EURASIP Journal on Applied Signal Processing 2002:9, 975–980

A DSP Based POD Implementation for High Speed Multimedia Communications

Chang Nian Zhang
Department of Computer Science, University of Regina, TRLabs, SK, Canada S4S 0A2
Email: zhang@cs.uregina.ca

Hua Li
Department of Mathematics and Computer Science, University of Lethbridge, Lethbridge, Alberta, Canada T1K 3M4
Email: huali@cs.uleth.ca

Nuannuan Zhang
Department of Computer Science, University of Regina, TRLabs, SK, Canada S4S 0A2

Jiesheng Xie
Department of Computer, China Agriculture University, Beijing 100083, China

Received 20 August 2001 and in revised form 10 August 2002

In the cable network services, the audio/video entertainment contents should be protected from unauthorized copying, intercepting, and tampering. Point-of-deployment (POD) security module, proposed by OpenCable™, allows viewers to receive secure cable services such as premium subscription channels, impulse pay-per-view, video-on-demand as well as other interactive services. In this paper, we present a digital signal processor (DSP) (TMS320C6211) based POD implementation for the real-time applications which include elliptic curve digital signature algorithm (ECDSA), elliptic curve Diffie Hellman (ECDH) key exchange, elliptic curve key derivation function (ECKDF), cellular automata (CA) cryptography, communication processes between POD and Host, and Host authentication. In order to get different security levels and different rates of encryption/decryption, a CA based symmetric key cryptography algorithm is used whose encryption/decryption rate can be up to 75 Mbps. The experiment results indicate that the DSP based POD implementation provides high speed and flexibility, and satisfies the requirements of real-time video data transmission.

Keywords and phrases: point-of-deployment, DSP, cellular automata, copy protection, ECDSA, DH key exchange.

1. INTRODUCTION

The next generation of the cable networks requires that a security module should be built and separated from the host devices (set top boxes and integrated digital televisions) in order to facilitate commercial sale of navigational devices. The point-of-deployment (POD) security module is being developed to satisfy these *separable security* requirements and to enable retail availability of Host devices [1, 2, 3].

The POD module supports two major functions.

(1) The POD will provide the cable operator with a secure device at the customer's location.

(2) The POD will act as a translator so that the Host device will only have to understand a single protocol, regardless of the type of network to which it is connected. Since the draft of the specification of the POD module was released in fall 1997, several POD products have been reported. All of them are the application specific integrated circuits (ASIC), and use data encryption standard (DES) as the preliminary technique for content encryption/decryption. But DES has been proved not secure enough and will be replaced by the new standards. Moreover, due to the nature of cable network services, different applications require different security levels. It is desirable for POD to provide versatile cryptography schemes. On the other hand, since the current specification of the POD module has not been accepted as an international standard, any further modifications of the standard will cause redesigning and rebuilding of the ASIC POD.

In order to provide a low cost and flexible POD, a DSP based POD implementation is proposed in this paper which satisfies the requirements of real-time video data transmission and can be applied in different security levels. The

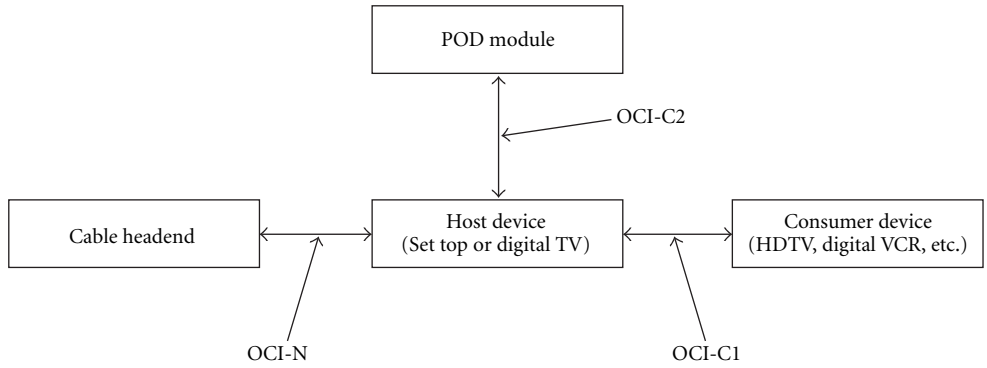


Figure 1: OpenCable network and consumer interfaces.

outline of the remainder of the paper is as follows. Section 2 introduces the POD security module including the overview of POD, its functionalities, and algorithms used in POD. Section 3 presents the POD implementation based on DSP. Section 4 is the conclusion.

2. FUNCTIONS OF POD MODULE

The set top box (STB) is a commonly used interface between digital television and the functions accessible via cable network in the architecture of next-generation television and video systems. It attaches a point-of-deployment (POD) security plug-in module to provide the security and copy protection of the contents.

Figure 1 illustrates logically how the POD module interface connects with other OpenCable interfaces. In Figure 1, OCI-N (OpenCable interface network) is the interface between a cable network and the Host device. OCI-C1 (OpenCable interface consumer 1) is the interface between a Host device and a digital consumer device. OCI-C2 (OpenCable interface consumer 2) is the interface between a Host device and the POD module.

The primary functions of the OpenCable POD module include: (1) provide conditional access to a Host device; (2) provide communication and control between the headend and the Host device. The POD module decrypt the contents under control of the headend and re-encrypt the contents for the purpose of copy protection between the POD module and Host device. Typically, the POD is authorized by the conditional access system to decrypt contents, and authorizes the Host by delivering either clear or CP (copy protection) encrypted content. The content passing the POD interface can be one of the following three formats: (1) Cleartext, (2) Passing through, (3) Rescramble. The copy protection between the POD and the Host works as follows.

Step 1 (Initialization of the POD and the Host evaluation). When the POD is powered on, it checks if the Host supports OpenCable™ content protection by checking the availability of the CP resource and verifying the authenticity of the device certificate.

Step 2 (Host authentication). The POD retrieves the Host certificate Data to initiate the authentication procedure and the Host replies to it. After this exchange, both the POD and the Host come up with the authentication key.

Step 3 (Key exchange). The POD sends its DH (Diffie-Hellman) public key to the Host and requests the Host's DH public key and then the Host sends its DH public key to the POD. After this exchange, both the POD and the Host come up with a common secret value. By using a method covered by intellectual property, they establish the shared secret keys derived from the Host authentication process.

Step 4 (Interface encryption). The POD uses the secret key to encrypt the content.

The cryptography schemes used in POD include:

(1) Elliptic curve digital signature algorithm (ECDSA), which is used in the Host authentication process for signing and verification.
(2) Diffie-Hellman (DH) public key agreement algorithm, which provides a method for POD and Host to compute a shared secret value, that is, used in the content encryption/decryption key generation.
(3) SHA-1 (secure hash algorithm) [4], which is used in the digital signature algorithm to generate a message digest of length 160 bits. For the POD, the SHA-1 algorithm is used for Host certificate signature verification, authentication key generation and copy protection key generation.
(4) Elliptic curve key derivation function (ECKDF) algorithm, which is used to generate the key for the content protection.

Moreover, a random number generator is included to generate DH private keys which will be compliant with the SHA-1 based algorithm. Each OpenCable device has a unique seed value which is set by the manufactory.

Figure 2 illustrates the cryptographic functions used in the POD copy protection.

3. A DSP BASED POD IMPLEMENTATION

3.1. Introduction of DSP C6211

Texas Instruments (TI) TMS320C6000 generation [5] is based on VelociTI™ architecture, an advanced architecture

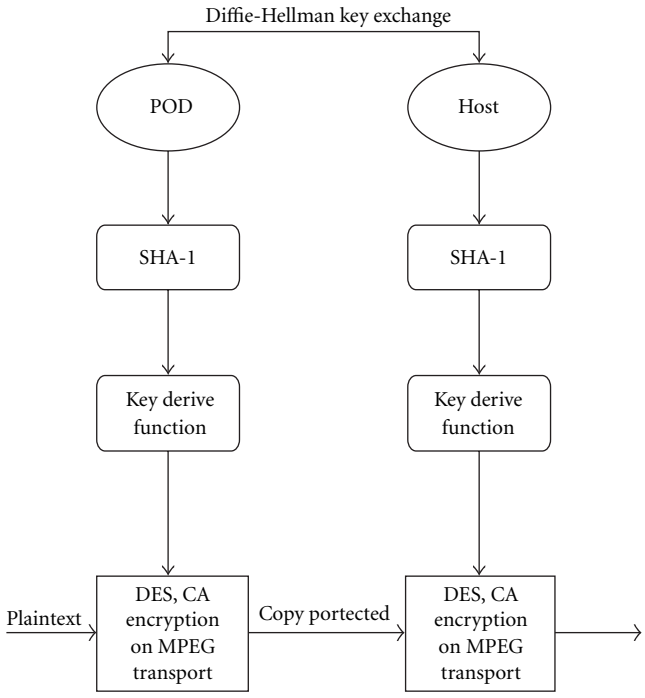


Figure 2: Cryptographic functions used in POD copy protection.

for DSPs with very long instruction word (VLIW). The VLIW architecture makes it very suitable for the multichannel and multifunction applications. TMS320C6211 (C6211 for short) provides 1200 MIPS (million instructions per second) at 150 MHz, and the TMS32062xx devices are the fixed-point DSP family. The cache architecture in C6211 provides low cost and high performance capabilities.

C6211 has 32 general purpose registers of 32 bit word length and eight highly independent functional units. The eight functional units provide six arithmetic logic units (ALUs) for a high degree of parallelism and two 16-bit multipliers. The development tools of C6211 include: C compiler, assembly optimizer to simplify programming and scheduling, and Windows™ debugger interface for visibility into source code execution [6]. The DSP based POD can greatly reduce the hardware design period, since it can easily reprogram when the specifications of POD are to be modified or new components are added.

3.2. Cryptography algorithms used in the DSP based POD

In order to make the POD more efficient, we use ECKDF which is based on the elliptic curve cryptography [7, 8] for the key derive function; and use cellular automata (CA) based symmetric-key cryptographic algorithm for media content protection.

ECDSA algorithm is applied in POD to authenticate the Host, which includes three parts: key schedule which is to set up the key, signature procedure, and verification process as illustrated in Figure 3.

Elliptic curve Diffie Hellman (ECDH) primitives is the basis for the operation of elliptic curve encryption scheme. For the POD, we use this algorithm to exchange the key between the POD and the Host. Figure 4 illustrates the flow chart of ECDH algorithm.

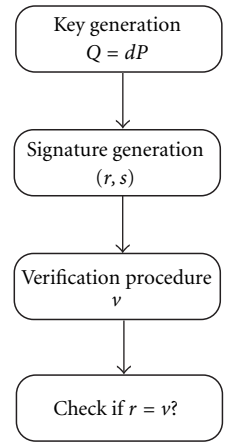


Figure 3: ECDSA algorithm.

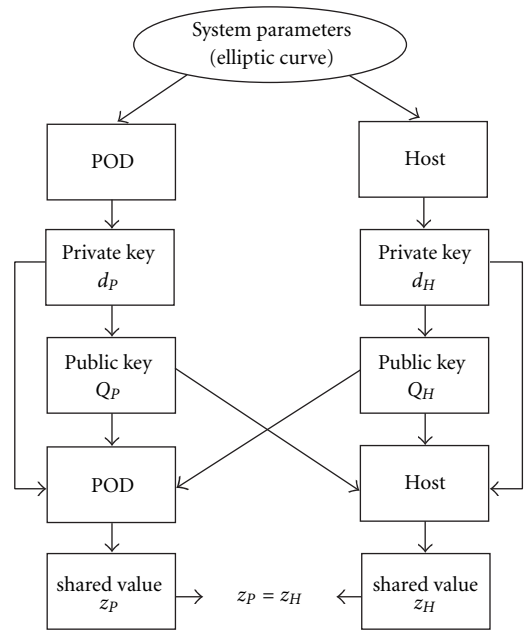


Figure 4: ECDH protocol between POD and Host.

Suppose POD (P) and Host (H) will communicate with each other, and require the key exchange. Here we use d_P and Q_P to represent the P's private key and public key which are obtained from the key schedule. d_H and Q_H denote H's private key and public key, respectively. P performs the following steps:

```
/* setup the scheme */
  create the elliptic curve;
/* compute the elliptic curve point */
    V_P = (x_p, y_p) = d_P Q_H ;
```

return the x component of V_P as the shared secret key (z_P).

Similarly, H uses the same primitive to get the shared secret key.

/* setup the scheme */
create the elliptic curve;
/* compute the elliptic curve point */
$V_H = (x_h, y_h) = d_H Q_P$;
return the x component of V_H as the shared secret key (z_H).

By running ECDH algorithm, we have $P_V = P_U$. That is, two parties get the same secret key.

In the POD implementation, ECKDF key derivation function is used to generate the common key for content encryption and decryption. The following is the description of ECKDF key derivation function:

check the length of input data (z);
initiate a Counter = 1;
for ($i = 0$; $i < n$; $i++$)
{
/* compute the hash value */
$k_i = h(z \mid \text{Counter})$;
increment Counter;
}
set the key $K = k_1 \mid k_2 \mid \cdots \mid k_n$,

where " | " means concatenation, and h stands for hash function SHA-1. By applying this function, we can generate different key sizes as required.

In the following, we introduce the cellular automata based symmetric-key cryptography algorithm and how it is applied in POD. Cellular automata (CA) is an array of cells where each cell is in any of the permissible states. For example, in a 2-state CA, each cell's state can be zero or one. In a k-neighborhood CA, at each clock cycle, the evolution of a cell value depends on its rule and the present states of its neighbors. The following three CA rules have special characteristics which can be applied in message encryption:

$$\begin{aligned} &\text{Rule 51: } x_i(t) = \overline{x_i(t-1)}, \\ &\text{Rule 195: } x_i(t) = \overline{x_{i-1}(t-1) \oplus x_i(t-1)}, \\ &\text{Rule 153: } x_i(t) = \overline{x_i(t-1) \oplus x_{i+1}(t-1)}. \end{aligned} \tag{1}$$

Theorem 1. *Applying complemented rules of 195, 153, and 51 to a CA forms a CA group [9].*

Theorem 2. *If a CA configures with rules 51, 153 and 195, then its state transition diagram consists of equal cycles of even length.*

Thus, if we choose rules of 51, 153, 195 as a group CA, then the fundamental transformations are self-inverse, that is, the decryption is carried out in the same way as encryption. Assuming the rule matrix is T, then we have

$$T^{2n} = T^n \cdot T^n = I \quad \text{(the identity matrix)}. \tag{2}$$

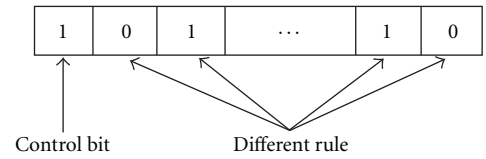


Figure 5: Overview of rule applied to message.

The CA-based block cipher scheme is as follows.

Encryption

$$E = T_1^{n_1} T_2^{n_2} \cdots T_q^{n_q}, \qquad C = EM. \tag{3}$$

Decryption

$$\begin{aligned} M &= E^{-1}C \\ &= (T_1^{n_1} T_2^{n_2} \cdots T_q^{n_q})^{-1} C \\ &= \left((T_q^{n_q})^{-1} \cdots (T_2^{n_2})^{-1} (T_1^{n_1})^{-1}\right) C \\ &= (T_q^{n_q} \cdots T_2^{n_2} T_1^{n_1}) C, \end{aligned} \tag{4}$$

where $T_1, T_2, \ldots, T_q$ are secret CA rules, which can be reviewed as the subkeys of the block cipher. The flexibility of CA based cryptosystem is that by choosing different values of n and q, we can achieve different security levels and data encryption/decryption rates according to the application requirements.

In Figure 5, the first bit is the rule control bit where "0" stands for rule 51, and "1" stands for rule 195 or 153 which will be selected by the corresponding bit. The core procedures of the CA algorithm is described as follows:

```
temp51 = (~Message) & (~Rule); /* Implement
the rule 51 */
switch(rule_sign)
{
    case 0: temp1 = Message ≫ 1;
        temp195 = (~(Message ^ temp1)) & Rule;
        temp_C_Block = temp195;
        break;
    case 1: temp2 = Message ≪ 1;
        temp153 = (~(Message ^ temp2)) & Rule;
        temp_C_Block = temp153;
        break;
}
      C_Block = temp51 | temp_C_Block.
```

Note that cycles used for encryption and decryption can be variable as well in CA based cryptography. For example, if we set $2n = 8$, that is, the message should be processed by applying 8 times of CA rule during the procedure of encryption and decryption, then we can choose the first four cycles for encryption and the other four cycles for decryption, or we can use the first three cycles for encryption and another five cycles for decryption.

Table 1: Test data for the encryption speed for CA (cycle < 5).

Test	*No_rule*	*No_cycle*	*No_clk*	*En_speed* (Mbps)
1	1	1	163	117.791
2	1	2	318	60.377
3	1	3	430	44.651
4	1	4	542	35.424
5	2	1	318	60.377
6	2	2	623	30.819
7	2	3	847	22.668
8	2	4	1071	17.927
9	3	1	430	44.651
10	3	2	887	21.646
11	3	3	1223	15.699
12	3	4	1559	12.316
13	4	1	542	35.424
14	4	2	1151	16.681
15	4	3	1599	12.008
16	4	4	2047	9.380
17	5	1	654	29.358
18	5	2	1415	13.569
19	5	3	1975	9.722
20	5	4	2535	7.574

Table 2: Test data for the encryption speed for CA (cycle ≥ 5).

Test	*No_rule*	*No_cycle*	*No_clk*	*En_speed* (Mbps)
1	1	5	654	29.358
2	1	6	766	25.065
3	1	7	878	21.868
4	2	5	1295	14.826
5	2	6	1519	12.640
6	2	7	1743	11.015
7	3	5	1895	10.132
8	3	6	2231	8.606
9	3	7	2567	7.480
10	4	5	2495	7.695
11	4	6	2943	6.524
12	4	7	3391	5.662
13	5	5	3095	6.204
14	5	6	3655	5.253
15	5	7	4215	4.555

The CA based cryptography is used for video content protection in the POD implementation. The experiment indicates that the encryption/decryption rate can be up to 75 Mbps, which satisfies the requirement of real-time data transmission in the cable network.

3.3. Implementation

The algorithms used in POD are programmed by C language and compiled with C6211 development tools, where the code composer studio compiles and converts the C programs into assembly language. Finally, an executable file in .out format is produced and loaded into the DSP.

Tables 1 and 2 list all the data tested from different rules and cycles. In these tables, *No_rule* stands for the number of rules, *No_cycle* means the number of process cycles for encryption, *No_clk* is the DSP cycles running this program on DSP, and *En_speed* is the speed of encryption which can be calculated by the following equation:

$$150 \times 128/\mathit{No\_clk}\,(\text{Mbps}). \tag{5}$$

4. CONCLUSION

POD is the security module to be used in the cable network and digital TV services. Its main function is to provide the cryptographic protocol in the interface between the POD and the Host, and to protect the content passing through the interface. In this paper, a DSP based POD implementation is proposed by using TMS320C6211. The experiment indicates that the proposed POD implementation provides high data speed and flexibility to real-time applications. In order to get the different degree of security and different speed of encryption/decryption, we use a simple symmetric key encryption algorithm—cellular automata cryptography for the content protection, whose encryption/decryption rate can be up to 75 Mbps.

REFERENCES

[1] OpenCable™, "OpenCable™ POD Copy Protection System," IS-POD-CP-INT01-000107, January 2000.

[2] OpenCable™, "OpenCable™ Host-POD Interface Specification," IS-POD-131-INT01-991027, October 1999.

[3] Hitachi, Intel, MEI, Sony and Toshiba companies, *Digital Transmission Content Protection Specification (Informational Version) Revision 1.0*, vol. 1, April 1999.

[4] National Institute of Standards and Technology(NIST), *Secure Hash Standard (SHS)*, FIPS Publication 180-1, April 1995.

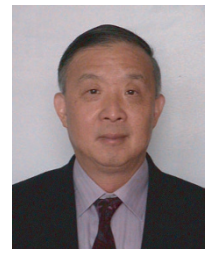
[5] Texas Instruments, "How to Begin Development Today with the TMS320C6211 DSP," Application report, SPRA474, September 1998.

[6] Texas Instruments, "TMS320C6000 Optimizing C Compiler—User's Guide," Digital signal processing solutions, 1999.

[7] Certicom research, "Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography," Working Draft, Version 0.5, Certicom Corp., 1999.

[8] M. Rosing, *Implementing Elliptic Curve Cryptography*, Manning Publications, Greenwich, Conn, USA, 1999.

[9] S. Nandi, B. K. Kar, and P. Pal Chaudhuri, "Theory and applications of cellular automata in cryptography," *IEEE Trans. on Computers*, vol. 43, no. 12, pp. 1346–1357, 1994.



Chang Nian Zhang received his B.S. degree in applied mathematics from University of Science Technology, China, and the Ph.D. degree in computer science and engineering from Southern Methodist University. In 1998, he joined Concordia University as a research assistant professor in Department of Computer Science. Since 1990, he has been with University of Regina, Canada, in Department of Computer Science. Currently he is a full professor and leads a research group in parallel processing, data security, and neural networks.

Hua Li received his B.E. and M.S. degrees from Beijing Polytechnic University and Peking University. He is a Ph.D. candidate in the Department of Computer Science, University of Regina. Currently, he works as an assistant professor at Department of Mathematics and Computer Science, University of Lethbridge, Canada. His research interests include parallel systems, reconfigurable computing, fault-tolerant, VLSI design, and information and network security. He is a member of IEEE.

Nuannuan Zhang was a graduate student in Department of Computer Science, University of Regina from September 1998 to September 2000.

Jiesheng Xie is a professor in the Department of Computer, China Agriculture University, Beijing. He was a visiting professor in the Department of Computer Science, University of Regina from September 1999 to August 2000.