Network Working Group
Request for Comments: 2660
Category: Experimental

E. Rescorla
RTFM, Inc.
A. Schiffman
Terisa Systems, Inc.

August 1999

# The Secure HyperText Transfer Protocol

## Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

## Copyright Notice



## 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) provides independently applicable security services for transaction confidentiality, authenticity/integrity and non-repudiability of origin.

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

## Table of Contents

## 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) [RFC-2616]. The ease of use of the Web has prompted its widespread employment as a client/server architecture for many applications. Many such applications 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, cryptographic algorithms and encapsulation formats through option negotiation between parties for each transaction.

### 1.1. Summary of Features

Secure HTTP is a secure message-oriented communications protocol designed for use in conjunction with HTTP. It is designed to coexist with HTTP's messaging model and to be easily integrated with HTTP applications.

Secure HTTP provides 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, [CMS] and [MOSS]. 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 public keys), as it supports symmetric key-only 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 message headers); 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 (e.g., should the request be signed or encrypted or both -- similarly for the reply?); cryptographic algorithms (RSA vs. DSA for signing, DES vs. RC2 for encrypting, etc.); and certificate selection (please sign with your "Block-buster Video 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.

S-HTTP differs from Digest-Authentication, described in [RFC-2617] in that it provides support for public key cryptography and consequently digital signature capability, as well as providing confidentiality.

## 1.2. Changes

This document describes S-HTTP/1.4. It differs from the previous memo in that it differs from the previous memo in its support of the Cryptographic Message Syntax (CMS) [CMS], a successor to PKCS-7; and hence now supports the Diffie-Hellman and the (NIST) Digital Signature Standard cryptosystems. CMS used in RSA mode is bits on the wire compatible with PKCS-7.

## 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 other data object. Note that since the cleartext message is carried transparently, headers and all, any version of HTTP can be carried within an S-HTTP wrapper.
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 integrates 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 intervention. For instance, there might be multiple keys available to sign the message. (See Section 3.2.4.9.3 for more on this topic.) Using this data, the sender applies the enhancements to the message clear-text 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 3.2.

### 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.)

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.