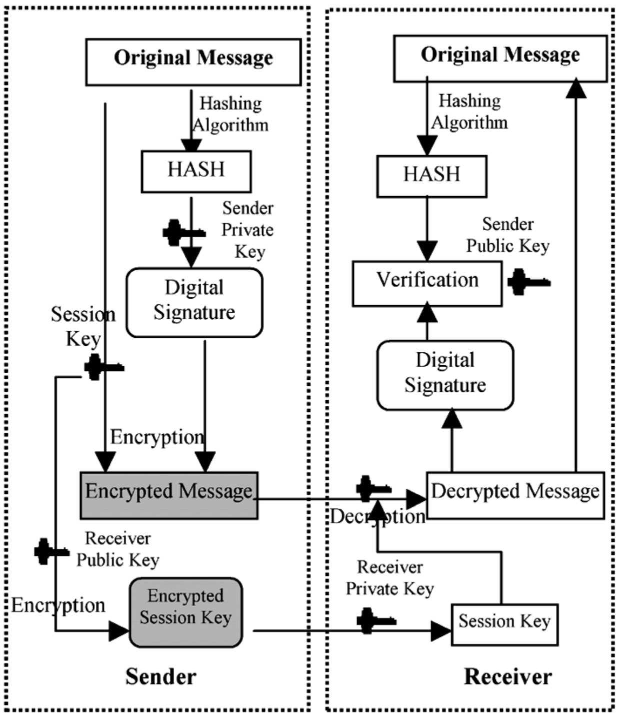

Fig. 11. Scheme 3-Authenticity, Confidentiality, Non-repudiation, and Integrity.

In this protocol, the sender first creates a hash value of the message, and then signs the hash value with her/his private key. After that, the sender puts the signature and the message together, and encrypts them using a random session key. The session key finally is encrypted with the receiver's public key. On the receiver side, after receiving the ciphertext, the receiver first decrypts the session key using his private key, and then decrypts the encrypted message using the session key. Finally, the receiver verifies whether or not the signature is correct.

VII. Implementation Considerations

In the proposed model, the SDCG, which contains the SA and other functions, is the most important part of the implementation. Two of the primary implementation issues of the SDCG are discussed as follows.

A. Real-Time Scheduling Mechanisms for SDCG

For each SDCG, new commands and tasks might arrive at any time. Most of these commands and tasks contain rigid time constraints for device control. Therefore, all of the involved SDCG behaviors must be predictable. This means the execution and associated transaction processes must be guaranteed to complete without violating the given time constraints. An SDCG must respond "fast enough" as defined by the characteristics of a command. In order to support time-critical command execution, appropriate real-time scheduling mechanisms are critical for the implementation.

Generally, the real-time scheduling approaches for time-critical applications are divided into two types: cyclic execution scheduling and preemptive scheduling. Apparently, cyclic

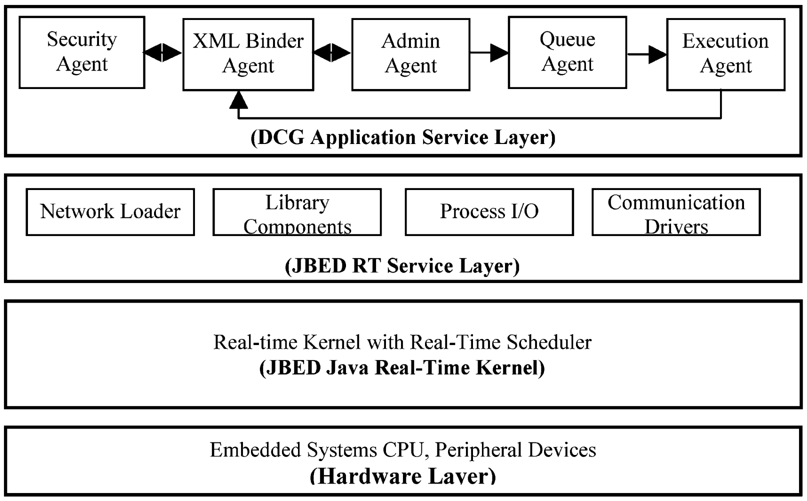

Fig. 12. Prototype implementation based on Java RTOS.

scheduling cannot satisfy the dynamic environment requirement. The preemptive scheduling mechanism is chosen to implement the SDCG.

There are two kinds of preemptive scheduling approaches: priority-based scheduling and deadline-driven scheduling. Because each SDCG may accept commands from distributed clients dynamically, where each command may be set at any priority, it is hard to coordinate which client shall use which priority, and whether a priority is set correctly. So priority-based scheduling is not an appropriate approach. For the deadline-driven scheduling approach, the execution right depended on whichever task is most quickly approaching its deadline. From the priority point-of-view, the execution priorities of tasks change on-the-fly as they approach their individual deadlines. In order to guarantee real-time performances in distributed dynamic situations, deadline-driven scheduling mechanisms are preferable over others. Using deadline-driven scheduling, the critical deadline of a dynamic command and control request can always be guaranteed at run-time.

B. Prototyping on a Real Time Java Platform

Java platform has been chosen for the prototype development, considering Java's broad popularity, simplified object model, strong notions of safety, security, as well as its multithreading supports. Java intrinsically provides methods to support "preemptive" operations, where a thread can be preempted at runtime to another thread. Although Java has not too much history of use in real-time systems because of its large size, nondeterministic behavior, and scheduling performance, there have been some considerable progresses in the development of Java-based microprocessors or real-time kernels that made Java-based real-time systems more of a reality [13]–[15].

Recently, three main approaches have emerged for real-time Java implementations: 1) Real-time Specification for Java Expert Group is chartered to produce a specification for additions to the Java platform to enable Java programs to be used for real-time applications [13]: 2) New Java Chips for directly executing Java code with real-time performance are designed and released, such as aJile Systems' Real-Time Processor for Java platform [14]; 3) Java Virtual Machine and real-time operating system kernel are combined together to supply an integrated real-time Java kernel, such as the Esmertec's JBED Real-time Java OS [15]. In our research, the third approach based was chosen to test the system prototype. Fig. 12 provides an overview of our JBED-based SDCG experimental prototype, wherein the SDCG core modules sit at the "application level" and are supported by basic JBED services and the JBED real-time kernel.

VIII. Conclusions and Future Work

Facing open and dynamic environments, effective security enhancing mechanisms and architectures are critical for the control of distributed systems and devices. This paper addresses the security challenges involved in the collaborative control of distributed device network under open and dynamic environments. By combining network security technologies, software agents, and XML processing technologies, the major security problems of authenticity, integrity, confidentiality, and execution safety are addressed. Two new modules, the SA and the SDCJ, are proposed as primary security supports. The architecture and applied security mechanisms are presented in detail.

The security of distributed systems is a multifaceted issue touching multiple disciplines, domains, departments and even cultures. For different industrial and practical application domains, security problems may have to be considered differently. For example, for the control of PLC devices in safety-related medical systems, special requirements like IEC 601-1-4 should be considered.

At the same time, there are other important issues to be investigated to extend the research. For example, there is a well-grounded concern about the effectiveness to implement security functions into one element of a system design, such as the SA proposed here. Generally, security should be built into a system at various levels to assure that the security functionality is both robust and difficult to remove. A detailed security effectiveness analysis on this aspect is under development in our research group. Furthermore, considering the limited resources and capabilities of low-level control devices, lightweight security mechanisms are important research topics to be addressed in our research plans. The trust mechanisms between different entities (clients, repositories, smart devices) are also under investigation. One of our undergoing developments following on this research is to implement a multi-agent based system to facilitate the automatic negotiation of security policies between different collaborating organizations.

REFERENCES

[1] The National Industrial Information Infrastructure Protocols (NIIIP) (2003). [Online]. Available: http://www.niiip.org/public/home.nsf

[2] P. Waurzyniak, "Electronic intelligence in manufacturing," *SME Manufact. Eng.*, vol. 127, no. 3, pp. 44–67, 2001.

[3] CNC Internetworking, e-Manufacturing Networks Inc.. (2003). [Online]. Available: http://www.e-manufacturing.com/products/internetworking.htm

[4] Scanning the Horizon-Hitachi Seiki Introduces Open CNC-PC Network Connectivity, MMS Online. (2003). [Online]. Available: http://www.mmsonline.com/articles/0699scan2.html

[5] W. Shen, F. Maturana, and D. H. Norrie, "MetaMorph II: An agent-based architecture for distributed intelligent design and manufacturing," *J. Intell. Manufact.*, vol. 11, no. 3, pp. 237–251, 2000.

[6] *Function Blocks for Industrial-Process Measurement and Control Systems*, 2000. IEC TC65/WG6, IEC-TC65/WG6 Committee.

[7] Function Blocks for Industrial-Process Measurement and Control Systems, Part 4—Rules for Compliance Profiles (2002, IEC TC65/WG6, IEC-TC65/WG6 Committee, 2002.06). [Online]. Available: http://www.holobloc.com/doc/ita/index.htm

[8] R. W. Brennan, X. Zhang, Y. Xu, and D. H. Norrie, "A reconfigurable concurrent function block model and its implementation in real-time Java," *J. Integ. Comput.-Aided Eng.*, vol. 9, pp. 263–279, 2002.

[9] J. Christensen. (2003) FBDK-The Function Block Development Kit. [Online]. Available: http://www.holobloc.com/fbdk/README.htm

[10] Y. Wei, "Implementation of IEC61499 Distributed Function Block Architecture for Industrial Measurement and Control Systems (IPMCS)," degree thesis, National Univ. Singapore, 2001/2002.

[11] *ISO/IEC 9594-8:1998, Information Technology—Open Systems Interconnecion—The Directory: Authentication Framework*. CCITT Rec. X.509 (08/97).

[12] B. Schneier, *Applied Cryptography*, 2nd ed. New York: Wiley, 1996.

[13] *Real Time Java Expert Group. Real-time Specification for Java*. Reading, MA: Addison-Wesley, 2000.

[14] The World's First Real-Time Direct Execution Processor for Java Platform (2003). [Online]. Available: http://www.ajile.com/aj100.htm

[15] *Esmertec. Jbed RTOS Package User Manual*, Esmertec, Inc., 2000.

Yuefei Xu received the B.Sc., M.Sc. and Ph.D. degrees from Northwestern Polytechnical University, China, in 1992, 1995, and 1998, respectively.

He is a Research Scientist with the Institute for Information Technology, National Research Council of Canada, Ottawa, ON. Before this, he has been working with the Intelligent Systems Group at the University of Calgary, Calgary, AB, Canada, for two years. His research interests are in the areas of information security, distributed systems, distributed industrial control systems, and embedded systems.

Ronggong Song received the B.Sc. degree in mathematics in 1992, M.Eng degree in computer science in 1996, and the Ph.D. degree in network security from Beijing University of Posts and Telecommunications, Beijing, China, in 1999.

He was a Network Planning Engineer at Telecommunication Planning Research Institute of MII, China, and a Postdoctoral Fellow at University of Ottawa, Ottawa, ON, Canada. Since 2001, he has been a Research Officer at the National Research Council of Canada, Ottawa, ON. His research interests are network security, e-commerce, privacy protection, trust management, and agent-based applications.

Larry Korba is the group leader of the Information Security Group, Institute for Information Technology, National Research Council of Canada, Ottawa, ON. He is currently involved in several projects related to the development security technologies. His research interests include network security, privacy protection, and computer supported collaborative work.

Lihui Wang received the B.Sc. degree from Beijing University of Printing, China, in 1982, and the M.Sc. and Ph.D. degrees from Kobe University, Japan, in 1990 and 1993, respectively.

He is a Research Officer, Integrated Manufacturing Technologies Institute, National Research Council of Canada (NRC), Ottawa, ON. He has worked for two years at Kobe University and two years at TUT, another national university in Japan, as an Assistant Professor before joining NRC. His research interests include CAD/CAM/CAPP/CAE, distributed intelligent systems, as well as Java and web-based systems. He has published over 100 scientific papers in refereed journals and conference proceedings.

Dr. Wang is currently a registered Professional Engineer, a senior member of SME, a member of ASME, and an Adjunct Professor at University of Western Ontario, London, ON.

Weiming Shen (SM'02) received the B.Sc. and M.Sc. degrees from Northern Jiaotong University, China, in 1983 and 1986, respectively, and the Ph.D. degree in 1996 from the University of Technology of Compiegne, France.

He is a Senior Research Scientist at the Integrated Manufacturing Technologies Institute, National Research Council of Canada, Ottawa, ON. He is an Adjunct Professor in systems design engineering at the University of Waterloo, Waterloo, ON, and an Adjunct Research Professor in software engineering at the University of Western Ontario, London, ON. He has been working on intelligent agents and their applications to engineering design, intelligent manufacturing, and virtual enterprises for about 12 years. He has published one book and about 180 papers in scientific journals and international conferences/workshops, and co-edited ten conference/workshop proceedings in the related areas. He is an associate editor or an editorial board member of six international journals and served as guest editor for six other international journals.

Dr. Shen is a member of ASME, ACM, AAAI, and a registered Professional Engineer in Ontario.

Sherman Lang received the B.A.Sc., M.A.Sc., and Ph.D. degrees in systems design engineering from the University of Waterloo, Waterloo, ON, Canada.

He has held positions with the Laboratory for Biomedical Engineering of the Medical Engineering Section, Division of Electrical Engineering, National Research Council of Canada (NRC), Ottawa, ON, the Autonomous Systems Laboratory, Institute for Information Technology, NRC, and the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong. He is a Senior Research Officer with the NRC's Integrated Manufacturing Technologies Institute, London, ON, and Group Leader of the Distributed Manufacturing Group, which focuses on intelligent distributed control and planning and reconfigurable manufacturing systems. His research interests include mobile robots, autonomous guided vehicles, mechatronic systems, vision and sensor systems, graph theoretic modeling of mechanisms, parallel kinematic mechanisms, sensor guided intelligent robotic control, system design, , and manufacturing systems.