```
                                          E. Rescorla, A. Schiffman
INTERNET-DRAFT                       Enterprise Integration Technologies
<draft-ietf-wts-shttp-01.txt>              Feb 1996 (Expires August-96)
```

```
                    The Secure HyperText Transfer Protocol
```

Status of this Memo

   This document is an Internet-Draft. Internet-Drafts are working
   documents of the Internet Engineering Task Force (IETF), its areas,
   and its working groups. Note that other groups may also distribute
   working documents as Internet-Drafts.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as ``work in progress.''

   To learn the current status of any Internet-Draft, please check the
   ``1id-abstracts.txt'' listing contained in the Internet-Drafts Shadow
   Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe),
   munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or
   ftp.isi.edu (US West Coast).

   This document describes S-HTTP version 1.2. The original draft of
   this specification, defining S-HTTP version 1.0, was distributed by
   the CommerceNet Consortium in June 1994; in December 1994 a revised
   specification describing S-HTTP version 1.1 was published as an
   Internet Draft (draft-rescorla-shttp-00.txt). In July 1995, an
   updated version of that draft was published as an Internet Draft.
   That document deprecated some unimplemented facilities, provides
   additional clarifying material, and made minor corrections to the
   12/94 version.

   This document implements a decision reached at the December 1995 IETF
   WTS meeting to break up the single S-HTTP document into two docu-
   ments, one describing the S-HTTP messaging protocol and negotiation
   syntax and one describing extensions to HTML to facilitate the use of
   S-HTTP. The companion document is draft-ietf-wts-shtml-00.txt [23].

Abstract

   This memo describes a syntax for securing messages sent using the
   Hypertext Transfer Protocol (HTTP), which forms the basis for the
   World Wide Web. Secure HTTP (S-HTTP) is an extension of HTTP, provid-
   ing independently applicable security services for transaction confi-
   dentiality, authenticity/integrity and non-repudiability of origin.

   The protocol emphasizes maximum flexibility in choice of key manage-
   ment mechanisms, security policies and cryptographic algorithms by
   supporting option negotiation between parties for each transaction.

Rescorla, Schiffman   [Page 2]

Internet-Draft   Secure HTTP

1. Introduction

The World Wide Web (WWW) is a distributed hypermedia system which has
gained widespread acceptance among Internet users. Although WWW
browsers support other, preexisting Internet application protocols,
the native and primary protocol used between WWW clients and servers
is the HyperText Transfer Protocol (HTTP) [18]. The ease of use of
the Web has prompted widespread interest in its employment as a
client/server architecture for many applications. Many such applica-
tions require the client and server to be able to authenticate each
other and exchange sensitive information confidentially. The original
HTTP specification had only modest support for the cryptographic
mechanisms appropriate for such transactions.

Secure HTTP (S-HTTP) provides secure communication mechanisms between
an HTTP client-server pair in order to enable spontaneous commercial
transactions for a wide range of applications. Our design intent is
to provide a flexible protocol that supports multiple orthogonal
operation modes, key management mechanisms, trust models, crypto-
graphic algorithms and encapsulation formats through option negotia-
tion between parties for each transaction.

1.1. Summary of Features

Secure HTTP supports a variety of security mechanisms to HTTP clients
and servers, providing the security service options appropriate to
the wide range of potential end uses possible for the World-Wide Web.
The protocol provides symmetric capabilities to both client and
server (in that equal treatment is given to both requests and
replies, as well as for the preferences of both parties) while
preserving the transaction model and implementation characteristics
of HTTP.

Several cryptographic message format standards may be incorporated
into S-HTTP clients and servers, particularly, but in principle not
limited to, PKCS-7 and PEM. S-HTTP supports interoperation among a
variety of implementations, and is compatible with HTTP. S-HTTP
aware clients can communicate with S-HTTP oblivious servers and
vice-versa, although such transactions obviously would not use S-HTTP
security features.

S-HTTP does not require client-side public key certificates (or pub-
lic keys), supporting symmetric session key operation modes. This is
significant because it means that spontaneous private transactions

can occur without requiring individual users to have an established
public key. While S-HTTP is able to take advantage of ubiquitous
certification infrastructures, its deployment does not require it.

S-HTTP supports end-to-end secure transactions, in contrast with the
original HTTP authorization mechanisms which require the client to
attempt access and be denied before the security mechanism is
employed. Clients may be "primed" to initiate a secure transaction
(typically using information supplied in an HTML anchor); this may be
used to support encryption of fill-out forms, for example. With S-
HTTP, no sensitive data need ever be sent over the network in the
clear.

S-HTTP provides full flexibility of cryptographic algorithms, modes
and parameters. Option negotiation is used to allow clients and
servers to agree on transaction modes (should the request be signed?
encrypted? both? what about the reply?); cryptographic algorithms
(RSA vs. DSA for signing, DES vs. RC2 for encrypting, etc.); and cer-
tificate selection (please sign with your "Mastercard certificate").

S-HTTP attempts to avoid presuming a particular trust model, although
its designers admit to a conscious effort to facilitate multiply-
rooted hierarchical trust, and anticipate that principals may have
many public key certificates.

1.2. Changes

This document describes S-HTTP/1.2. The prior draft described S-
HTTP/1.1. This version adds a number of minor changes, including a
new hash construction and a new way of binding cryptographic parame-
ters to HTML anchors. S-HTTP/1.2 messages will be readable by S-
HTTP/1.1 agents and vice versa, provided that compatible algorithms
are used.

1.3. Processing Model

1.3.1. Message Preparation

The creation of an S-HTTP message can be thought of as a a function
with three inputs:

> 1. The cleartext message. This is either an HTTP message or some
> data object.
> 2. The receiver's cryptographic preferences and keying material.
> This is either explicitly specified by the receiver or subject
> to some default set of preferences.
> 3. The sender's cryptographic preferences and keying material.
> This input to the function can be thought of as implicit
> since it exists only in the memory of the sender.

In order to create an S-HTTP message, then, the sender merges the

sender's preferences with the receiver's preferences. The result of
this is a list of cryptographic enhancements to be applied and keying
material to be used to apply them. This may require some user inter-
vention. For instance, there might be multiple keys available to sign
the message. (See Section 7 for more on this topic.) Using this data,
the sender applies the enhancements to the message cleartext to
create the S-HTTP message.

The processing steps required to transform the cleartext message into
the S-HTTP message are described in Sections 2 and 3. The processing
steps required to merge the sender's and receiver's preferences are
described in Sections 4 and 5.

1.3.2. Message Recovery

The recovery of an S-HTTP message can be thought of as a function of four distinct inputs:

> 1. The S-HTTP message.
> 2. The receiver's stated cryptographic preferences and keying material. The receiver has the opportunity to remember what cryptographic preferences it provided in order for this document to be dereferenced.
> 3. The receiver's current cryptographic preferences and keying material.
> 4. The sender's previously stated cryptographic options. The sender may have stated that he would perform certain cryptographic operations in this message. (Again, see sections 4 and 5 for details on how to do this.)

In order to recover an S-HTTP message, the receiver needs to read the headers and discover what sorts of cryptographic transformations were performed on the message, then remove them using some combination of the sender's and receiver's keying material, in the process while taking note of what enhancements were applied.

The receiver may also choose to verify that the applied enhancements match both the enhancements that the sender said he would apply (input 4 above) and that the receiver requested (input 2 above) as well as the current preferences to see if the S-HTTP message was appropriately transformed. This process may require interaction with the user to verify that the enhancements are acceptable to the user. (See Section 7 for more on this topic.)

1.4. Modes of Operation

Message protection may be provided on three orthogonal axes:

signature, authentication, and encryption. Any message may be signed, authenticated, encrypted, or any combination of these (including no protection).

Multiple key management mechanisms are provided, including password-style manually shared secrets, public-key key exchange and Kerberos [19] ticket distribution. In particular, provision has been made for prearranged (in an earlier transaction) symmetric session keys in order to send confidential messages to those who have no key pair.

Additionally, a challenge-response (``nonce'') mechanism is provided to allow parties to assure themselves of transaction freshness.

1.4.1. Signature

If the digital signature enhancement is applied, an appropriate certificate may either be attached to the message (possibly along with a certificate chain) or the sender may expect the recipient to obtain the required certificate (chain) independently.

1.4.2. Key Exchange and Encryption

In support of bulk encryption, S-HTTP defines two key transfer mechanisms, one using public-key enveloped key exchange and another with externally arranged keys.

In the former case, the symmetric-key cryptosystem parameter is passed encrypted under the receiver's public key.

In the latter mode, we encrypt the content using a prearranged session key, with key identification information specified on one of the header lines. Keys may also be extracted from Kerberos tickets.

1.4.3. Message Integrity and Sender Authentication

Secure HTTP provides a means to verify message integrity and sender authenticity for a HTTP message via the computation of a Message Authentication Code (MAC), computed as a keyed hash over the document

using a shared secret -- which could potentially have been arranged
in a number of ways, e.g.: manual arrangement or Kerberos. This
technique requires neither the use of public key cryptography nor
encryption.

This mechanism is also useful for cases where it is appropriate to
allow parties to identify each other reliably in a transaction
without providing (third-party) non-repudiability for the transac-
tions themselves. The provision of this mechanism is motivated by our
bias that the action of "signing" a transaction should be explicit

Internet-Draft   Secure HTTP

and conscious for the user, whereas many authentication needs (i.e.,
access control) can be met with a lighter-weight mechanism that
retains the scalability advantages of public-key cryptography for key
exchange.

1.4.4. Freshness

The protocol provides a simple challenge-response mechanism, allowing
both parties to insure the freshness of transmissions. Additionally,
the integrity protection provided to HTTP headers permits implementa-
tions to consider the Date: header allowable in HTTP messages as a
freshness indicator, where appropriate (although this requires imple-
mentations to make allowances for maximum clock skew between parties,
which we choose not to specify).

1.5. Implementation Options

In order to encourage widespread adoption of cryptographic facilities
for the World-Wide Web, Secure HTTP deliberately caters to a variety
of implementation options despite the fact that the resulting varia-
bility makes interoperation potentially problematic.

We anticipate that some implementors will choose to integrate an out-
board PEM program with a WWW client or server; such implementations
will not be able to use all operation modes or features of S-HTTP,
but will be able to interoperate with most other implementations.
Other implementors will choose to create a full-fledged PKCS-7 imple-
mentation (allowing for all the features of S-HTTP); in which case
PEM support will be only a modest additional effort. Without com-
pletely prescribing a minimum implementation profile (although see
section 8) then, we recommend that all S-HTTP implementations support
the PEM message format.

2. HTTP Encapsulation

A Secure HTTP message consists of a request or status line (as in
HTTP) followed by a series of RFC-822 style headers followed by an
encapsulated content. Once the content has been decoded, it should
either be another Secure HTTP message, an HTTP message, or simple
data.

For the purposes of compatibility with existing HTTP implementations,
we distinguish S-HTTP transaction requests and replies with a dis-
tinct protocol designator ('Secure-HTTP/1.2'). However, if a future
version of HTTP (i.e., 'HTTP/2.0') subsumes this document use of a
new protocol HTTP designator would provide the same backwards compa-
tibility function and a distinction between such a future version of
HTTP and Secure-HTTP would be unnecessary.

Internet-Draft   Secure HTTP

2.1. The Request Line

For HTTP requests, we define a new HTTP protocol method, 'Secure'.
All secure requests (using this version of the protocol) should read:

        Secure * Secure-HTTP/1.2

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase®
Smarter legal research.