International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 4, Issue 1, January-February 2015 **ISSN 2278-6856**

Software Co-Design of Pell's Equation with Jacobi Symbol for SSL Protocol Execution

[1]M Kondala Rao, [2]P S Avadhani, [3]D Lalitha Bhaskari

[123]Department of Computer Science and Systems Engineering, Andhra University, India

Abstract

Modern mobile devices like cell phones or PDAs allow for a level of network connectivity similar to that of standard PCs, making access to the Internet possible from anywhere at any time. Going along with this evolution is an increasing demand for cryptographically secure network connections with such resource-restricted devices. The Secure Sockets Layer (SSL) protocol is the current de-facto standard for secure communication over an insecure network like the Internet and provides protection against eavesdropping, message forgery and replay attacks. To achieve this, the SSL protocol employs a Pell's equation with Jacobi Symbol, which can result in unacceptably long delays on devices with modest processing capabilities. In this paper we introduce a hardware/software co-design approach for accelerating SSL protocol execution in resource-restricted devices. The software part of our co-design consists of MatrixSSL TM, a lightweight SSL implementation into speed up the public-key Pells Equation with Jacobi Symbol operations performed during the SSL handshake.

Key Words:- Pells Equation, Jacobi Symbol, Legendre Symbol, SSL Protocol

1 Introduction

The current de-facto standard for secure communication over an insecure, open medium like the Internet is the Secure Sockets Layer (SSL) protocol [9] and its successor, the Transport Layer Security (TLS) protocol [8, 33]. Both use a Pell's Equation of public-key and secret-key cryptographic techniques to ensure confidentiality, integrity and authenticity of communication between two parties (typically referred to as client and server). The SSL protocol is composed of two layers and includes several sub-protocols. At the lower level is the SSL Record Protocol, which specifies the format used to transmit data between client and server (including encryption and integrity checking) [9]. It encapsulates various higher-level protocols, one of which is the SSL Handshake Protocol. The main tasks of the handshake protocol are the negotiation of a set of cryptographic algorithms, the authentication of the server (and, optionally, of the client), as well as the establishment of a pre-master secret via asymmetric (i.e. public-key) techniques [9]. Both the client and the server derive a master secret from this pre-master secret, which is then used by the record protocol to generate shared keys for symmetric encryption and message authentication [9]. The handshake protocol, on the other hand, relies on services provided by the record protocol to exchange messages between client and server.

The SSL/TLS protocol is supports Pells Equation for the cryptographic operation, and allows the communicating parties to make a choice among them [9]. At the beginning of the handshake phase, the client and the server negotiate a cipher suite, which is a well-defined Pells Equation Public Key Cryptography algorithms for authentication, key agreement, symmetric encryption, and integrity checking. Both SSL and TLS specify the use of Pells Equation for authentication, and key exchange. In addition, certain cryptographic operations (e.g. generation of signatures, key exchange) can be executed Pells equation is much faster than RSA in a multiplicative group like Z^*p. The advent of the wireless Internet has created a strong demand for secure communication via mobile devices such as cell phones or PDAs. However, these devices are battery-operated, and hence severely constrained by computational resources (processing power, memory, network bandwidth, etc.). When implementing SSL for mobile devices, great care must be taken to utilize the scarce resources as efficiently as possible [2, 3, 14]. The delay a user experiences when establishing an SSL connection depends heavily on the execution time of the public-key operations carried out during the handshake (i.e. authentication and key agreement). If an Pell's based cipher suite is used, the client has to perform two modular exponentiations, one for the verification of the server's certificate and one for the encryption of the pre-master secret. Even though these exponentiations involve public exponents (which are usually small), they constitute a significant overhead. The hardware acceleration of the public-key operations carried out during the handshake is desirable. The straightforward approach to hardware acceleration of public-key cryptography is the integration of a dedicated co-processor to off-load the computationally expensive parts of an algorithm (e.g. modular exponentiation in the case of Pells) from the main processor [7, 17]. In the embedded realm, however, fixed-function hardware accelerators in the form of cryptographic co-processors exhibit a number of disadvantages. Co-processors for Pell's Equation with Jacobi Symbol generally demand large silicon area, which poses a particular problem for low-cost embedded devices. Given the algorithm-agile nature of the SSL protocol, it seems questionable whether a cryptographic co-processor can meet the desired level of flexibility at moderate hardware cost. Modern security protocols, such as SSL or IPSec, are constantly evolving

and hence changing their repertoire of crypto algorithms (e.g. to phase out compromised algorithms, to include new algorithms, or to adapt the minimal key size of algorithms), which again calls for a flexible and scalable approach to hardware acceleration. In this paper we present a new methodology for hardware acceleration of the SSL handshake based on hardware/software co-design [35] of the involved cryptographic Pell's Equation with Jacobi Symbol algorithms. The specific co-design approach we followed in our work is the integration of custom instructions into a general-purpose processor to speed up the processing of performance-critical arithmetic operations carried out in Pells equation (e.g. multiplication in finite fields). Hardware/software co-design at the granularity of instruction set extensions is particularly area-efficient and allows one to retain the full flexibility of a "pure" software solution, which makes this approach perfectly well suited for hardware acceleration of the SSL protocol in low-cost embedded systems. Our experimental results show that, due to the lightweight implementation of the SSL stack, the speed-up gained at the low-level field arithmetic propagates almost lossless up to the application layer. We also compare the results of our co-design with the performance figures of a pure software implementation of the SSL protocol, namely Open SSL [28]. This comparison confirms that hardware/software co-design in the form of instruction set extensions for public-key cryptography, in particular pells equation, is a good way to accelerate the SSL handshake.

2 Public-Key Cryptography

The SSL/TLS protocol makes heavy use of public-key cryptography during the handshake phase to accomplish such tasks as authentication and key establishment. In this section we briey discuss implementation aspects of both classical public-key cryptosystems (Pell's Equation with Jacobi Symbol) as well as elliptic curve cryptosystems in the context of the SSL handshake.

2.1 RSA, DSA, Difie-Hellman

The RSA cryptosystem operates in the residue class ring Zn, where n is the product of two large primes. DSA and Difie-Hellman, on the other hand, use the multiplicative group Z* p (or a subgroup thereof) as underlying algebraic structure. The basic operation of all these cryptosystems is exponentiation, i.e. the repeated application of the ring or group operation, namely multiplication, to an element of the ring (resp. group). Of course, the multiplications are performed modulo n (or modulo p, respectively), which means that said exponentiation is actually a modular exponentiation of the form $c = m^e \bmod n$ [24]. In case of the RSA algorithm, the modulus n is a product of primes, the exponent e satisfies gcd (e; Ø(n)) = 1, and the base m is in the interval [0; n - 1], i.e. m Zn. The security of the RSA cryptosystem is closely related to the Integer Factorization Problem (IFP), even though no mathematical proof exists that the factorization of n is needed to break RSA. Factoring an RSA modulus is widely believed to be computationally infeasible if its prime factors are large (e.g. > 512 bits). On the other hand, the security of DSA and Difie-Hellman relies on the Discrete Logarithm Problem (DLP) in Z*p, which is defined as follows: Given a generator g for Z* p (or a subgroup thereof) and an element a of said (sub)group, find the integer x such that a = gx mod p. The DLP is considered intractable, provided that the group Z* p and the generator g are properly chosen. The standard algorithm for computing a modular exponentiation m^e mod n is the square-and-multiply algorithm, which is also referred to as binary exponentiation method [24] since it uses the binary expansion of the exponent e. Two variants of the binary method are described in [24]; one scans the bits of e from left to right (i.e. MSB first), the other from right to left (i.e. LSB first). Assuming an exponent e of length $l = 1 + \log_2 e$ bits, the square-and-multiply algorithm executes l modular squaring and roughly l=2 modular multiplications, with the exact number depending on the Hamming weight of e. The number of modular multiplications can be reduced if some extra memory for storing powers of the base m is available. For example, the k-ary exponentiation method (also called window method) processes k bits of the exponent e at a time, thereby reducing the number of modular multiplications to l=k in the worst case. However, the k-ary exponentiation requires pre-computation and storage of 2k powers of the base m (see Algorithm 14.82 in [24]), which is why this method is rarely implemented on resource-constrained embedded devices like smart cards. If the base m is fixed and known a-priori (which is, for example, the case when generating a DSA signature), the number of both modular multiplications and squaring can be reduced through the fixed-based comb method as described in [24]. The execution time of a modular exponentiation depends heavily on the implementation of the two operations it consists of, namely modular multiplication and modular squaring. Both operations include a modular reduction, which can be efficiently performed using the well-known Montgomery technique [25]. Kofic et al [22] describe several optimized software algorithms for Montgomery multiplication, among these is the so-called Coarsely Integrated Operand Scanning (CIOS) method. The CIOS method executes a total of $2s^2 + s$ single-precision (i.e. w-bit) multiply instructions, whereby n denotes the number of w-bit words that are needed to accommodate an n-bit operand, i.e. s = n/w (see [22] for a detailed analysis).

2.2 Pell's Equation

We describe a cyclic group G_p over the Pell's equation $x^2 - D * y^2 \equiv 1 \pmod P$, where P is an odd prime. Some properties of the group G_p are then deduced. These properties are also found in the group G_N over the Pell's equation $x^2 - D * y^2 \equiv 1 \pmod N$, where N is a product of two primes. This group G_N then developed to be a public key crypto scheme based on Pell's equations over the ring Z_N *. From the group G_N, we find a group isomorphism mapping f : $G_N \rightarrow Z_N$ *such that a solution (x, y) of the Pell's equation $x^2 - D * y^2 \equiv 1 \pmod N$, can easily be transformed to unique element u in Z_N * . This implies that the plain texts/cipher texts in the in the group G_N can easily transformed to the corresponding plain texts/cipher texts in the Pells equation scheme.

Legendre symbol

Let a be an integer and p > 2 a prime, Define the Legendre Symbol (a/p)=0,1,-1 as follows

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p \text{ divides } a \\ 1 & \text{if a is Quadratic Residue mod p} \\ -1 & \text{if a Non Quadratic Residue mod p} \end{cases}$$

Jacobi Symbol

If p is a positive odd integer with prime factorization

$$P = \prod_{i=1}^{r} p_i^{ai} .$$

The Jacobi symbol (n / p) is defined for all integers n by the equation

$$(n / p) = \prod_{i=1}^{r} (n / p_i)^{ai}$$

where (n / p_i) is the legendre symbol. The computational cost of a modular exponentiation can be reduced if the exponent e and/or the base m are suitably chosen, which is possible for RSA as well as Pell's Equation with Jacobi Symbol. For example, it is common practice to choose a small public exponent in Pell's Equation; a typical value is 2^{16} +1. In this case, operations involving the public exponent (e.g. Pells Equation encryption) are significantly faster than operations involving the private exponent, even when the latter are supported by the Chinese Remainder Theorem [24]. On the other hand, Pells equation with Jacobi symbol can use special primes to simplify the reduction operation. In addition, the generator g used in Pells key exchange can be small (e.g. g = 2), which reduces the cost of a modular exponentiation. Pells Equation implementations generally take advantage of a generator g that generates a (large) subgroup of Z* p, 160-bit subgroup when p is a 1024-bit prime, which considerably alleviates the computational burden of a modular exponentiation with g as base.

3 Secure Sockets Layer (SSL) Protocol

The Secure Sockets Layer (SSL) protocol and its successor, the Transport Layer Security (TLS) protocol, are standardized protocol suites for enabling secure communication between a client and a server over an insecure network [8]. The main focus in the design of these protocols lay in modularity, extensibility, and transparency. Both SSL and TLS use a combination of asymmetric (i.e. publickey) and symmetric (i.e. secret-key) cryptographic techniques to authenticate the communicating parties and encrypt the data being transferred. The actual algorithms to be used for authentication and encryption are negotiated during the handshake phase of the protocol. SSL/TLS supports traditional public-key Pells Equation with Jacobi Symbol cryptosystems.

3.1 SSL Handshake

The SSL protocol contains several sub-protocols, one of which is the handshake protocol. After agreeing upon a cipher suite that defines the cryptographic primitives to be used and their domain parameters, the server (and possible the client too) is authenticated and a pre-master secret is established using publickey techniques. When using an Pells equation with Jacobi Symbol cipher suite, the pre-master secret is established through key transport: The client generates a random number and sends it in Pells-encrypted form to the server.

Client sends the message its supported cipher suites to the server, who confirms the selected suite in its own message. Then, the server transmits its certificate and an optional request for authentication to the client. In most cases there is no mutual authentication and only the server presents its certificate to the client. The client is rarely authenticated during the handshake phase, but rather thereafter, e.g. by sending a password to the server. The client then verifies the server's certificate and answers with the message, containing the material needed for the server to derive the shared pre-master secret. If the public key extracted from the server's certificate cannot be used for encryption (e.g. because it is only authorized to signing), then the server sends a message including a second public key. The cipher suite is just a status message, telling both parties to use the negotiated suite from now on. The final message is then the first one encrypted with the selected cipher and the symmetric key, derived from the pre-master secret. The expensive steps in this process are the verification of the certificate's signature (using, for example, Pells equation with Jacobi Symbol) and the establishment of the shared pre-master secret, which is usually done in one of two ways, depending on the cipher suite chosen.

- If Pells Equation with Jacobi Symbol is chosen, the client creates a random value, encrypts it with the server's public key (one modulo exponentiation) and sends the result back to the server, who can decrypt it (another modulo exponentiation).

3.2 Advantages of Pell's Equation with Jacobi Symbol over RSA Cipher Suites

When an RSA cipher suite is used for the handshake, the client has to perform two modular exponentiations: one to verify the RSA signature contained in the server's certificate, and the other to encrypt the pre-master secret. Both of these exponentiations are carried out with public exponents, which are usually small [18, 31]. Unfortunately, when using an Pells equation with Jacobi symbol based cipher suite, the situation is less favourable for the client. In summary, It is widely believed that RSA cipher suites are to prefer over their Pells equation with Jacobi Symbol based counterparts when the SSL client runs on an embedded device with modest resources, while Pells equation cipher suites yield considerable better performance (i.e. throughput) figures on the server side [15, 30].

- Results from the literature confirm that a 1024-bit Pells Equation signature can be verified significantly faster than a 160-bit RSA signature. However, the picture changes with higher security levels (i.e. longer keys).

- Even though this paper focuses on client-side acceleration of SSL, it should be noted that Pells equation with Jacobi symbol offers significant performance-advantages for SSL servers [15, 16]. RSA cipher suites are highly computation-intensive on the server side [7, 36], which may also impact the overall latency of the handshake, in particular if the server is under heavy load.
- Using a cipher suite with ephemeral Pells equation key exchange provides forward secrecy, whereas RSA-based key transport does not [4]. Another advantage of Pells equation key exchange is that both the client and the server can contribute randomness to the generation of the pre-master secret, which is not the case with RSA-based key transport.

4. Algorithms and Results

4.1 Algorithm

Key Generation:

Assume that the Sender to send messages E (cipher text) and Receiver case receive the messages E and M .

1. Choose two primes p and q $(p \neq q)$
2. put η_p = p mod 4 and η_q = q mod 4 where η_p , $\eta_q \in \{1,-1\}$
3. Find non square integer D > 0 such that Legender symbols

$$(D/p) = -\eta_p \quad \text{and} \quad (D/p) = -\eta_q$$

4. Compute n = p * q and m = (p+η_p) (q+η_q) / 4
5. Select a integer value for S such that the jacobi symbol ((S^2- D)/n) = -1.
6. Select a integer value for e such that (e , m)=1. and makes {n, e, S, D} as public.
7. Solve d * e ≡ (m+1)/2 mod m for d and keeps as private key
8. D is Key for the Cipher Text.

Encryption:

1. Let M be a message to communicate / encrypt
2. Compute $j_1 = ((M^2-D)/n)$
3. If $j_1 = 1$ go to step (4) else go to step (6)
4. Compute $x \equiv (M^2 + D) / (M^2 - D) \pmod n$ And $y \equiv 2M / (M^2-D) \pmod n$
5. Go to step (8)
6. If $j_1 = -1$ go to step (7) else go to stop().
7. Compute $x \equiv ((M^2+D)(S^2 + D) + 4DMS) / ((M^2-D)(S^2- D)) \pmod n$ And $y \equiv (2S(M^2+D) + 2M(S^2 + D)) / ((M^2-D)(S^2- D)) \pmod n$
8. Compute $j_2 = x \pmod 2$ where $j_2 \in \{0, 1\}$ (nothing that $x^2- Dy^2=1 \pmod n$ for these values of x, y and assume that (y, n) =1)
9. Put $X_l = x$ and $Y_l = y$
10. Compute $(X_{l+1}, X_l) \pmod n$ such that

$$\text{if } i \neq j$$
$$X_{i+j} = X_j + D\,Y_i\,Y_j \text{ and}$$
$$Y_{i+j} = X_i\,Y_j + X_j\,Y_i$$
$$\text{if } i = j$$
$$X_{2i} = X_i^2 + D\,Y_i^2 \text{ or}$$
$$2X_i^2 - 1 \text{ and } Y_{2i} = 2X_i\,Y_i$$

11. Compute $E = DYX_l (X_{l+1} - x\,X_l)^{-1} \pmod n$ (here E is the cipher text) with 0<E<n
12. Send the {E, j_1, j_2) = cipher text.

Decryption

After the ciphertext (E, j_1,j_2), checks that $x^2 - D\,y^2 \equiv 1$

1. Compute $X_{2l} = (E^2 + D) / (E^2 - D) \pmod n$ and
   i. $Y_{2l} = (2E / (E^2 - D)) \pmod n$
2. Compute $X_d(X_{2e}) \equiv X_{2de}(x) \pmod n$ and
   i. $X_{d+1}(X_{2e}) \equiv X_{2de+2e}(x) \pmod n$
   ii. We have $X_{2ed} = \sigma\ x \pmod n$ and $j_2 \equiv x \pmod 2$
3. Compute σ and therefore determines x (mod n) And find $y \equiv \sigma\ Y_{2de} \equiv \sigma\ (X_{2de+2e} - X_{2e}\,X_d) / DX_{2e} \pmod n$ we have $t \equiv x + y\sqrt{D} \pmod n$ Compute t^1 such that

$$t^1 = t \text{ if } j_1 = 1 \text{ else if } j_1 = -1$$
$$t^1 = t\ (S-\sqrt{D}) / (S+\sqrt{D})$$
$$\text{And } t^1 = (M+\sqrt{D}) / (M-\sqrt{D}) \pmod n$$

4. Compute $M \equiv (t^1+1)(\sqrt{D}) / (t^1-1) \pmod n$

The Sender sends message (M) to key generation function that produces a secure private key (d). This private key is then encrypted with public key cryptosystem using the sender private key to form the result. Both the message and the result are pretended and then transmitted. The receiver takes the message (M) and produces a secure private key (d). The receiver also decrypts the result using the sender public key. If the calculated secure private key (d) matches the decrypted results, the result is accepted as valid. Because only the sender knows the private key and only the sender could have produced a valid result.

Results

Timings for RSA for varying bit strengths (512 to2560)

Table I (RSA Scheme)

Key Generation µs (milliseconds)	Encryption µs(milliseconds)	Decryption µs (milliseconds)
0.057984	0.054362	0.90318
0.194653	0.065302	2.098904
0.465994	0.078851	3.365591
0.657473	0.089712	4.437929
1.613467	0.105439	5.804798
2.057411	0.116585	7.430849
4.052181	0.126361	9.001286
6.734534	0.146157	11.523721
8.324152	0.1634512	14.748235

Timings for Pell's Equation with Jacobi Symbol for varying bit strengths (512 to 2560)

Table 2 (Pell's with Jacobi symbol Scheme)

Key Generation µs(milli seconds)	Encryption Time µs(millisec onds)	Decryption Time µs (millisecon ds)
0.049876	0.044362	0.80498
0.187692	0.059502	1.797604
0.453265	0.069651	3.032591
0.563421	0.079212	4.142814
1.602341	0.099439	3.129204
2.056323	0.109938	7.230849
4.052181	0.113361	7.249686
6.712345	0.121213	8.122532
8.312423	0.128745	8.768374

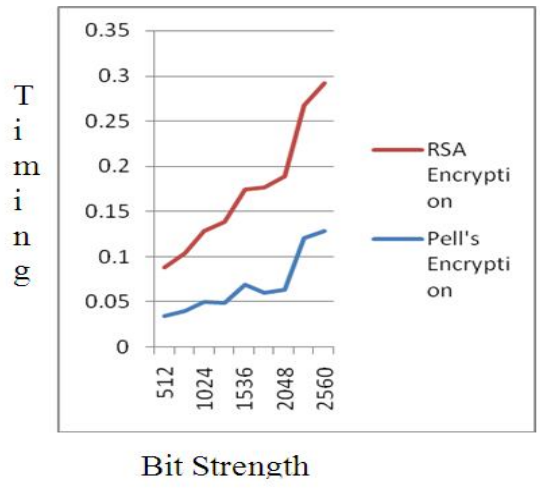

Figure 1 (Comparison of the Encryption algorithms of Pell's with Jacobi symbol, with existing RSA algorithm)

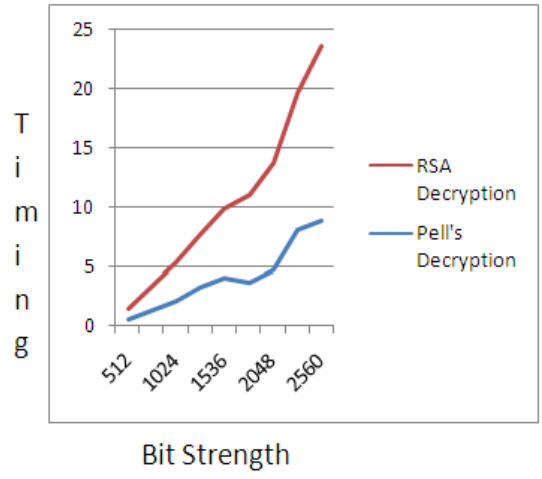

Figure 2 (Comparison of the Decryption algorithms of Pell's with Jacobi symbol, with existing RSA algorithm)

5 Conclusions

We presented a hardware/software co-design of the SSL handshake based on instruction set extensions for the low-level arithmetic operations carried out in public-key Pells Equation with Jacobi Symbol cryptography. Our solutions offer a significant gain in performance for field arithmetic as well as for an entire handshake. In addition, we have shown that the speed-up achieved in the low-level operations (i.e. the field arithmetic) propagates almost lossless up to the highest layers of the SSL protocol. So, by speeding up field multiplication and squaring using instruction set extensions, the entire high-level SSL handshake can be sped up by almost the same factor.

References

[1]. G. Apostolopoulos, V. G. Peris, P. Pradhan, and D. Saha. Securing electronic commerce: Reducing the SSL overhead. IEEE Network, 14(4):8{16, July 2000.

[2]. P. G. Argyroudis, R. Verma, H. Tewari, and D. E. O'Mahony. Performance analysis of cryptographic protocols on handheld devices. In Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications (NCA 2004), pp. 169{174. IEEE Computer Society Press, 2004.

[3]. D. G. Berbecaru. On measuring SSL-based secure data transfer with handheld devices. In Proceedings of 2nd IEEE International Symposium on Wireless Communication Systems (ISWCS 2005), pp. 409{413. IEEE, 2005.

[4]. I. F. Blake, G. Seroussi, and N. P. Smart. Advances in Elliptic Curve Cryptography. Cambridge University Press, 2005.

[5]. S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. M□oller. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). Internet Engineering Task Force, Network Working Group, RFC 4492, May 2006.

[6]. M. K. Brown et al., PGP in constrained wireless devices. In Proceedings of the 9th USENIX Security Symposium (SECURITY 2000), pp. 247{261. USENIX Association, 2000.

[7]. C. Coarfa, P. Druschel, and D. S. Wallach. Performance analysis of TLS Web servers. ACM Transactions on Computer Systems, 24(1):39{69, Feb. 2006.

[8]. T. Dierks and E. K. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. Internet Engineering Task Force, Network Working Group, RFC 4346, Apr. 2006.

[9]. A. O. Freier, P. Karlton, and P. C. Kocher. The SSL Protocol Version 3.0. Internet Draft, available for download at http://wp.netscape.com/eng/ssl3/draft302. txt, Nov. 1996.

[10]. J. Gaisler. The LEON-2 Processor User's Manual (Version 1.0.10). Available for download at http://www.gaisler.com/doc/leon2-1.0.10.pdf, Jan. 2003.

[11]. J. Grofisch□adl and G.-A. Kamendje. Architectural enhancements for Montgomery multiplication on embedded RISC processors. In Applied Cryptography and Net-work Security | ACNS 2003, LNCS 2846, pp. 418{434. Springer Verlag, 2003.
[12]. J. Grofisch□adl and E. Savafis. Instruction set extensions for fast arithmetic in finite fields GF(p) and GF(2m). In Cryptographic Hardware and Embedded Systems | CHES 2004, LNCS 3156, pp. 133-147. Springer Verlag, 2004.
[13]. J. Grofisch□adl, S. Tillich, A. Szekely, and M. Wurm. Cryptography instruction set extensions to the SPARC V8 architecture. Preprint, submitted for publication.
[14]. V. Gupta and S. Gupta. Experiments in wireless internet security. In Proceedings of the 3rd IEEE Conference on Wireless Communications and Networking (WCNC 2002), vol. 2, pp. 860{864. IEEE, 2002.
[15]. V. Gupta, S. Gupta, S. Chang Shantz, and D. Stebila. Performance analysis of elliptic curve cryptography for SSL. In Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe 2002), pp. 87{94. ACM Press, 2002.
[16]. V. Gupta et al., Speeding up secure Web transactions using elliptic curve cryptography. In Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS 2004), pp. 231{239. Internet Society, 2004.
[17]. N. Gura et al., An end-to-end systems approach to elliptic curve cryptography. In Cryptographic Hardware and Embedded Systems | CHES 2002, LNCS 2523, pp. 349{365. Springer Verlag, 2002.
[18]. P. Gutmann. Performance characteristics of application-level security protocols. Unpublished manuscript, available for download at http://www.cs.auckland.ac. nz/~pgut001/pubs/appfisec.pdf, 2005.
[19]. D. R. Hankerson, A. J. Menezes, and S. A. Vanstone. Guide to Elliptic Curve Cryptography. Springer Verlag, 2004.
[20]. Institute of Electrical and Electronics Engineers (IEEE). IEEE Std 1363-2000: IEEE Standard Specifications for Public-Key Cryptography, Aug. 2000.
[21]. A. A. Karatsuba and Y. P. Ofman. Multiplication of multidigit numbers on automata. Soviet Physics - Doklady, 7(7):595{596, Jan. 1963.
[22]. Cfi . K. Kofic, T. Acar, and B. S. Kaliski. Analyzing and comparing Montgomery multiplication algorithms. IEEE Micro, 16(3):26{33, June 1996.
[23]. M. Koschuch, J. Grofisch□adl, U. Payer, M. Hudler, and M. Kr□uger. Workload characterization of a lightweight SSL implementation resistant to side-channel attacks. In Cryptology and Network Security | CANS 2008, LNCS 5339, pp. 349-365.Springer Verlag, 2008.
[24]. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.
[25]. P. L. Montgomery. Modular multiplication without trial division. Mathematics of Computation, 44(170):519{521, Apr. 1985.
[26]. National Institute of Standards and Technology (NIST). Recommendation for Key Management { Part 1: General (Revised). Special Publication 800-57, available for download at http://csrc.nist.gov/publications/PubsSPs.html, Mar. 2007.
[27]. National Security Agency (NSA). NSA Suite B Cryptography. Fact sheet, available online at http://www.nsa.gov/ia/programs/suitebficryptography/ , Mar. 2008.
[28]. OpenSSL Project. OpenSSL 0.9.7k. Available for download at http://www. openssl.org, Sept. 2006.
[29]. PeerSec Networks, Inc. MatrixSSL 1.7.1. Available for download at http://www. matrixssl.org, Sept. 2005.
[30]. N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. IEEE Transactions on Mobile Computing, 5(2):128{143, Feb. 2006.
[31]. N. R. Potlapally, S. Ravi, A. Raghunathan, and G. Lakshminarayana. Optimizing public-key encryption for wireless clients. In Proceedings of the 37th IEEE Inter- national Conference on Communications (ICC 2002), volume 2, pp. 1050{1056. IEEE, May 2002.
[32]. S. Ravi, A. Raghunathan, and N. R. Potlapally. Securing wireless data: System architecture challenges. In Proceedings of the 15th International Symposium on System Synthesis (ISSS 2002), pp. 195{200. ACM Press, Oct. 2002.
[33]. E. K. Rescorla. SSL and TLS: Designing and Building Secure Systems. Addison-Wesley, 2000.
[34]. Standards for Eficient Cryptography Group (SECG). SEC 1: Elliptic Curve Cryptography. Available for download at http://www.secg.org/download/aid-385/sec1fifinal.pdf, Sept. 2000.
[35]. W. H. Wolf. Hardware-software co-design of embedded systems. Proceedings of the IEEE, 28(7):967{989, July 1994.
[36]. L. Zhao, R. Iyer, S. Makineni, and L. Bhuyan. Anatomy and performance of SSL processing. In Proceedings of the 5th International Symposium on Performance Analysis of Systems and Software (ISPASS 2005), pp. 197{206. IEEE Computer
[37]. Society Press, 2005.

AUTHOR

Mr. M. Kondala Rao is a research scholar in the Department of Computer Science and Systems Engineering of Andhra University, under the supervision of Prof.P.S.Avadhani and Prof.D.Lalitha Bhaskari. He received his M.Tech (CST) from Andhra

University. He is a Life Member of CRSI. His research areas include Network Security, Cryptography, and Web Security.

Dr. P. S. Avadhani is a Professor in the department of Computer Science and Systems Engineering and Principal of AU College of Engineering, Andhra University. He has guided 10 Ph. D students and right now he is guiding 12 Ph. D Scholars. He has guided more than 100 M.Tech. Projects. He received many honors like best researcher award and best academician award from Andhra University, chapter patron award from CSI for CSI-Visakhapatnam Chapter and he has been the member for many expert committees, member of Board of Studies for various universities, Resource person for various organizations. He is a Life Member in CSI, CRSI AMTI, ISIAM, ISTE, YHAI and in the International Society on Education Technology.

Dr. D. Lalitha Bhaskari is a Professor in the Department of Computer Science and Systems Engineering of Andhra University. She is guiding more than 8 Ph. D Scholars from various institutes. Her areas of interest include Theory of computation, Data Security, Image Processing, Data communications, Pattern Recognition. She is a Life Member of CSI and CRSI. Apart from her regular academic activities she holds prestigious responsibilities like Associate Member in the Institute of Engineers, Associate Member in the Pentagram Research Foundation, Hyderabad, India. She also received young engineer award from Institute of Engineers (India) in the year 2008.