Design and Implementation of Secure Communication Channels over UPnP Networks[*]

Jiunn-Jye Lee, Chun-Ying Huang, Li-Yuan Lee, and Chin-Laung Lei
{jye,huangant,conky}@fractal.ee.ntu.edu.tw, lei@cc.ee.ntu.edu.tw
National Taiwan University

Abstract

The scale of smart living spaces can be varied from small, e.g. a household, to large, e.g. a building or a campus, scales. As the scale of the space increases, we can expect that the requirements for the two features – zero-configuration and secure data communication channels – are getting more important. The feature of zero-configuration reduces the cost to setup the network and secure data communication channels guarantee both the privacy and confidentiality of possible sensitive data transmitted in the network. In this paper, we integrated two technologies, UPnP and secure group communication techniques, to construct an almost zero-configuration secure environment for smart living spaces. A secure and flexible communication environment is constructed as follows. An UPnP controller is implemented to manage devices in the same administrative domain and hence these devices can be treated as members in the same communication group. Then, by leveraging group key management algorithms, we successfully build both point-to-point and broadcast secure channels over the UPnP network.

Keywords: *Key management, secure communication channel, UPnP network.*

1 Introduction

As the maturing of computer and communication technologies, integrating modern technologies into people's daily life becomes an inevitable trend. Traditional information systems, such as bulletin boards on campuses, touring maps in national parks, and billboards in shopping malls, can now be replaced by interconnected smarter devices. Since nodes in the same network are able to communicate with each other, services and content displayed at the user end can be presented much more interactively and dynamically.

To build a communication network that supporting intelligent devices, in addition to those basic design issues such as scalability, fault tolerance, and availability, there are two more design issues that must be taken into consideration. The first is the ease of system configuration because it could greatly reduce the cost of maintenance. For example, deploying more than hundreds of information systems in a large shopping mall may require repeating similar setups on each device. A easy-to-configure system can thus reduce the cost of deployment. Second, as the information loaded on these intelligent devices can be customized to the users, the security and privacy of data that transmitted between the device and the network must be protected.

According to the factors discussed above, we adopt the concept of the Universal Plug and Play (UPnP) Device Architecture [1] for the ease of device discovery and management. The UPnP architecture supports zero-configuration networking. When a device joins the network, it can be automatically discovered and integrated into the existing system. The device then conveys its own capabilities to other devices and also receives the information about capabilities of other devices. With the benefits brought by the UPnP architecture, service providers do not have to worry about the complicated network settings, and thus can concentrate more on the content.

Based on the UPnP architecture, to provide secure communication channels, some aspects have to be taken into account. In the proposed architecture, control messages are managed by a central control point. Since application layer services may require both unicast and multicast communication, the control point must have the ability to transform message for the two different secure channels. Furthermore, the introducing to key management algorithm should not break the zero-configuration property of UPnP architecture.

[*]This work is supported in part by the Ministry of Economic Affairs under the Grant No. 95-EC-17-A-02-S1-049, and by the National Science Council under the Grants No. NSC 95-3114-P-001-001-Y02 and NSC95-2218-E-002-038.

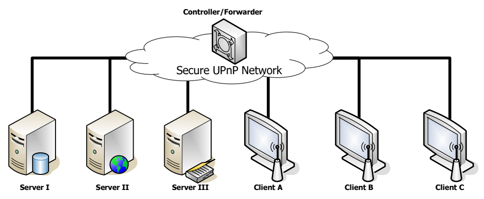

Figure 1. The system architecture of the secure UPnP environment.

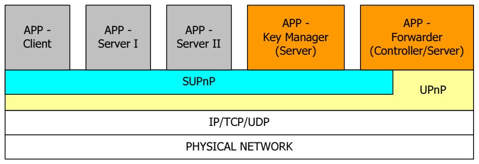

Figure 2. The protocol architecture of the secure UPnP environment.

Thus, we proposed a secure-UPnP (SUPnP) framework to integrate both the UPnP architecture and secure communication channels.

The rest of this paper is organized as follows. We first introduce the proposed system architecture in Section 2. Section 3 shows the details of the SUPnP protocol, which includes the node registration protocol, the construction of secure data communication channels and the message relaying protocol. Several aspects of the proposed framework will be discussed in Section 4 and we finally conclude in Section 5.

2 System Architecture

For the convenience of discussion, in this paper, we assume that devices are connected by a local area network (LAN). Practically, this assumption can be relaxed by the establishment of secure tunnels between devices or by appropriated configured routers. Figure 1 illustrates the proposed system architecture. A centralized control point device, abbreviated as the controller, is accounted for managing the whole system. Once a UPnP-compatible device wants to join the network, it sends an IGMP join message to introduce itself and receives IGMP messages sent from the controller. After the device has joined into the system, it will obtain its own keys through the SUPnP node registration protocol, which will be in detail in section 3.1. On completing the registration procedure, services on the device will be advertised to other existing devices.

Devices except the controller in the proposed system are categorized into client devices and server ones. Under UPnP network, it is unnecessary for the devices to have the knowledge of other devices' network settings (such as IP addresses, domain names, service ports, ..., etc.). Once a request message is generated by a client device, it will be sent to the controller through the secure unicast channel. The controller will analyze the request and then broadcast it to the back-end server devices through the secure broadcast channel. On receipt of a request, each server device parses the request header and decides whether it should process the request or not. When a server replies, the controller will forward the replied messages to the original client device. According to the demand of different service types, a client request may require one or more responses from the server devices. The detail of secure channel construction for client devices, the server devices, and the message relaying service will be further explained in Section 3.2, 3.3 and 3.4, respectively.

3 The Design of SUPnP

In this section, we explain how the secure UPnP environment is built in detail. Figure 2 shows the protocol architecture of the SUPnP design. The design of the SUPnP network is a layered design. As the under layer follows the UPnP basic device definition [2], the SUPnP protocol is able to coexist with any other UPnP devices.

Most of the smart devices are built on top of the SUPnP layer. These devices can be classified into two categories, namely the client and the server devices. By definition, the major work of a client devices is to interact with the environments and make requests to server devices. On the contrast, server devices are in charge of answering requests from clients. Beside generic clients and servers, we have two special components in the SUPnP network. One is the "key manager", which is run as a server device, and the other is the "forwarder", which is also run as a server cooperating with an UPnP controller. The key manager is responsible to maintain the relationship of devices in the SUPnP network. It also generates required secret keys for those devices that have successfully joined the secure group. Thus, when a device enters the network, it has to register to the key manager. The forwarder is a bridge between clients and servers. All the messages exchanged between clients and servers should be relayed and transformed by the forwarder.

In our SUPnP network design, when a client submits a request, the request is relayed[1] through the forwarder

[1]The design for that all messages exchanged between clients and servers are relayed through the forwarder is for convenient

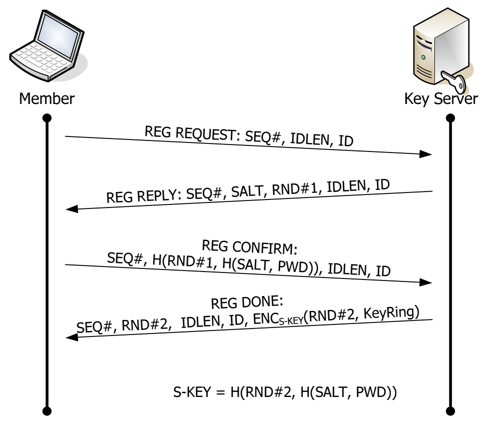
Member
Key Server
REG REQUEST: SEQ#, IDLEN, ID
REG REPLY: SEQ#, SALT, RND#1, IDLEN, ID
REG CONFIRM:
SEQ#, H(RND#1, H(SALT, PWD)), IDLEN, ID
REG DONE:
SEQ#, RND#2, IDLEN, ID, ENCS-KEY(RND#2, KeyRing)
S-KEY = H(RND#2, H(SALT, PWD))

Figure 3. The node registration protocol for new comers in the SUPnP network.

and then broadcasted to all servers. In the meantime, the client collects all the responses, which is also relayed through the forwarder, from the servers in a given time constraint. Thus, it is apparently that we need different secure channels for clients and servers because a request from a client should be known to all servers but not known to other clients. In the later parts of this section, we will introduce the registration protocol for new member to join/leave the secure group, the construction of secure channels for both clients and servers, and also how messages are relayed between clients and servers.

3.1 The Node Registration Protocol

The node registration protocol used in the SUPnP network is illustrated in Figure 3. The protocol is designed with two important ideas – simplicity and security – in mind. We would like to minimize the number of configurations that required on the devices and also the protocol of course must be secure. As a result, in our registration protocol, each device has to maintain only its device ID and password. By using a challenge-response like protocol design, it is unnecessary to send passwords as plain texts during the device registration process.

All the nodes in the SUPnP network must be registered first to access the infrastructure. The node registration protocol works as follows. When a device has joined the UPnP network and wants to be a member of the SUPnP network, it has to send a registration request (REG REQUEST) along with a randomly generated sequence number and its device ID. At this time, the new comer does not know where the server is. However, as the message is relayed through the forwarder, the registration request is then broadcasted to all the servers and only the key manager would reply for the registration protocol, which is the second registration acknowledgement (REG REPLY) message. Note that during a registration process, the sequence number and the ID is the key to uniquely identify the process on both the member side and the key manager side. Therefore, all the messages for a registration process must contain the same sequence number and identity. Thus, besides the common required fields, the REG REPLY message also contains the SALT used to protect user password and a randomly generated number RND#1.

On receipt of the REG REPLY message, the new comer has to make a confirmation to let the key manager know that it has the right password. To prove that, firstly, it has to compute a value s, which is the hash of SALT plus the password. Then, it has to compute a value r, which is the hash of RND#1 plus the value s, and finally send the value r back to the key manager. The server now should be able to make sure the identity of the new comer since it knows the RND#1, the SALT, and the password[2] of the given ID. If the user identity can be verified, the key manager can send back a registration done (REG DONE) message to the new comer. Otherwise, a REG FAIL registration failure message is returned. If the registration process is successfully finished, a symmetric key S-KEY shared between the new member and the key manager can be established by computing the hash of RND#2, which is included in the REG DONE message, and the value s. If the new comer is going to be a server in the SUPnP network, all the extra keys used for secure group communications well be sent to it encrypted using S-KEY. It should be noted that when a new comer has successfully joined or left the SUPnP network, all the corresponding user and key information should be synchronized with the forwarder, which is responsible for relaying messages between the secure client and server networks.

3.2 Secure Client Channels

As we mentioned before, we need different secure channels for clients and servers in the SUPnP network.

and security considerations. Here we just ignore the discussion of the detailed fault-tolerant and scalability issues of such a design. Readers with interests on this issue may refer to our brief discussions in Section 4.

[2]The key manager does not really need to know the plain text password for all user IDs. To enhance the security, we can only store the SALT and the hashed result of SALT plus the plain text password, instead of the plain text password itself.

The secure client channels is by nature constructed after the symmetric key S-KEY is established between the key manager and the new comer. Thus, if the new comer is a client device, which is differentiated by the device ID known to the key manager, there will be no more secret keys assigned to the new comer.

3.3 Secure Server Channels

If the new comer of the SUPnP network is a server device, all the required secure group keys should be delivered to that device with the REG DONE message. Therefore, when the symmetric key S-KEY is established, the device should also share a group key with other servers in the same communication group. We do not assume which secure group communication mechanism to be used in the SUPnP network. Most kinds of these mechanisms should be able to work with our framework. In general, key management mechanisms for secure group communications can be classified into three categories, namely centralized, decentralized, and distributed [5]. Since the network has a centralized forwarder, the number of members varies dynamically, and the main purpose of the secure communication is to broadcast requests, we suggest to use centralized key management mechanisms in the SUPnP network. For the reason of such a choice, we will discuss later in Section 4.

At the current stage, we choose logical key hierarchy (LKH) [8, 9] as the core group key management algorithm because it is simple and also efficient among all the other choices. Like other group key management algorithms, LKH also maintains a set of key encryption keys (KEKs) for members in the secure group. In LKH, the key distributor center (KDC) maintains a logical tree like the one in Figure 4. The number of leaf nodes equals to the maximum number of members (i.e. capacity) in the group. However, the KDC should assign KEKs to all the nodes in the logical tree. When a new member joins, the KDC should send all the KEKs on the path from the root node of the logical tree to the position that the member joins. For example, if a new member joins at the fifth position in Figure 4, then it has to know k_{1-8}, k_{5-8}, k_{5-6}, and k_5. The shared group secret key can be delivered to all members in the secure group by encrypting using the key k_{1-8}.

A group key management mechanism always needs to re-key, i.e., updates parts of the KEKs, when the memberships of the group changes. The purpose of re-key is to keep both forward secrecy and backward secrecy. The former prevents a left member from getting something from the secure channel and the later prevents a new comer from knowing what are transmitted before it joins. In LKH, all KEKs on the path from the joined/left member to the root node have to be updated. For example, if the fifth member in Figure 4 is left, k_{1-8}, k_{5-8}, k_{5-6}, and k_5, which are known to the left member, should be updated. The KDC is responsible to update these affected keys. At the same time, the KDC should also notify all the active members who hold the updated KEKs. To minimize the re-key overheads, the notification should be done in a one time multicast/broadcast. Suppose the KDC has updated k_{1-8}, k_{5-8}, and k_{5-6} with k'_{1-8}, k'_{5-8}, and k'_{5-6}, respectively. To minimize the cost, the KDC can encrypt $\{k'_{1-8}\}$ using k_{1-4}, encrypt $\{k'_{1-8}$ and $k'_{5-8}\}$ using k_{7-8}, also encrypt $\{k'_{1-8}, k'_{5-8}$, and $k'_{5-6}\}$ using k_6. Finally, it sends all these encrypted updated keys in one shoot. The cost for LKH is summarized here. For a group containing N members, the KDC has to maintain $(2N-1)$ KEKs. On the other hand, each member only need to store $\log(N)$ KEKs. When a re-key is required, only $\log(N)$ KEKs are affected and key updates can be done in one shoot (with a key size in an order of $\log(N)$).

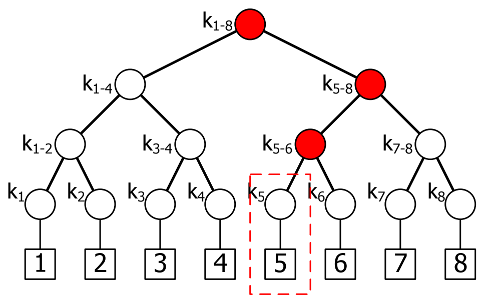

Figure 4. A sample tree topology for the logical key hierarchy (LKH) key management algorithm.

Applying LKH to our SUPnP environment is fairly simple. When a device is successfully registered with the key manager with a server identity, all the required KEKs on the path from the root to the assigned logical position for that new comer are updated first. Then, those updated KEKs are sent to the new server device using the REG DONE response.

3.4 Message Relaying

The forwarder is an important component in the SUPnP network. It is bound with the UPnP controller. The two secure channels for client and the server devices are bridged by the forwarder. When messages are exchanged between the two different secure networks, these encrypted messages must be transformed to make them understandable by message receivers. Thus, the forwarder must have the same knowledge to key manager, which includes all the symmetric keys S-KEY that established with all the members and all the KEKs used in secure group communication.

The relayed messages are handled in the following two ways. For a request message sent from a client device, since the message must be encrypted using the shared S-KEY between the client and the key manager, the forwarder can decrypt the request and then broadcast the request securely using the group secret key. Note that all the server devices in the SUPnP network can read the request and response to it depending on what kinds of service they provide. On replying a request, a server can encrypt the response by using either its symmetric key or the group secret key. In either way, the forwarder is able to decrypt the response and then re-encrypt the message using the symmetric key of the receiver.

4 Discussions

In this section, we discuss several aspects related to the proposed architecture and protocols. The following issues will be addressed: The choice of centralized group key management; fault-tolerant and scalability of the UPnP controller/SUPnP forwarder; the co-existence of SUPnP and UPnP networks; the possibility of extending such an architecture to deploy over wide area networks; and finally, a guide to develop smart applications over the SUPnP infrastructure.

4.1 Centralized Group Key Management

The design choice of the secure group key management mechanism, as we mentioned before, is centralized key management mechanism. This choice is based on the following reasons. First, since the original UPnP network already has a (centralized) controller, it is naïve for us to leverage that device. The second reason is that the server devices in the SUPnP network can be joined or left dynamically. Therefore, it may be not suitable to use distributed key management mechanisms such as [6] because such key management mechanisms require members to know each other and then compute the shared secret key using contributory protocols. In our SUPnP network, to reduce the time to query group memberships and repeating the contributory protocols, we decide not to use distributed mechanisms. As to decentralized mechanisms [3, 4], for the reason of dynamically changed memberships and the use of only secure broadcast channels, we also do not consider decentralized ones. Decentralized key management mechanisms divide a whole group into several subgroups and thus each subgroup must have a subgroup leader. Electing subgroup leader may be a problem because members in the SUPnP network are not supposed to know which member will stay longer in the network. Such an uncertainty may lead to a very unstable subgroup. On the other hand, since the major use of the secure group communication channel is to broadcast requests, it is not necessary to relay through subgroup leaders.

Centralized key management mechanisms are easier to implement and maintain. The only thing for all the members in the group is to find the key manager. The cost for key updates can be also reasonably bounded in log scales. However, problems of centralized key management mechanisms are the ability for fault-tolerant and the scalability. We will address this issue later.

4.2 Fault-Tolerant and Scalability

The techniques used to build services on clusters are getting matured. To achieve the requirement for fault-tolerant and scalability, we can apply techniques similar to that in [7]. In summary, we can setup multiple controllers and make them all on-line at the same time. The states of memberships, secret keys, and logical tree topologies can be synchronized between these devices using the same protocol as synchronizing between the key manager and the forwarder. Load can be shared by dispatching or migrating members of the SUPnP network to different controllers and these orphaned members caused by the failure of controllers can be also picked up by active controllers.

4.3 Co-Existence of SUPnP and UPnP

The SUPnP is built on top of an UPnP basic device. Without the SUPnP support, devices should be able to send unencrypted messages to each other. To do so, the SUPnP must have the ability to tell which messages should be processed by the SUPnP layer and which messages should bypass the SUPnP layer. For this reason, we encapsulated the SUPnP message with a dedicated protocol header, as shown in Figure 5. In the header, all the first six fields are in 16-bit length and values should be stored in big-endian. The data are placed right after the sixth field. The "magic" field store a constant number. All the SUPnP data should beginning with this magic number. The flag field indicates how to process this message. It indicates whether the message is a control message or a data message, is encrypted or not, is sent by a client or by a server, us-

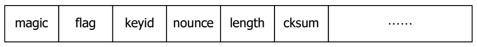

Figure 5. The SUPnP protocol data and control message header.

ing unicast channel or broadcast channel, etc. Then, there is a keyid field to indicate which key is used to decrypted the message (if its encrypted). Finally, there is a field to store the total length of this message including data and a checksum to verify the integrity of the message header. To tell whether a message is a SUPnP message or not, we can check both the constant magic number and the checksum value.

In the design of SUPnP, it is able to allow general unencrypted messages, unencrypted SUPnP messages, and encrypted SUPnP messages to pass the SUPnP layer. However, to make sure that all the messages are delivered in a secure manner, we suggest not to allow all unencrypted data messages bypass the SUPnP layer.

4.4 Extension of the SUPnP Network

As the scale of the smart living space grows, such a space may be built across several different subnetworks. The UPnP architecture is originally proposed for personal/home environment. Thus, the SUPnP network is also bound inside a local area network. Since it is possible to construct a virtual private local area network over the Internet. In this way, we can extend the SUPnP network across the network boundaries. However, this requires a device to have more network configurations beforehand. When a device is going to enter the SUPnP network, it must have proper configurations to access the virtual private network. Instead of having each device capable of VPN access abilities, another better solution might be to have those cooperated subnetworks formed a virtual local area network. When virtual local area networks are constructed at the network level, it is unnecessary to touch all the devices.

4.5 Application Development Guidance

Based on the SUPnP infrastructure, content providers can easily implement their services on the network. To build a new service, content providers have to design their own request and reply message formats. These messages can be injected into the network by customized front-end clients and back-end servers using our SUPnP library. After finishing the registration procedures for both the clients and servers, the new application will be integrated seamlessly and ready to use in the SUPnP network.

5 Conclusions and Future Works

The UPnP technique was originally used to simplify the configuration of personal or home network devices. In this paper, we extend the UPnP technologies and build an intelligent secure network. Keep the ideas of simplicity and security in mind, the proposed protocol is suitable for the construction of a flexible and easy-to-use smart living spaces. Our future works focus on further analyses on the proposed protocols, extending the scalability of the proposed architecture, and develop applications that leverage the almost zero-configuration secure communication environments. To simplify the deployment of the SUPnP network, we also prepare to construct the SUPnP network over wireless environments.

References

[1] UPnP device architecture version 1.0.1. UPnP Forum, Dec. 2003.

[2] S. Lawrence. UPnP basic device definition version 1.0. UPnP Forum, Dec. 2002.

[3] S. Mittra. Iolus: a framework for scalable secure multicasting. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 277–288. ACM SIGCOMM, 1997.

[4] R. Molva and A. Pannetrat. Scalable multicast security with dynamic recipient groups. *ACM Transactions on Information and System Security*, 3(3):136–160, 2000.

[5] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys (CSUR)*, 35(3):309–329, 2003.

[6] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to group communication. In *CCS'96: Proceedings of the 3rd ACM conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.

[7] P.-L. Tsai, C.-Y. Huang, Y.-Y. Huang, C.-C. Hsu, and C.-L. Lei. A clustering and traffic-redistribution scheme for high-performance ipsec virtual private networks. In *HiPC'05: Proceedings of the International Conference on High Performance Computing, LNCS 3769*, pages 432–443. Springer, 2005.

[8] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. *RFC 2627*, June 1999.

[9] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, 2000.