Performance and Security Aspects of Client-Side SSL/TLS Processing on Mobile Devices

Johann Großschädl

University of Luxembourg,
Laboratory of Algorithmics, Cryptology and Security (LACS),
6, rue Richard Coudenhove-Kalergi, L–1359 Luxembourg, Luxembourg
johann.groszschaedl@uni.lu

Abstract. The SSL/TLS protocol is the de-facto standard for secure Internet communications, and supported by virtually all modern e-mail clients and Web browsers. With more and more PDAs and cell phones providing wireless e-mail and Web access, there is an increasing demand for establishing secure SSL/TLS connections on devices that are relatively constrained in terms of computational resources. Therefore, the efficient implementation of the cryptographic primitives executed on the client side of the SSL/TLS protocol is essential for the emergence of a wireless Internet with strong end-to-end security. In addition, the cryptographic primitives need to be protected against side-channel analysis since an attacker may be able to monitor, for example, electromagnetic emanations from a mobile device. Using an RSA-based cipher suite has the advantage that all modular exponentiations on the client side are carried out with public exponents, which is uncritical in terms of performance and side-channel leakage. However, the current migration to AES-equivalent security levels makes a good case for using an Elliptic Curve Cryptography (ECC)-based cipher suite. We demonstrate in this paper that, for high security levels, ECC-based cipher suites outperform their RSA counterparts on the client side, even though they require the integration of diverse countermeasures against side-channel attacks. In addition, we propose a new countermeasure to protect the symmetric encryption of application messages (i.e. bulk data) against Differential Power Analysis (DPA) attacks. This new countermeasure, which we call *inter-block shuffling*, is based on an "interleaved" encryption of several 128-bit blocks of data (using, for example, the AES), and randomizes the order in which the individual rounds of the individual blocks are executed. Our experimental results show that inter-block shuffling is a highly effective countermeasure as it provides excellent DPA-protection at the expense of a slight degradation in performance.

Extended Abstract

In the past, research in network security was conducted under the assumption that the endpoints of a communication channel are secure; an adversary could only attack the communication itself. A typical attack in this scenario started

with eavesdropping on network traffic, followed by the modification, injection, or replay of messages with the goal to compromise the security of (parts of) the network [9]. However, with the current paradigm shift to more and more cell phones, PDAs, and other mobile or embedded devices being used to access the Internet, this adversary model must be adapted to incorporate attacks on the communication endpoints themselves too. For example, an adversary can try to obtain the secret key(s) used to encrypt the communication by analyzing side-channel information (e.g. power consumption or EM emanations) leaking from a device [10,5]. An EM attack on a cell phone or PDA may even be conducted without the owner of the device being able to notice it [13]. Therefore, secure networking does not only require cleverly devised protocols, but also a secure implementation of these protocols and the associated cryptographic algorithms [11]. In particular, the cryptographic algorithms must be protected against all possible forms of side-channel attack.

The "de-facto" standard for secure communication over an insecure, open network like the Internet is the Secure Sockets Layer (SSL) protocol [4] and its successor, the Transport Layer Security (TLS) protocol [3]. Both use a combination of public-key and secret-key cryptographic techniques to guarantee the confidentiality, integrity, and authenticity of data transfer between two parties (typically a client and a server). The SSL protocol is composed of two layers and includes a number of sub-protocols [4]. At the lower level is the SSL Record Protocol, which specifies the format of data transmission between client and server, including encryption and integrity checking [4]. It encapsulates several 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[1]), as well as the establishment of a *pre-master secret* via asymmetric (i.e. public-key) techniques [4]. 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.

The SSL/TLS protocol is algorithm-independent (or algorithm-agile) in the sense that it supports different algorithms for one and the same cryptographic operation, and allows the communicating parties to make a choice among them [4]. At the beginning of the handshake phase, the client and the server negotiate a *cipher suite*, which is a well-defined set of algorithms for authentication, key agreement, symmetric encryption, and integrity checking. Both SSL and TLS specify the use of RSA or DSA for authentication, and RSA or Diffie-Hellman for key establishment. In 2006, the TLS protocol was revised to support Elliptic Curve Cryptography (ECC) [1,8], and since then, cipher suites using ECDH for key exchange and ECDSA for authentication can be negotiated during the handshake phase [2]. The results from [6] and [7] clearly show that SSL/TLS

[1] Most Internet applications use SSL only for server-side authentication, which means that the server is authenticated to the client, but not vice versa. Client authentication is typically done at the application layer (and not the SSL layer), e.g. by entering a password and sending it to the server over a secure SSL connection.

servers reach significantly better performance and throughput figures when the handshakes are carried out with ECC instead of RSA. On the client side, however, the situation is not that clear since the cryptographic operations executed during the handshake seem to favor RSA cipher suites over their ECC-based counterparts. When using an RSA cipher suite, all modular exponentiations on the client side are performed with public exponents, which are typically small [4]. In the case of an ECC-based cipher suite, however, the client has to execute two scalar multiplications for ephemeral ECDH key exchange, and at least one double-scalar multiplication to validate the server's certificate[2], which is quite costly. In addition, the scalar multiplications for ECDH key exchange need to be protected against Simple Power Analysis (SPA) attacks, whereas RSA-based key transport (or, more precisely, the encryption of a random number using the public RSA key from the server's certificate) is rather uncritical with respect to side-channel leakage from the client.

It is widely presumed that, due to efficiency reasons, RSA cipher suites are better suited for SSL/TLS handshake processing on resource-restricted clients than ECC-based cipher suites. For example, Gupta et al. compared in [6] the handshake time of OpenSSL 0.9.6b using a 1024-bit RSA cipher suite versus a 163-bit ECC cipher suite, and found the former outperforming the latter by 30% when executed on a PDA operating as client. VeriSign, a major international Certification Authority (CA), prefers RSA cipher suites over their ECC-based counterparts for mobile clients since, as mentioned in [16], "very few platforms have problems with RSA." However, the ongoing migration to AES-equivalent security levels (e.g. 256-bit ECC, 3072-bit RSA) makes a good case to reassess the "ECC vs. RSA" question for client-side SSL processing. Surprisingly, the relative performance of RSA and ECC-based cipher suites on the client side has not yet been studied for security levels beyond 1024 and 163 bits, respectively (at least we are not aware of such a study). With the present paper we intend to fill this gap and demonstrate that a handshake with a cipher suite based on 256-bit ECC is only slightly slower than a handshake with 3072-bit RSA, while ECC wins big over RSA at higher security levels. To support these claims, we provide a detailed performance analysis of a "lightweight" SSL implementation into which we integrated a public-key crypto library optimized for client-side SSL processing on mobile devices. We also show that the protection of ECDH key exchange against side-channel attacks has almost no impact on the overall handshake time.

Besides ECDH key exchange, also the symmetric encryption of application data (i.e. "bulk data") using a block cipher such as the AES may leak sensitive information through power or EM side channels, which can be exploited by an adversary to mount a Differential Power Analysis (DPA) attack [10]. Numerous

[2] Instead of sending a single certificate to the client, the server may also send a chain of two or more certificates linking the server's certificate to a trusted certification authority (CA). However, throughout this paper we assume that the certificate chain consists of just one certificate, and hence a single signature verification operation is sufficient to check the validity of the certificate.

countermeasures against DPA attacks on the AES have been proposed in the past 10 years; from a high-level point of view they can be broadly categorized into Hiding and Masking [12]. Typical examples of the hiding countermeasure to protect a software implementation of the AES include the random insertion of dummy instructions/operations/rounds and the shuffling of operations such as S-box look-ups. The goal of hiding is to randomize the power consumption by performing all leaking operations at different moments of time in each execution. Masking, on the other hand, conceals every key-dependent intermediate result with a random value, the so-called mask, in order to break the correlation between the "real" (i.e. unmasked) intermediate result and the power consumption. However, masking in software is extremely costly in terms of execution time, whereas hiding provides only a marginal protection against DPA attacks [12]. Therefore, these countermeasures are not very well suited for an SSL/TLS client since the amount of data to be encrypted can be fairly large, and hence a significant performance degradation is less acceptable than, for example, for a smart card application that encrypts just a few 128-bit blocks of data.

In order to solve this problem, we introduce *Inter-Block Shuffling (IBS)*, a new countermeasure to protect the AES (and other round-based block ciphers) against DPA attacks. IBS belongs to the category of "hiding" countermeasures and encrypts/decrypts several 128-bit blocks of data in a randomly interleaved fashion. It can be applied whenever large amounts of data are to be encrypted or decrypted, which is often the case when transmitting emails or HTML files over an SSL connection. The SSL record protocol specifies a payload of up to 2^{14} bytes, which corresponds to 1024 blocks of 128 bits [4]. A straightforward encryption of this payload starts with the first block, then continues with the second block, and so on, until the last block has been processed. However, when using IBS, the individual rounds of the blocks are executed "interleaved" and in random order. More precisely, the encryption starts with the first round of a randomly chosen block, followed by the first round of another randomly chosen block, and so on, until the first round of each block has been performed. Then the encryption of the up to 1024 blocks continues with the second round (again the blocks are processed in random order), followed by the remaining rounds until all rounds of all 1024 blocks have been executed. Of course, IBS can only be used with a non-feedback mode of operation such as the Counter Mode or the Galois/Counter Mode [14,15]. Contrary to IBS, the shuffling countermeasures sketched in the previous paragraph randomize the sequence of operations within *one* block, hence they can be referred to as "intra-block shuffling." Our experimental results show that IBS is significantly more effective than other software countermeasures (in particular intra-block shuffling) as it achieves a high degree of DPA-resistance at the expense of a small performance degradation.

References

1. I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 1999.

2. S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Möller. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). Internet Engineering Task Force, Network Working Group, RFC 4492, May 2006.
3. 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.
4. 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.
5. C. H. Gebotys, S. C. Ho, and C. C. Tiu. EM analysis of Rijndael and ECC on a wireless Java-based PDA. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 250–264. Springer Verlag, 2005.
6. 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)*, pages 87–94. ACM Press, 2002.
7. V. Gupta, D. Stebila, S. Fung, S. Chang Shantz, N. Gura, and H. Eberle. Speeding up secure Web transactions using elliptic curve cryptography. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS 2004)*, pages 231–239. Internet Society, 2004.
8. D. R. Hankerson, A. J. Menezes, and S. A. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Verlag, 2004.
9. C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall, 2002.
10. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer Verlag, 1999.
11. M. Koschuch, J. Großschädl, U. Payer, M. Hudler, and M. Krüger. Workload characterization of a lightweight SSL implementation resistant to side-channel attacks. In M. K. Franklin, L. C. Hui, and D. S. Wong, editors, *Cryptology and Network Security — CANS 2008*, volume 5339 of *Lecture Notes in Computer Science*, pages 349–365. Springer Verlag, 2008.
12. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer Verlag, 2007.
13. E. Mills. Leaking crypto keys from mobile devices. CNET News, available online at `http://news.cnet.com/8301-27080_3-10379115-245.html`, Oct. 2009.
14. N. Modadugu and E. K. Rescorla. AES Counter Mode Cipher Suites for TLS and DTLS. Internet draft, available for download at `http://tools.ietf.org/pdf/draft-ietf-tls-ctr-01.pdf`, June 2006.
15. J. A. Salowey, A. K. Choudhury, and D. A. McGrew. AES Galois Counter Mode (GCM) Cipher Suites for TLS. Internet Engineering Task Force, Network Working Group, RFC 5288, Aug. 2008.
16. VeriSign, Inc. Secure Wireless E-Commerce with PKI from VeriSign. White paper, available for download at `https://www.verisign.com/server/rsc/wp/wap/index.html`, Jan. 2000.