TLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 29, 2007

T. Otto

April 27, 2007

A Privacy-enhancing TLS ciphersuite
draft-otto-tls-sigma-ciphersuite-00.txt

Status of this Memo

Copyright Notice

Abstract

This document describes a TLS ciphersuite which is based on the SIGMA protocol. By its careful adoption in the TLS handshake protocol, the proposed ciphersuite is able to inherit features of the SIGMA protocol. The ciphersuite provides active identity protection, forward secrecy, deniability and adjustable security strength. A similar ciphersuite offering these features has not yet been proposed so far.

Table of Contents

1. Introduction

This document specifies such a new ciphersuite, which encapsulates the SIGMA protocol [SIGMA] into the TLS handshake messages and therefore inherits its valueable features. Further information about SIGMA can be found on the author's website, which is http://www.ee.technion.ac.il/~hugo/sigma.html

In the remainder of this document, we use the term TLS-SIGMA for our proposal.

TLS-SIGMA offers

Forward Secrecy:

This is achieved by the authenticated Diffie-Hellman key exchange which is the cryptographic core of the SIGMA protocol.

Adjustability:

The cryptographic strength is determined by the choice of the Diffie-Hellman group. We call this feature adjustable security strength.

Active Identity Protection:

The Identity of the Client is protected against active attacks. This is achieved because the server autenticates prior to the client. Only if the client could identity the server properly, he sends his identity.

Deniability:

In contrast to many other ciphersuites, the conversation between client and server is deniable, in the sense, that by carrying out the TLS-SIGMA handshake, there exists no proof for the server

having talked to the client, at least none which can withstand at a court, and vice versa.

One might argue that there already exist numerous TLS ciphersuites with a DH key exchange, for example TLS_DHE_RSA_WITH_AES_256_CBC_SHA, and ask where the particular advantages of this ciphersuite are.

The crucial point is that with RSA as key exchange mechanism and the mutual authentication case, the client computes in CertificateVerify a signature over all handshake messages (see Section 7.4.8 of [RFC2246]), that is

CertificateVerify = SIG(Client; g^x, g^y, CertServer, CertClient)

and thus provides an undeniable proof that the conversation has taken place.

1.1. TLS and its handshake protocol

TLS has its origin in the SSL protocol developed by Netscape Communications in early 1990s. In the meantime, it became the major protocol to establish a cryptographially protected context between two communicating parties.

One of the most valuable features of TLS is its flexibility in that initially, both sides agree on a set of cryptographic algorithms, a so-called ciphersuite. Such a ciphersuite comprises an algorithm for authentiation and key exchange, a stream or block cipher for bulk encryption and finally, an algorithm for hashing.

While SSL realized this flexibility by a complicated negotiation, TLS has facilitated the procedure, in that the client sends the server all his supported ciphersuites, whereafter the server selects one of them according to his policy or aborts the protocol, if none suitable is among them.

TLS is designed having addition of further ciphersuites in mind.

The TLS handshake protocol's main intention is to

o negotiate certain session parameters,

- authenticate the server to the client, and optionally, the client to the server and
- establish a shared cryptographic secret.

If the handshake has finished successfully, a cryptographically protected channel is established between the two parties, which can be used to exchange securely further data. The message flow of the TLS handshake protocol is shown the following figure.

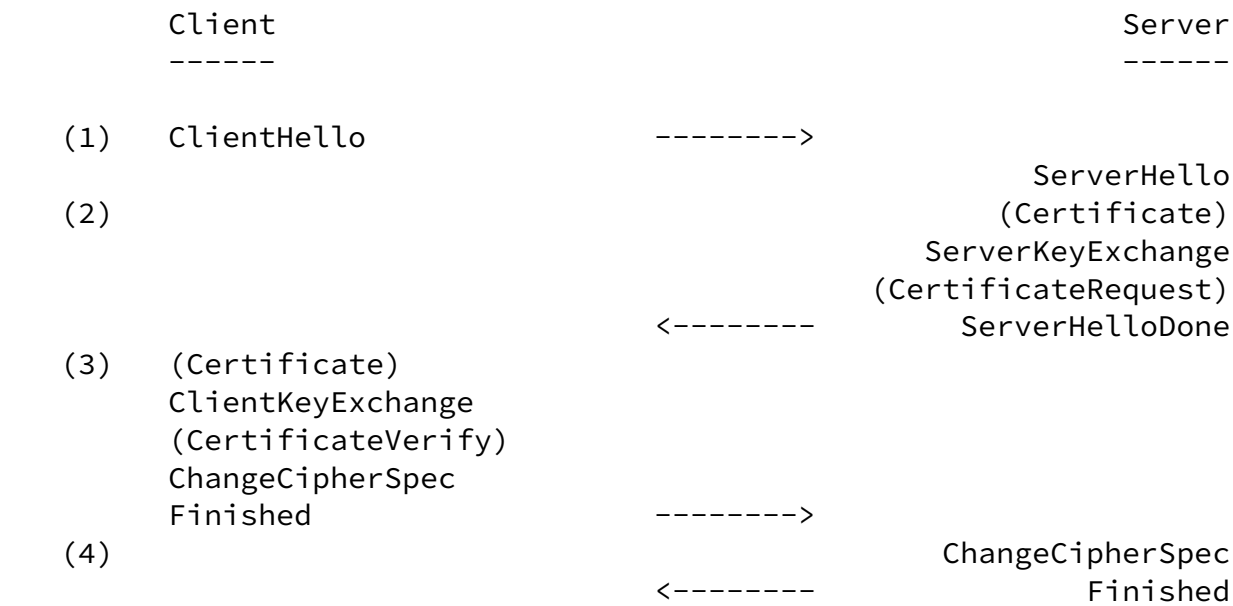

Figure 1: TLS handshake

1.2. The SIGMA protocol

SIGMA is a family of cryptographic key-exchange protocols that provide perfect forward secrecy via a Diffie-Hellman exchange authenticated with digital signatures. It has been proposed already in 1995. It has gained many popularity by building the cryptographic basis for the signature-based modes of IKE and IKEv2.

The protocol has very valuable features which motivated us to incorporate it into TLS.

The SIGMA specification offers two subprotocols, SIGMA-I and SIGMA-R, where I and R stand for Intiator and Responder. SIGMA-I is a three-message protocol and provides active identity protection for the initiator, while SIGMA-R consists of four messages and provides active identity protection for the responder. Obviously, only the SIGMA-I seems to be suitable to be built-in in TLS, so that we restricts on it in the following.

Figure Figure 2 depicts the message flow of SIGMA-I.

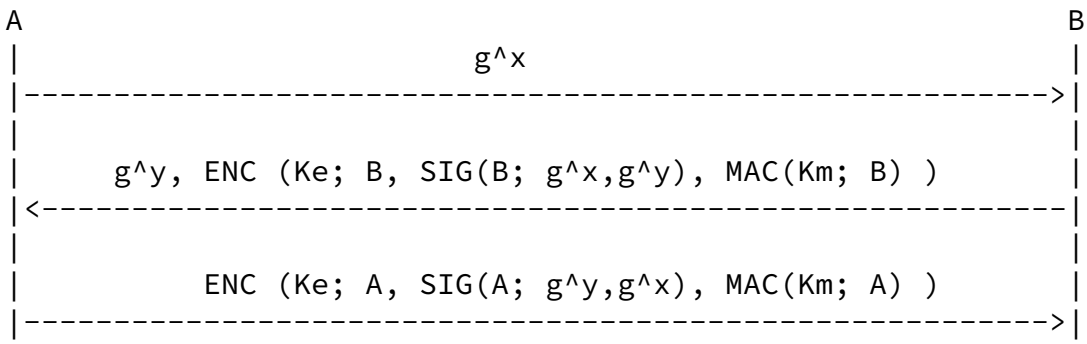

Figure 2: SIGMA-I

The SIGMA specification allows to replace the peer's exponential by a nonce, but we omit this modification. The protocol derives Ke, Km and a session key Ks from the Diffie-Hellman shared key, but they

have to be computationally independent. On page 20 of [SIGMA] the refinement to add some sense of direction to the MAC, i.e. we replace MAC(Km;A) MAC(Km; "0",A) and MAC(Km;B) by MAC(Km; "1",B).

Finally, we replace (according to the rationale on page 21 of [SIGMA]) the pair (SIG(B; g^x,g^y), MAC(Km; B)) by SIG(B; MAC(Km; g^x,g^y,B))) and vice versa for the pair (SIG(A; g^y,g^x), MAC(Km; A)).

The terminology does not deviate too much from existing work. The semantic is as follows. ENC(K;X) stands for encryption of X with key K. g^x and g^y are Diffie-Hellman keys. SIG(A;X) stands for A's signature on the content X. MAC (K;X) stands for computing a MAC over X keyed by K.

Ke and Km are derived from the Diffie-Hellman shared secret g^(xy) through a PRF, while they must be cryptographically independent.

1.3. Requirements notation

In this document, several words are used to signify the requirements of the specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.4. Terminology

This document frequently uses the following terms:

client:

One side of the connection.

server:

The other side of the connection.

2. Protocol Overview

This section describes how SIGMA-I is built in the TLS handshake protocol. Specifying a new ciphersuite means to re-define the semantic or content of existing handshake messages or to add extensions to the initial Hello exchange.

SIGMA-I fits perfectly in the message flow, if the client takes the role of the initiator, and the server of the responder.

First, the client sends in an extension of the TLS ClientHello his Diffie-Hellman public key g^x to the server, together with the DH group he desires. Possible choices are the prime groups defined in IKEv2 [RFC4306] or in [RFC3546]. Table Figure 3 summarizes the choices.

DH group specifier	bits	defined in
0x0001	768	RFC 4306
0x0002	1024	RFC 4306
0x0003	1536	RFC 3546
0x0004	2048	RFC 3546
0x0005	3072	RFC 3546
0x0006	4096	RFC 3546

Figure 3: DH groups

The server then verifies whether the selected / proposed DH group is accceptable. If no, the TLS handshake fails and the server sends a corresponding message to the client. Otherwise, the server selects a private key y, computes g^y and sends this parameter in an extension of the ServerHello back. The Certificate message contains the server's certificate (which corresponds to the identity B in the SIGMA-I message flow), ServerkeyExchange contains the encrypted signature and hash according to message 2 in Figure X.

Both sides are now able to compute the premaster secret. The server computes SK = (g^x)^y, the client computes SK = (g^y)^x. The master secret and keyblock are derived as specified in TLS v1.0.

The client sends now in the Certificate message his certificate

(which corresponds to the identity A in the SIGMA-I message flow), and in ClientKeyExchange the encrypted signature and MAC, according to message 3 in Figure Figure 2. The CertificateVerify message is not sent. For RSA ciphersuites, this message would contain a signature over all previously exchanged handshake messages. Applying this signature would destroy SIGMA's properties.

According to the rationale above, we show the message flow for TLS-SIGMA :

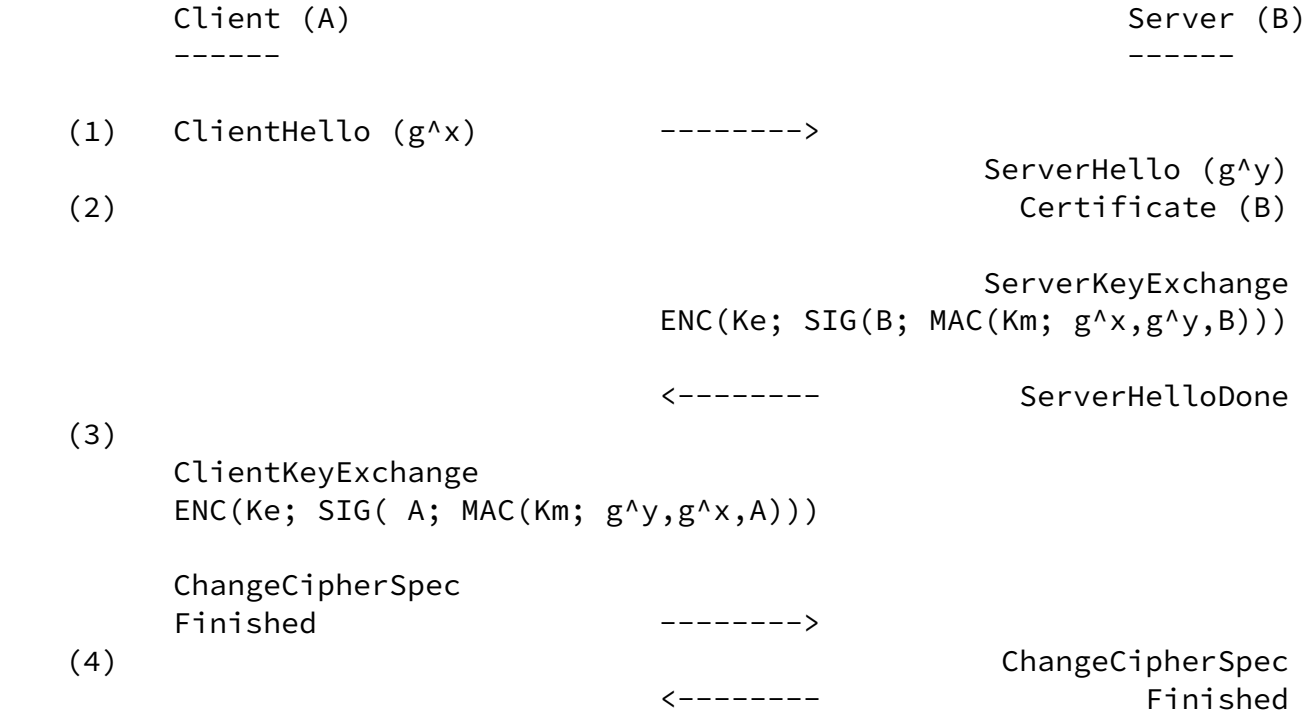

Figure 4: TLS-SIGMA ciphersuite

3. IANA Considerations

-TBD-

4. Security Considerations

-TBD-

5. Acknowledgments

Add your name here.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

6.2. Informative References

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[RFC2408] Maughan, D., Schneider, M., and M. Schertler, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[SIGMA] Hugo Krawczyk, "SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols", Springer LNCS Advances in Cryptography - CRYPTO 2003 Proceedings, LNCS 2729, 2003.

[TLSPSK-Perf] Mario Di Raimondo, Rosario Gennaro, Hugo Krawczyk, "Deniable Authentication and Key Exchange.", CCS 06 (Conference on Computer and Communications Security) URL: , October 2006.

Author's Address

Thomas Otto
Germany

Email: t.otto@tu-bs.de

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).