Design and Implementation of Headend Servers for Downloadable CAS

Soonchoul Kim[1], Hyuncheol Kim[2], and Jinwook Chung[1(✉)]

[1] Department of Computer Engineering, Sungkyunkwan University, Suwon, Korea
{choulsim,jwcheong}@skku.edu
[2] Department of Computer Science, Namseoul University, Cheonan, Korea
hckim@nsu.ac.kr

Abstract. This paper presents the design and implementation of headend servers for a downloadable conditional access system (DCAS) that can securely transmit CA code via a broadband channel. To design DCAS headend server, we define core functions to be performed in the headend and categorize them into four sections such as authentication, provisioning, personalization and key management. In order to verify the stability and effectiveness of the implemented headend servers, we construct a testbed using them and a legacy cable headend system in a laboratory. The experimental results show that the DCAS headend servers are well designed.

Keywords: DCAS · Downloadable CAS · DCAS network protocol

1 Introduction

The Conditional Access (CA) application modules, which execute methodology to extract the secured keys for descrambling and decryption, are locked-in into STBs as unchangeable elements. DCAS uses a secure microprocessor (SM) soldered onto a circuit board instead of a removable. DCAS technology can remove the lock-in issue for CAS or STB vendors, and it is also much more flexible and easier to manage and distribute a CAS module onto an STB [1]. As shown in Fig. 1, the DCAS headend server communicates with a trusted authority (TA) to authenticate an STB which accesses to a cable network. The DCAS host includes a secure micro (SM) and a transport processor (TP) for supporting DCAS [4]. The SM performs a DCAS protocol and stores a CA code transmitted from a download server. The TP is used for decryption of video encrypted by the CA code [2].

In this paper, we improve the previously proposed protocol for reducing the processing time of DCAS messages without degrading a security level and apply it to a DCAS headend server. In Sect. 2 we analyze the security vulnerability about the previous proposed DCAS network protocol and describe the improved protocol to solve known problems. Also, it includes an efficient session control management. In the Sects. 3 and 4, the design and implementation of DCAS headend server are described and experimental results are provided respectively. Finally, the conclusion is followed.

K. Kim and N. Joukov (eds.), *Information Science and Applications 2017*,
Lecture Notes in Electrical Engineering 424, DOI 10.1007/978-981-10-4154-9_90

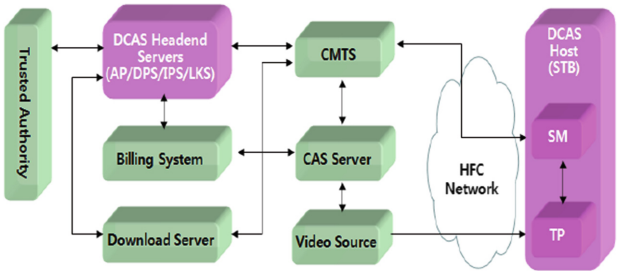


Fig. 1. System architecture for DCAS in HFC network.

2 Improved DCAS Protocol and Session Control Management

A DCAS network protocol requires the mutual authentication between DCAS headend server and DCAS host so that two parties establish a session key securely. The session key is used for content encryption and key transmission. The designed protocol is based on pre-shared key and related in 3rd party authentication among TA, AP, and SM. Accordingly, it is very important that the needed key materials among them should be delivered without an exposure. We estimated the secrecy of the key establishment phase on a formal security analysis tool of Burrows-Abadi-Needham logic [6].

The designed DCAS protocol include the initial assumptions which both TA and SM share a pre-shared key(Ki), and Session_ID, NONCE_SM, RAND_TA, Kc are randomly generated every session, and an SSL session key(KBS) is applied between TA and AP.

$$A| \equiv A \overset{K_{AB}}{\longleftrightarrow} B \; B| \equiv A \overset{K_{AB}}{\longleftrightarrow} B \tag{1}$$

$$A| \equiv B| \equiv A \overset{K_{AB}}{\longleftrightarrow} B \; B| \equiv A| \equiv A \overset{K_{AB}}{\longleftrightarrow} B \tag{2}$$

Let's generalize the protocol messages related to session key generation in order to verify (1) and (2). In (3) B can decrypt the message using B's private key as shown in (9). We can get (10) by message syntax rule from the initial assumption and (9). In (10) B cannot confirm the freshness of RA, therefore logical postulates cannot be applied to (3) no more. Because (4) and (5) are communicating over a known SSL protocol and KBS is secured, they are verified for themselves.

In (6), A can decrypt the message using A's private key as shown in (11). We can get (12) by message syntax rule from the initial assumption and freshness rule. Also, we can get (13) from nonce verification rule, and additional logical postulates. We can get (14) by belief rule in (13). In the same method, (7) and (8) can be concluded in (15) and (16) respectively.

$$\begin{aligned} &\text{AuthRequest message: A} -> \text{B} \\ &\{R_A, \{ID_A\}_{K_S}\}_{K_B}, \{R_A, \{ID_A\}_{K_S}\}_{K_A^{-1}} \end{aligned} \tag{3}$$

$$\text{AuthRequest message: B} -> \text{S} \quad \{\{ID_A\}_{K_S}\}_{K_{BS}} \tag{4}$$

$$\text{AuthResponse message: S} -> \text{B} \quad \{R_{S1}, R_{S2}\}_{K_{BS}} \tag{5}$$

$$\text{AuthResponse message: B} -> \text{A} \quad \{R_A, R_{S1}\}_{K_A}, \{R_A, R_{S1}\}_{K_B^{-1}} \tag{6}$$

$$\text{SKeyShare message: A} -> \text{B} \quad \{R_A^{'}, R_A\}_{K_B}, \{R_A^{'}, R_A\}_{K_A^{-1}} \tag{7}$$

$$\text{SKeyShareConfirm message: B} -> \text{A} \quad \{R_A, A \overset{K_{AB}}{\leftrightarrow} B\}_{K_{AB}}, \{R_A^{'}, \{R_A, A \overset{K_{AB}}{\leftrightarrow} B\}_{K_{AB}}\}_{K_B^{-1}} \tag{8}$$

$$B \lhd R_A, \{ID_A\}_{K_S} \quad B \lhd \{R_A, \{ID_A\}_{K_S}\}_{K_A^{-1}} \tag{9}$$

$$B| \equiv A| \sim (R_A, \{ID_A\}_{K_S}) \tag{10}$$

$$A \lhd R_A, R_{S1} \quad A \lhd \{R_A, R_{S1}\}_{K_B^{-1}} \tag{11}$$

$$A| \equiv \#(R_A, R_{S1}) \quad A| \equiv B| \sim (R_A, R_{S1}) \tag{12}$$

$$A| \equiv B| \equiv (R_A, R_{S1}) \quad A| \equiv B| \sim R_{S1} \tag{13}$$

$$A| \equiv B| \equiv R_{S1} \tag{14}$$

$$B| \equiv A| \sim R_A^{'} \tag{15}$$

$$B| \equiv \#(A \overset{K_{AB}}{\leftrightarrow} B) \tag{16}$$

But because B cannot confirm that A trusts KAB, logical postulates cannot be progressed further. By applying BAN logic to the designed protocol, it has been found that AP server needs the fact that SM module trusts the session key (KAB). Accordingly, the proposed DCAS protocol need be improved abut known vulnerabilities. To solve it, a SKeyShare message includes a hash value of the session key which is generated by SM module. The SM's session key is regarded as a temporary key until it is confirmed from AP server.

AP server verifies the own hash value of the session key which generates together key materials extracted from the SKeyShare message. The DCAS network protocol is to accomplish session key share in order to deliver SM client for safe keeping onto STB. Figure 2 shows the timing that session keys are generated, validated, and destroyed. Firstly, the announcement phase (DCASAnnounce, DPRequestInfo) is to inform MSO's DCAS region and information needed to invoke terminals.

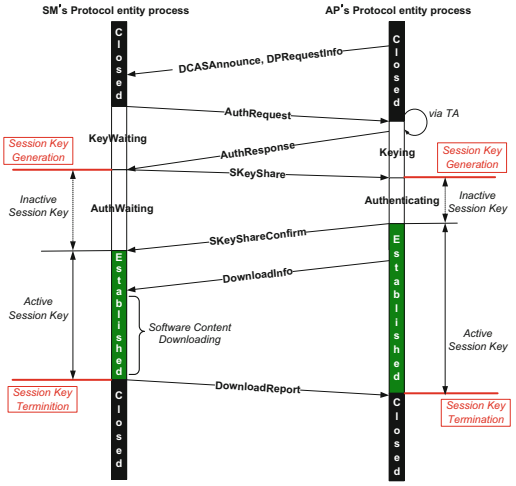


Fig. 2. Session request and response in DCAS network protocol.

The AP periodically sends out announcement messages to STBs including SM. Secondly, the keying phase (AuthRequest/AuthResponse) is to request, register, and pair keying information among SM, AP, and TA. The trust for keying information is guaranteed via TA. Thirdly, the authentication phase (SKeyShare/SKeyShareConfirm) is to establish the secured channel.

3 Design and Implementation of DCAS Headend Servers

A configuration of DCAS headend is shown in Fig. 3. The DCAS headend consists of four servers, authentication proxy (AP), DCAS provisioning server (DPS), the local key server (LKS) and integrated personalization server (IPS). The AP authenticates SM in a DCAS host according to DCAS protocol and generates several encryption keys. The DPS manages an SM configuration and distributes a policy for CA download. The LKS stores all keying information in the headend and provides a secure interface for the

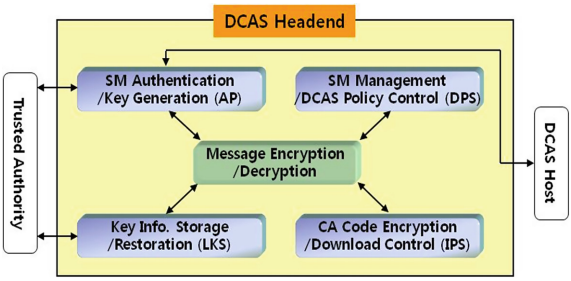


Fig. 3. The configuration of DCAS headend.

retrieval of key records. The IPS encrypts a CA code using the key delivered from AP and controls a mechanism for downloading.

The AP server includes five function blocks as shown in Fig. 4. The DPPB performs DCAS protocol for the mutual authentication and f or downloading a CA code. To reduce the processing time of DCAS messages without degrading a security level, we apply an HMAC instead of an RSA signature for message authentication and add several parameters to strengthen a security level. The SCB controls a creation and deletion of session related to a DCAS host. The ABM authenticates a DCAS STB with TA using information transferred from SCB and transmits the result to an SCB. The KMB stores keying information into DB and simultaneously transmits it to LKS. The DCB delivers a download command to IPS using download policy received from DPS when an authentication process is finished.

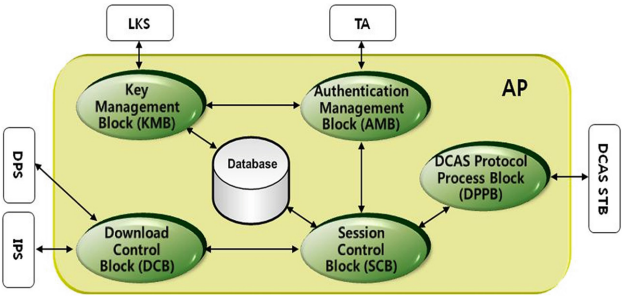


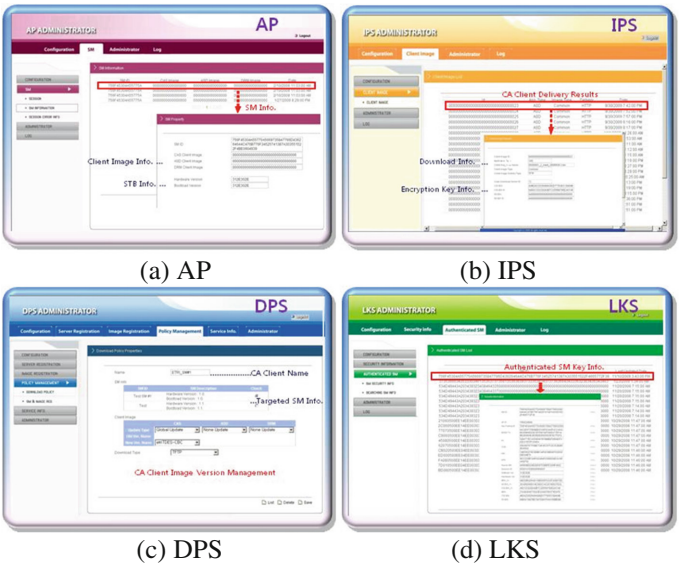
Fig. 4. Block diagram of AP server

4 Experimental Results

By testing various conditional environments, we verified that any SM or AP does not generate any unexpected state under even random message flow. The experimental conditions are as follows:

- Condition – 1: the STB with virgin state SM attachés into DCAS network, and then authenticates with AP. Successfully the STB downloads and runs SM client.
- Condition – 2: the STB with authenticated SM receives update info of SM client from AP, and then authenticates with AP.
- Condition – 3: the STB with authenticated SM receives update info of SM client from AP, and then stops in process of authentication with AP.
- Condition – 4: the STB with unauthorized SM tries authentication request in order to obtain SM client, and then sends a fake info to AP.

Figure 5 shows four DCAS headend servers. In the experiment test, a CA code was downloaded successfully from the DCAS headend to the DCAS host according to the DCAS protocol and it was installed in the SM automatically.



(a) AP (b) IPS

(c) DPS (d) LKS

Fig. 5. Implemented DCAS headend servers

5 Conclusion

In this paper, we designed and implemented DCAS headend servers for downloading a CA code securely via a broadband channel. According to the categorized core functions into four sections, four headend servers for authentication, provisioning, personalization and key management was designed and implemented. In order to reduce the processing time of DCAS messages without degrading a security level, we improved the previously proposed DCAS protocol and applied it to a DCAS headend server. For verifying the stability and effectiveness of the implemented headend servers, we construct a test-bed using them and a legacy cable headend system in a laboratory. Through experimental results, we confirmed that our enhanced DCAS protocol and the implemented DCAS headend servers can operate stably, securely, and effectively.

References

1. Lookabaugh, T., Fahrny, J.: Openness and secrecy in security systems: polycipher downloadable conditional access. In: The Cable Show Conference, May 2007
2. Jeong, Y., et al.: A noble protocol for downloadable CAS. IEEE Trans. Consum. Electron. **54**(3), 1236–1243 (2008)
3. Xu, J., Feng, D.: Security flaws in authentication protocols with anonymity for wireless environments. ETRI J. **31**(4), 460–462 (2009)
4. Koo, H.-S., et al.: Key establishment and pairing management protocol for downloadable conditional access system host devices. ETRI J. **32**(2), 204–213 (2010)

5. Chen, T.-H., Shih, W.-K.: A robust mutual authentication protocol for wireless sensor networks. ETRI J. **32**(5), 704–712 (2010)
6. Burrows, M., Abadi, M., Needham, R.M.: A logic of authentication. Proc. Roy. Soc. London A **426**, 233–271 (1989). A preliminary version appeared as Digital Equipment Corporation Systems Research Center report No. 39 (1989)