

RESEARCH ARTICLE

An embedded DSP hardware encryption module for secure e-commerce transactions

J. Hu^{1*}, X. D. Hoang² and I. Khalil¹

¹ School of Computer Science and Information Technology, City Campus, RMIT University, GPO Box 2476V, Melbourne, 3001 Victoria, Australia

² Department of Computer Science, PTIT, Hanoi, Vietnam

ABSTRACT

Cryptography is one of the key elements in providing security for modern e-commerce systems. It is well known that software-based encryption has built-in security weaknesses due to storing and managing digital certificates/keys in a high-risk environment such as a local hard disk or software. This makes embedded hardware encryption a superior solution. However, most existing embedded hardware encryption modules need additional dedicated software in order to implement a secure e-commerce application, which increases cost as well as adds complexity. In this paper, a new embedded hardware DSP (digital signal processor) encryption module, using the RSA (Rivest, Shamir, and Adleman) algorithm, is developed for secure e-commerce transactions from the client side. The goal is to seamlessly integrate the embedded DSP hardware encryption module, which combines computational power and flexibility in programming, with a widely available web browser that provides the required e-commerce functions. The integrated system can store and process security sensitive data inside the plug-in hardware. The proposed scheme tries to maximize security strength while limiting overheads by utilizing a widely available web browser to perform e-commerce functions such as product searching, etc. A fully functional web e-commerce system has been developed as a proof of concept. Our major contribution is a design of a functional RSA plug-in encryptor which can store and encrypt sensitive information originated from the e-commerce process using standard web browsers. Implementation details addressing challenging issues such as big integer, large message, and communication components have been provided which have never been reported in the public literature. This can be very useful for real-life industry security applications. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS

e-commerce security; RSA encryption; embedded hardware; web browser

*Correspondence

J. Hu, School of Computer Science and Information Technology, City Campus, RMIT University, GPO Box 2476V, Melbourne, 3001 Victoria, Australia.

E-mail: jiankun@cs.rmit.edu.au

1. INTRODUCTION

Concern about security is a limiting factor in the growth of e-commerce. There are many important issues in e-security such as user authentication, access control, anti-virus protection, non-repudiation of origin, data confidentiality, data integrity, intrusion detection, *etc* [1–4]. Cryptographic techniques have long been in use to address the problems of confidentiality, data integrity, and non-repudiation [5]. It is well known that public-key cryptography can provide very strong cryptosystems [5–7]. In the family of public-key cryptosystems RSA (Rivest, Shamir, and Adleman) encryption is the most widely used scheme [5,8–11]. A brief introduction of the RSA algorithm is provided in

Section 3. Currently, most e-commerce security mechanisms using the RSA are built into software. This has generated the following security concern: although the RSA public key cryptosystem has avoided the problems of key distribution and management inherent in secret key cryptosystems, the trustworthy Certificate Authorities (CA) need to assure that a party is indeed in possession of the private key corresponding to the public key included in the certificate. Protection of the private key remains a problem as it must remain secure during its lifetime and beyond [1].

Any software, which requires access to the private key, must protect that key. In most cases existing technology fails to provide strong assurance for the protection of private keys in the PC environment [1]. A hardware cryptosystem

is a good solution to these problems because it can provide high computing power as well as enhanced security features. In designing enhanced security, removable smart cards are the best solution presently available [1]. Cryptographic Hardware for the RSA cryptosystem is an on-going research topic [2,5–7]. Research and development on hardware RSA encryption can be roughly classified into three categories:

1. *Category 1: Dedicated VLSI (Very Large Scale Integrated) Systems* The goal is hardware design of an efficient RSA cryptosystem architecture. In Ref. [9] a VLSI design for the implementation of RSA and IDEA encryption engine is presented. In Ref. [12], PAM (Programmable Active Memory) technology is used to speed up long integer multiplication and hence speed up the RSA encryption/decryption. There many other reports along this line [8,10,13–15], to name just a few.
2. *Category 2: Hardware Cryptosystem Tamper-Resistance* A hardware cryptosystem is regarded as the best option for the maximum security [1]. However, there still exist many attacks if an adversary has physical access to a tamper-proof device. A description of one powerful attack, called fault-based cryptanalysis, was released by Bellcore [16]. Our paper does not intend to address this problem. Interested readers are referred to Refs. [7,10,12] and the references therein.
3. *Category 3: Embedded Hardware Cryptosystems* Though dedicated VLSI hardware can provide very high computing power, it involves high investment cost and lacks flexibility. Embedded hardware, such as smart cards, and especially DSP (digital signal processors) is an ideal solution [1]. A DSP can provide more computing power than that of normal microprocessors and higher flexibility than that of VLSI chips. It also has a low cost due to mass production. Another significant advantage is that nearly every PC has a PCI interface and communication ports that support DSP boards. Implementation of the RSA cryptosystem on a DSP chip is non-trivial. The RSA cryptosystem is based on the theory of large prime number factorization and thus requires very intensive modulo computation as well as storage-intensive big number processing, which is very difficult to accommodate on a single DSP chip with limited computing capacity and storage space. In the literature very few reports of RSA implementations on DSP hardware have been found [17,18]. In Ref. [17] a design and implementation of RSA cryptosystem using multiple DSP chips is presented. The system allows for additional DSP chips to be inserted in allocated slots to improve performance. In this system, the DSP application can be controlled by a PC application using UART serial channels. However, no information is provided about the communication mechanism or the effect of the communications on the overall performance of the system. In Ref. [18] a fast RSA encryption module is developed through exploiting the computational characteristics of

Motorola 56300 DSP family. Currently most hardware RSA cryptosystems have been developed either as an isolated encryption module that is not easily integrated into existing e-commerce systems to provide secure transactions, or as a complicated and expensive system that requires additional dedicated software packages to perform e-commerce functions. On the other hand web browsers have played a dominant role in B2C (business to customer) e-commerce services. Obviously, the integration of existing e-commerce resources, such as web browsers, with the embedded hardware encryption module is of significant interest. Comparing with conventional e-commerce systems deploying standalone and dedicated software, the proposed integrated systems would have the benefits of a higher security level by using embedded RSA encryption hardware as well as limited overhead cost and reduced complexity by using existing web browser software. In this paper, based on our previous work [19–21], a novel DSP RSA hardware encryption module is developed using a general floating-point Texas Instruments' TMS320C6711 DSK board and software development tools. The hardware RSA encryption module is integrated into the client side of a functional e-commerce system to enhance its security performance. The use of multi-DSP hardware encryption/decryption modules on the server side of the web system is also an option. This topic is not discussed here and left for future work. Experimental results have shown that the DSP hardware RSA encryption module could be used efficiently to improve security for e-commerce transactions. Although commercial plug-in encryption hardware products exist, for example, Gili USB Stick Encryption [22], most of them are symmetric cryptosystem based. Our major contribution is a design of a functional RSA plug-in encryptor which can store and encrypt sensitive information originated from the e-commerce process using standard web browsers. Implementation details addressing challenging issues such as big integer, large message, and communication components have been provided which have never been reported in the public literature. This can be very useful for real-life industry security applications. The choice of DSP hardware TMS320C6711 is simply for the convenience of illustrating the concepts due to our existing relevant expertise, majority technical issues (if not all of them) such as system architecture, big integer implementation, message block size calculation, etc. are common or similar in other DSP hardware. Therefore technical solutions presented in this paper are widely applicable.

The remaining of the paper is organized as follows: Section 2 presents the development of the DSP hardware encryption model including the process flow and the communication mechanism between the client web browser and the DSP hardware encryption engine module. Section 3 discusses the implementation of the system in detail and Section 4 shows the experimental results with performance

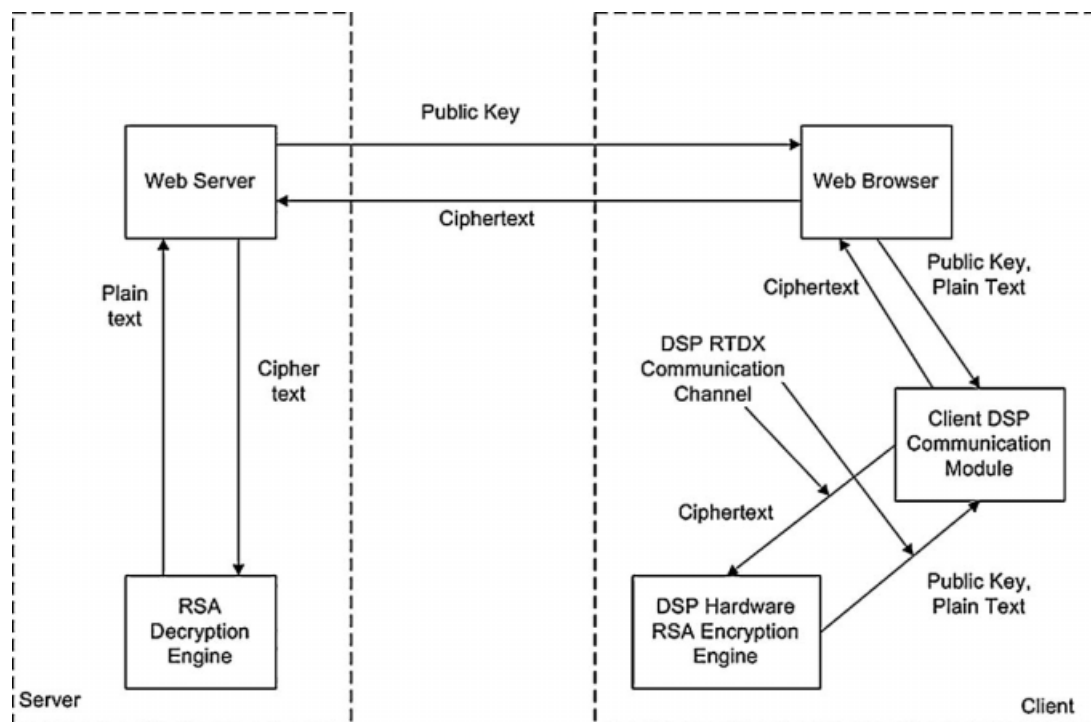


Figure 1. Proposed encryption model using DSP hardware for secure e-commerce systems.

analysis. Section 5 is devoted to conclusions. In this paper, the details of RSA encryption algorithm, RSA key generation procedures and the DSP hardware structure are not discussed. Interested readers would find them in the Refs. [3,19,20,23,24] in this paper.

2. ARCHITECTURE OF THE NEW EMBEDDED DSP HARDWARE ENCRYPTION MODEL FOR WEB-BASED E-COMMERCE SYSTEMS

2.1. The new embedded encryption model

For secure e-commerce transactions an implementation of the RSA public key encryption algorithm embedded in DSP hardware is proposed as shown in Figure 1.

The system is based on the client/server model and consists of two major parts: the server and the client. The server consists of a web server, a database server, and an embedded RSA software decryption engine. The client includes a web browser, a client DSP communication module, and a DSP hardware RSA encryption engine. The server and the client communicate with each other using TCP/IP protocols in the Internet/Intranet environments. On the server side, the RSA decryption engine is only activated when there are decryption requests from the web server. The client DSP communication module can be downloaded and installed

from server to client. It can also run embedded inside the client web browser. The system works as follows:

- (1) The client web browser requests a web page that contains the client DSP communication module as an embedded script. The server sends the requested page to the client. The client DSP communication module is then installed and activated. The server can also send its public key to the client in the requested web page if the client does not obtain it from another source such as a Certificate Authority [1]. How to effectively deliver and install the product is not the concern of this paper. The user can purchase the product directly from the software retailer and install it accordingly.
- (2) When the user enters data (plain text) into a data entry form in a web page and submits the information, the client DSP communication module captures the plain text, it then establishes an input communication channel to the DSP hardware encryption engine, and sends the public key and the plain text to the encryption engine.
- (3) Upon receiving the public key (or obtaining it by other methods and storing it in the DSP chip) and the plain text, the DSP hardware RSA encryption engine encrypts the plain text, and then sends the cipher text back to the client DSP communication module through an output communication channel.

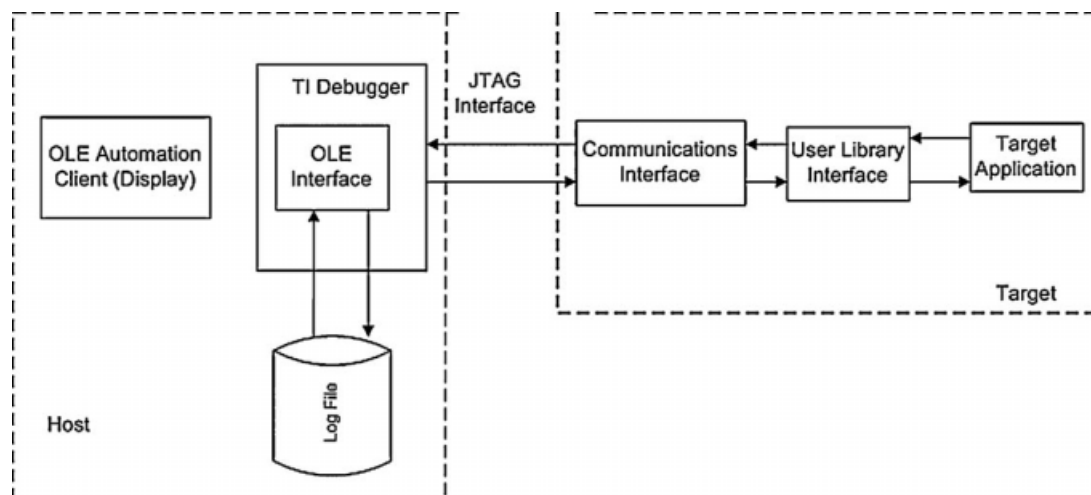


Figure 2. RTDX communications between host and target [23–25].

- (4) Next, the client DSP communication module forwards the cipher text to the browser, and the browser submits the encrypted information to the server.
- (5) The server receives the cipher text and passes it to the RSA decryption code which will decrypt the cipher text using the server's private key and return the plain text to the server.

2.2. Communication mechanism between the client and the DSP hardware application

The client DSP communication module embedded in the web browser is responsible for all communications between the client (the web browser) and the DSP hardware RSA encryption engine. Since the communication module is an embedded component of the browser, it exchanges the information with the browser via the browser's object programming interface. Communications with the hardware encryption engine proceed using DSP Real-Time Data Exchange (RTDX) channels, which come as a software library together with the DSP application development tools. The RTDX communication channels can support real-time and full duplex data exchanges. The RTDX communication scheme is shown in Figure 2 [23–25].

RTDX communication channels can be operated in two modes: non-continuous and continuous mode. In non-continuous mode, the transferred data are recorded into a log file that has been specified in RTDX configuration. Non-continuous mode is used when the programmers want to capture a finite amount of data and save it into a log file. In continuous mode, data are logged into a circular memory buffer in the RTDX Host Library. This mode is used when users want to continuously obtain and display the data from a target application, with no requirement for storage of the data in a log file [23]. For the purpose of this project, we use RTDX in continuous mode to process the data in real

time, while avoiding the storage of plain text data on the hard disk.

3. SYSTEM IMPLEMENTATION

3.1. Model implementation

A functional prototype system has been built for the proof of concepts. The hardware and software setup are provided in the Appendix A. Following are the illustration of our solutions to technical challenges for building such an application system.

3.1.1. DSP hardware RSA encryption engine.

The DSP hardware RSA encryption engine consists of two parts: the implementation of RSA encryption algorithm to encrypt the messages and the RTDX communication library to communicate with the web browser through the client DSP communication module. The RSA public key encryption algorithm was invented by R. Rivest, A. Shamir, and L. Adleman. It can be summarized as following [3,24].

RSA Public Key Encryption Algorithm

1. Public Key: N product of two large prime numbers, p and q (p and q must be kept in secret) E relatively prime to $(p-1)(q-1)$
2. Private Key: $De^{-1} \bmod (p-1)(q-1)$
3. Encryption: $C = m^E \bmod (n)$
4. Decryption: $M = C^d \bmod (n)$

3.1.1.1. Big integer representation for RSA implementation On most modern 32-bit computer architectures, integers are defined as a maximum of 32 bits, and optionally the size can be reduced to 16 bits. The maximum values representable by a 16-bit and 32-bit unsigned

Table I. BigInteger data structure.

```

Data Structure of big integer:
typedef struct {
    unsigned short *digits; // pointer to array of digits
                                // the least significant
                                // digit at index 0
    unsigned int size; // size of the big integer
} BigInteger;

```

integers are $2^{16} - 1 = 65\,535$ and $2^{32} - 1 = 4\,294\,967\,295$, respectively. However, the RSA algorithm requires keys on the order of 10^{100} or larger, which are clearly larger than the standard integer types. The solution is to use an array of fixed integers to represent the big integers or multiple precision integers. This allows the accommodation of any integer values, and is limited only by the size of the physical memory. In Ref. [25], a fast radix-4 modular multiplication hardware algorithm is reported. Numbers are stored in a redundant representation and additions/subtractions are performed without carry propagation. The algorithm was found efficient, especially in applications where modular multiplications are carried out iteratively. However, the described algorithm is suitable for specifically designed VLSI chips, not a DSP with a general architecture. In this project, the base for multiple precision integer representation is selected as 65 536 or 2^{16} . The major advantage of representing the integer in base 65 536 over base 10 is that it requires far less memory. This is very important, especially in the embedded systems that typically have limited memory. Furthermore, with base 65 536 (2^{16}), fast shift operations can be used to replace high-cost multiplication and division operations. This, in turn gives better computational performance. A larger base than 65 536 will make implementation of the normal arithmetic operations more complicated because on most currently used computing systems, the largest integer supported is 32-bit length. With 32-bit integers the built-in machine level arithmetic functions cannot be used directly as the results would overflow the 32-bit registers. For example, if base 2^{24} is used, it is much complicated to compute $2^{23} \times (2^{24} - 1)$ as the result is greater than 2^{32} . The computation of $(2^{16} - 1) \times (2^{16} - 1)$ is much simpler since the result is less than 2^{32} and the built-in arithmetic functions can be used directly.

Table I shows the C++ definition of the BigInteger data structure. Each digit in the BigInteger type can store a 16-bit integer number ranging from 0 to 65 535. The

maximum size of the big integer is $2^{32} = 4\,294\,967\,296$ digits, hence, the biggest integer value can be represented is $65\,536^{4\,294\,967\,296}$. This integer value is sufficient to represent very large key lengths such as 2048-bit key length (about 616 base-10 digits).

3.1.1.2. BigInteger package implementation

The BigInteger package has been implemented with all the big integer arithmetic functions to support RSA encryption/decryption algorithm. The implementation of arithmetic functions is based on efficient arithmetic implementations in Ref. [3, chapter 16]. Other functions such as input and output are also implemented to convert string messages to BigInteger and vice versa.

3.1.1.3. Encryption of a large message

In the RSA algorithm, the equation $C = m^e \text{ mod } (n)$ is used to encrypt message m using the public key (n, e) . In order to make RSA encryption work properly, the message m must be represented as an integer and value of (m) must be less than value of (n) [10]. Therefore, for large message, the first step is to decide on a message block size based on the size of public key n . Next, the message is divided into blocks and converted to the appropriate big integers. The final step is to use the RSA algorithm to encrypt the blocks and compose a cipher text output. The size of the block could be a fixed or changeable value depending on the system's design. Theoretically, the block size could be any value provided that value of (m_i) must be less than value of (n) . However, the block size should not be too small as a block cipher with a small block size may be vulnerable to the attacks based on statistical analysis [24]. One such attack involves simple frequency analysis of cipher text blocks. Therefore, to maximize security, the block size must be as large as possible [24, p. 227] and roughly the same size as public key n . On the other hand a large block size increases the complexity of the implementation. In this project, the selected message block size in characters is equal to the bit-size of public key divided by 2.5 (taking rounded off integer value). Table II shows the details of the block size calculation from the key size.

3.1.1.4. RTDX communication library implementation

The Texas Instruments' RTDX library has been included in the DSP hardware RSA encryption engine to support real-time communication between the DSP and the client DSP communication module. The built-in RTDX library provides functions to initialize the RTDX communications, open communication channels, read/write data to

Table II. Calculation of block size based on key length.

-
1. Input the public key n expressed in decimal value
 2. Calculate the bit-size k from decimal value of n , i.e., finding the smallest integer k such that $n \leq (10^k - 1)$. For convenience, the value 1 will be ignored throughout this process as numbers under discussions are very big
 3. The block size t of the message m_i can be calculated via
Value of $(m_i) < 2^{t \cdot 8} < 10^k$ or $k/t > 8/(\log_2(10)) \approx 2.4082$. To make sure that the value of $(m_i) < \text{value of } (n)$ in all cases, we choose the 2.5 as the ratio, or block size in chars = public key size in terms of base-10 digits divided by 2.5 (taking rounded off value)
 4. Output block size t
-

the channels and close the channels. In order to make the RTDX communication channels work properly, they need to be configured with the correct mode and a suitable data buffer size.

3.2. Client DSP communication module

As discussed above, because the client DSP communication module plays the role of a communication bridge, it must be able to communicate with both the web browser and the DSP hardware RSA encryption engine. The module exchanges information with the DSP application using RTDX channels while it communicates with the browser via the OLE (Object Linking and Embedding) interface.

The implementation of the communication module consists of two groups of functions: the browser interface functions and the DSP interface functions. The browser interface functions are used to exchange data with the web browser and are callable from the browser using scripting languages such as JavaScript. The DSP interface functions are implemented based on RTDX exported library functions that provide communication channels to and from the DSP application. In order to maximize the performance of the RTDX communication channels, the number of read/write operations to the channels in a session should be minimized [23,26]. In our implementation, we use the read/write operations in batch mode to improve the performance of the RTDX channels. In order to use the client communication module, it is embedded in a web page using an HTML <OBJECT> tag. The module is designed so that it can be downloaded and installed automatically on the client machine.

3.3. Server RSA decryption engine

The decryption engine was developed using Microsoft Visual C++ and linked as a *Dynamic Link Library* (DLL) to run on the server. It supplies all of the necessary functions of an RSA decryption library. The decryption engine was installed on the server and integrated into the web server. The engine is only active when there is a request for message decryption from the web server. The implementation of RSA algorithm on the server component is also based on efficient arithmetic implementations in Ref. [[24], chapter 16]. The decryption engine also provides interface functions to make it callable from server-side scripts such as Microsoft VBScript and JScript.

4. EXPERIMENTAL RESULTS

Several experiments were conducted on the client/server systems using hardware and software described above with the various RSA key lengths and message sizes. Encryption performance was tested with RSA key pairs ranging from 128 bits to 2048 bits and message sizes of 512 bytes, 1024

Table III. Overall DSP encryption speed on client.

Message (bytes)	Speed (K bits/s) based on key length						
	128	256	512	768	1024	1536	2048
512	2.959	1.214	0.389	0.184	0.097	0.048	0.020
1024	3.108	1.260	0.414	0.185	0.108	0.049	0.024
2048	3.170	1.288	0.415	0.193	0.115	0.049	0.027

Table IV. Server software decryption speed.

Message (bytes)	Speed (K bits/s) based on key length						
	128	256	512	768	1024	1536	2048
512	11.770	5.251	1.736	0.839	0.447	0.216	0.094
1024	13.386	5.390	1.842	0.832	0.486	0.218	0.120
2048	14.027	5.491	1.847	0.866	0.508	0.214	0.125

bytes, and 2048 bytes. The performance statistics of the model are shown in Tables III and IV. Table III shows the overall encryption speed, including communication overheads between the client and the DSP, and the encryption speed within the DSP. Table IV presents the server decryption performance.

4.1. DISCUSSION

The system's performance is fast enough for small to medium size messages, especially with RSA key lengths up to 1024 bits. In our experiments the encryption time of credit card information is less than 0.5 and less than 1.3 s with the key lengths of 256 and 512 bits respectively, which is very satisfactory from point of user's experience. It should be noted that speed performance is not the primary consideration of this project. Therefore, an external plug-in DSP board is chosen to maximize the convenience of the client and the system security as the keys stored on this board are much more securely protected than keys stored in software on the PC or on the PC hard disk [1]. Plugging a DSP card into the PC's internal PCI slot could reduce overheads in transmission and improve the overall speed of the system; however, it is also inconvenient since it requires opening the PC case and represents a potential security risk since the keys are now stored in a software-accessible location within the PC. Experiments have shown that the TI optimizing compiler used in our project generates code that is poorly parallelized, which leaves scope for significant performance improvement. Experience indicates that assembly coding of critical sections could also provide significant improvement of the overall system speed performance.

5. CONCLUSIONS

This paper has proposed a method of integrating a DSP-based hardware RSA encryption module with existing web systems for e-commerce applications. The architecture can

maintain positive features of existing e-commerce systems while improving security with the use of RSA hardware encryption in a way that limits the overheads involved. All sensitive data as well as the processing process of these data are placed inside the external plug-in DSP RSA encryptor, which has avoided the security vulnerable PC environment. A successful efficient implementation of RSA DSP hardware encryption module using TI TMS320C6711 DSP board is presented. The paper also provides details of efficient communication between a web browser and this RSA hardware module. A fully functional e-commerce system including the web and database server was built to test the concept proposed in this paper. The testbed built in this project is used for the purpose of a proof of concept; hence the speed performance is not the primary consideration. Further speed improvement can be obtained either by assembly coding of critical sections or by the use of an improved optimizing compiler. It should be noted that the external plug-in RSA encryptor can be protected by using password or PIN. For stronger protection, the emerging biometric security mechanism can be adopted [27–30].

ACKNOWLEDGEMENTS

The authors acknowledge the financial support from the ARC (Australia Research Council) Discovery Grant with Project ID DP0985838, and National Foundation for Science and Technology Development (NAFOSTED) of Vietnam.

APPENDIX A—Hardware and Software Setup

Hardware

The client machine is a PC with 700 MHz Pentium-III CPU, and 128 MB SDRAM running Microsoft Windows Professional 2000. A Texas Instruments' TMS320C6711 150 MHz DSP Starter Kit (DSK) [23] is connected to the client computer via a parallel port.

The server used in experiments is a PC with 1000 MHz Pentium-III CPU, and 512 MB SDRAM running Microsoft Windows 2000 Server with Microsoft SQL server.

Software Development Tools

The DSP C6711 development Tools: The Texas Instrument Code Composer Studio 2.0 and the attached RTDX libraries were used to develop the DSP hardware RSA encryption engine.

Development Tools for client and server applications: Microsoft Visual C++ 6.0 was used to develop the server's decryption engine and the client DSP communication module that is used to communicate with the DSP encryption engine.

Web browser: The web browser used in the experiments was Microsoft Internet Explorer. It was chosen because it supports the ActiveX/OLE (Object Linking and Embedding) technology and is one of the most commonly used web browsers.

Web and database server: The web server software used was Microsoft Internet Information Services 5.0. A functional e-commerce website was built on the web server with Microsoft SQL Server 7.0 as the backend database. The e-commerce website provides customers with online registration for various types of commercial events. The customers can register their details with the system, select the events in which they are interested, and then make an online payment. In this paper, the design and the implementation of the web and database applications are not discussed due space limitations.

REFERENCES

1. Gollmann D. E-commerce security. *Computing & Control Engineering Journal* 2000; **11**(3): 115–118.
2. Oppliger R. Shaping the research agenda for security in E-commerce, *IEEE 10th International Workshop on Database & Expert Systems Applications*, September 1999, p. 810, Florence, Italy.
3. Hoang XD, Hu J, Bertok P. A program based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference. *Journal of Network and Computer Applications* 2009; **32**: 1219–1228.
4. Hu J, Qiu D, Chen HH, Yu X. A simple and efficient data processing scheme for HMM based anomaly intrusion detection. *Special Issue of Advances on Network Intrusion Detection. IEEE Network* 2009; **23**(1): 42–47.
5. Hu J, Bertok P, Tari Z. Taxonomy and framework for integrating dependability and security. Chapter 6 Part II: Modelling the Interaction between Dependability and Security. In *Information Assurance: Dependability and Security in Networked Systems*, Qian Y, Joshi J, Tipper D, Kirshanamurthy P (eds). Elsevier, The Netherlands, 2008; 149–170. ISBN: 978-0-12-373566-9.
6. Parno B. The trusted platform module (TPM) and sealed storage. Technical Reports, June 2007, RSA Laboratories. www.rsa.com/rsalabs/node.asp?id=2002. Retrieved on September 21, 2009.
7. Miyamoto A, Homma N, Aoki T, Satoh A. Enhanced power analysis attack using chosen message against RSA hardware implementations. *IEEE International Symposium on Circuits and Systems* 3282–3285.
8. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 1978; **21**(2): 120–126.
9. Royo A, Moran J, Lopez JC. Design and implementation of a coprocessor for cryptography applications. *IEEE*

- Proceedings of European Design and Test Conference*, March 1997, pp. 213–217.
10. Buldas A, Pöldre J. A VLSI implementation of RSA and IDEA encryption engine. *Proceedings of 15th NORCHIP conference*, Tallinn 1997, pp. 281–288.
 11. Schneier B. *Applied Cryptography, Protocols, Algorithms, and Source Code* in C. John Wiley & Sons, Inc., USA, 1996.
 12. Izumi M, Sakiyama K, Ohta K. A new approach for implementing the MPL method toward higher SPA resistance. *International Conference on Availability, Reliability and Security (ARES'09)*, March 16–19, 2009, pp. 181–186.
 13. Shand M, Bertin P, Vuillemin J. Hardware speedups in long integer multiplication. *ACM Symposium on Parallel Algorithms and Architectures* 1990; **1**: 138–145.
 14. Kim CH, Oh S, Lim L. A new hardware architecture for operations in GF (2n). *Transactions IEEE Computers* 2002; **51**(1): 90–92.
 15. Takagi N, Yajma S. Modular multiplication hardware algorithm with a redundant representation and their application to RSA cryptosystem. *IEEE Transactions on Computers* 1992; **41**(7): 887–891.
 16. Bellcore Press Release. New Threat Model Breaks Crypto Codes. September 1996, www.bellcore.com/PRESS/ADVSR96/facts.html.
 17. Yen SM, Kim S, Lim S, Moon SJ. RSA speedup with Chinese remainder theorem immune against hardware fault cryptanalysis. *IEEE Transactions on Computers* 2003; **52**(4): 461–472.
 18. Er MH, Wong DJ, Sethu A, Ngeow KS. Design and implementation of RSA cryptosystem using multiple DSP chips. *IEEE International Symposium on Circuits and Systems* 1991; **1**: 49–52.
 19. Hu J, Xi Z, Jennings A, Lee HYJ, Wahyudi D. DSP application in e-commerce security. *Processing of IEEE International Conference on Acoustics, Speech, and Signal*, 2, 2001, pp. 1005–1008.
 20. Hu J, Hoang XD, Low N. Selection and implementation of a data encryption model for E-commerce secure transactions. *TU02–Data Protection and Security I, IEEE International Conference on Telecommunications*, Romania, Bucharest, June 2001.
 21. Hoang XD. E-Commerce Security Enhancement and Anomaly Intrusion Detection using Machine Learning Techniques. PhD thesis, 2006, RMIT.
 22. Gili USB Stick Encryption. www.freedomdownloadcenter.com/Utilities/File_Encryption_Uilities/Gili_USB_Stick_Encryption.html. Retrieved on September 21, 2009.
 23. Texas Instruments' Online document: "TMS320C6711–Floating point DSP," URL: focus.ti.com/docs/prod/folders/print/tms320c6711d.html. Retrieved on the September 29, 2009.
 24. Menezes AJ, Oorschot PC, Vanstone SA. *Handbook of Applied Cryptography*. CRC Press, USA, 1997.
 25. Takagi N. A radix-4 modular multiplication hardware algorithm for modular exponentiation. *IEEE Transactions on Computers* 1992; **41**(8): 949–956.
 26. Keil D. Real-Time Data exchange. Texas Instruments White Paper, Digital Signal Processing Solutions, February 1998.
 27. Han F, Hu J, Yu X, Feng Y, Zhou J. A novel hybrid crypto-biometric authentication scheme for ATM based banking applications. *IAPR International Conference on Biometrics (ICB2006)*, 5–7 January, 2006, Hong Kong China. Published at Lecture Notes in Computer Science, Springer, 2005, 3832, pp. 675–681.
 28. Han F, Hu J, Yu X. A biometric encryption approach incorporating fingerprint indexing in key generation. *International Conference on Intelligence Computing (ICIC06), Kunming, China, 2006*. Published at *Computational Intelligence and Bioinformatics, Lecture Notes in Computer Science, Springer*, 0302–9743, 4115, 2006, pp. 342–251.
 29. Wang Y, Hu J, Philip D. A fingerprint orientation model based on 2D Fourier Expansion (FOMFE) and its application to singular-point detection and fingerprint Indexing. *Special Issue on Biometrics: Progress and Directions, IEEE Transactions on Pattern Analysis and Machine Intelligence*, April 2007.
 30. Hu J. Mobile fingerprint template protection: progress and open issues. *IEEE ICIEA Conference*, Singapore, 3–5 June, 2008.