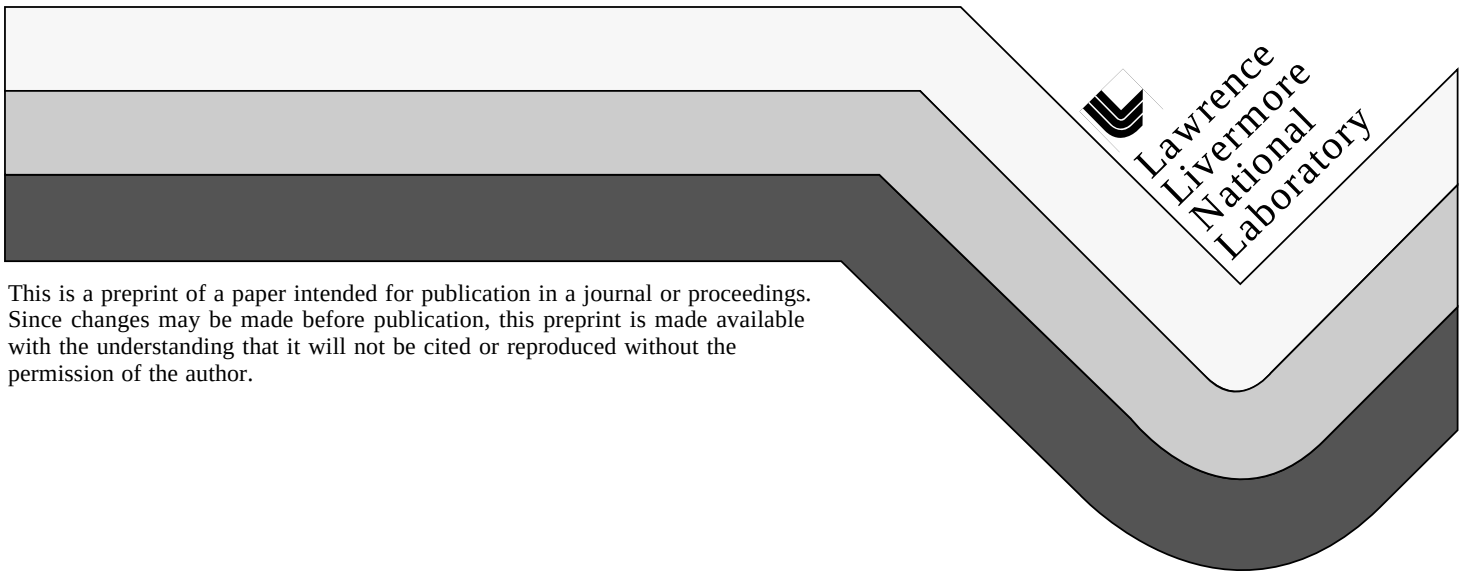
UCRL-JC-125607
PREPRINT

# Group Telemetry Analysis Using the World Wide Web

J. Kalibjian

This paper was prepared for submittal to the
***International Telemetering Conference '96***
***San Diego, CA***
***October 28-31, 1996***

**October 1996**



# Group Telemetry Analysis Using the World Wide Web

**Jeffrey R. Kalibjian**
**Lawrence Livermore National Laboratory**

**Keywords**
secure data sharing, world wide web, hypertext transfer protocol, dual asymmetric key cryptography

## Abstract

Today it is not uncommon to have large contractor teams involved in the design and deployment of even small satellite systems. The larger (and more geographically remote) the team members, the more difficult it becomes to efficiently manage the disbursement of telemetry data for evaluation and analysis. Further complications are introduced if some of the telemetry data is sensitive. An application is described which can facilitate telemetry data sharing utilizing the National Information Infrastructure (Internet).

## Introduction

The World Wide Web (WWW) has transformed the Internet from a research aid into a multi-media display case. The WWW is based on the Hypertext Transfer Protocol (HTTP)-----an object oriented stateless protocol which can be used to build distributed systems in which data representation can be negotiated. Secure HTTP (S-HTTP) is a security enhanced version of HTTP that supports application level cryptographic enhancements (e.g public key cryptography). Although more commonly used for multi-media applications, the World Wide Web offers great promise as a "group-ware" or general data sharing mechanism. Secure HTTP has made it possible to design and implement Web based applications which can accomplish secure data sharing among a group of business or research partners.

After reviewing HTTP, Internet security and the World Wide Web, this paper discusses the design and implementation of a secure Web based data sharing tool. Some design trade-offs impacting data analysis will also be explored.

## HTTP

The Hypertext Transfer Protocol [1] is a very simple communication protocol. A client makes a TCP/IP (Transmission Control Protocol/Internet Protocol, the underlying communication protocol used on the Internet) connection to the server. The server will accept the connection upon which the client will make a document request. The connection domain and the document requested are contained in a Universal Resource Locator (URL), the server responds to the request, the client collects the response, and

finally, the server terminates the connection to the client. The server then continues to listen for other requests. A key element of the interaction is that the server will treat each subsequent request as brand new; that is, it maintains no state. This is contrasted with other Internet protocols (e.g. File Transfer Protocol, FTP) which do maintain state. Thus, HTTP interaction amounts to a connection, request, response, and closure.

A request is basically an action (known more formally as a method) that can be applied to the entity (an object identified by a Universal Resource Identifier, URI) requested. The request also identifies the protocol version in use. More common methods include GET ( retrieve data identified by the URI), and POST (creates a new object linked to the object specified).

A response consists of a status line which contains of a protocol version, status code and its associated text phrase. It is on this line that the server confirms the client request to "speak" in the requested protocol (HTTP). Optional header fields follow including date, and originating location. The message section is next in which the Multipurpose Internet Mail Extension (MIME) [2] content type of the returned data is indicated (typically text/HTML), as well as the number of characters in the message, followed by a blank line, and then the message itself.

Typically, HTTP servers and clients return messages making use of the Hyper Text Markup Language (HTML). HTML [3] can be used to represent formatted text documents, tables, forms, in-line graphics, and hypertext (linked) information. It forms the basis for much of the information obtainable from the World Wide Web. Because of the flexibility of the HTTP protocol, web server and clients capabilities are in a continuous state of evolution delivering ever more increasing power. An example of this is JAVA [4]. JAVA is an object oriented programming language developed by Sun Microsystems. Small programs written in JAVA can be embedded in web pages, so that when the web page is accessed, the program is downloaded to the client and executed. Such small programs are called applets

Key elements of the request/reply paradigm, then, are the concept of negotiated protocol spoken, as well as utilization of MIME headers to specify message content types.

WWW servers may communicate information about objects it receives or manipulates via the Common Gateway Interface specification (CGI) [5]. The server and a CGI program communicate via command line arguments or environment variables. The CGI programs themselves can be written in many programming (e.g. C) or scripting (e.g. Perl) languages. The analogy of such a capability on the client is known as a helper application. This is a stand alone program that is activated by the client browser on detection of a specified MIME type. Data received by the client is forwarded to the helper application for processing. Since the helper application is a stand alone program, it cannot use the web browser window to report or display results of its actions. Instead, it must manage such capabilities on its own.

## Security

When transporting sensitive information over public networks, one must generally have three capabilities present to insure the information will not being comprised. First, there must be assurance that the information being transported can only be read by the intended recipient (privacy). The second notion is that of authentication. The recipient must be able to guarantee that the person he receives data from is truly, "that person." Finally, there must be a guarantee that the contents of the message have not been altered in the message's travel from sender to recipient-----that is, one must have confidence in the message integrity.

Dual asymmetric key cryptography [6] can facilitate these security capabilities. Two keys are generated which have the unique feature that information encrypted with one key, can only be decrypted using the other. Encryption is the process of “disguising” clear text so it cannot be understood. One key is kept private, the other public. If Person A wishes to send a private message to Person B, he may encrypt the message using Person B's public key. Person A is assured that only Person B's private key can decrypt the message. In order for Person B to be assured that the message he is receiving is from Person A, Person A may sign the encrypted message by using his private key. Person B can be assured that only Person A's public key could decrypt the signature. Person B may be assured the message he received was not tampered with, if Person A calculates a checksum of the message he wishes to send before encrypting the message. If the checksum is passed along in the encrypted (and possibly signed) message, Person B can calculate a checksum on the decrypted (and possibly authenticated) file by calculating his own checksum and comparing it with the checksum sent in the message.

At this point is should become clear that both private and public keys need to be protected. The security for the private key is obvious, one desires that only they alone may read their own private messages. This is usually implemented by password protecting utilization of the private key on the host system it resides on. Security is needed for the public key so one can guarantee that no other key may be substituted for their own. This is provided by having a certifying authority sign the public key (forming what is known as an X.509 [7] certificate). The signature indicates that the name associated with the public key (in the certificate) is indeed the “that person.” The certifying authority may require many forms of identification before signing the certificate (e.g. birth certificate, social security number, etc).

## SSL, S-HTTP

These cryptographic principles are utilized in two specifications which have given the World Wide Web security; namely, Secure Sockets Layer (SSL) [8] and the Secure Hyper Text Transfer Protocol (S-HTTP) [9]. SSL is designed to run under the protocol being used for application communication. Thus, while SSL is most commonly used to support secure communication under HTTP, it can also be used to effect secure communication making use of other Internet protocols like FTP. In the SSL

communication process, a client (as usual) contacts the server. The server responds by sending the client its public key certificate. The client validates the signature on the certificate (assuming it has access to the certifying authority public key), generates a symmetric (session) encryption key (this type of key has the property of being able to both encrypt as well as decrypt clear text), and uses the server's public key to encrypt the symmetric key, so it may be sent back to the server. To achieve client authentication, the client would send his certificate back to the server.

While the SSL specification provides for both client and server authentication (as well as data privacy and integrity), in its first implementation, as used in Netscape products (and described above), client authentication was not implemented. The Secure HTTP effort by CommerceNet (a consortium of high technology companies attempting to bring about more rapid utilization of the Internet for commerce activities) implemented both client and server authentication. In the S-HTTP model, security is achieved at the application level; i.e., HTTP has been expanded to incorporate security. In some respects this makes SHTTP more powerful than SSL in that the negotiation features of HTTP apply to security. Thus, for instance, any combination of privacy, authentication and data integrity checking may be specified in a client/server interaction---whereas in the current production versions of SSL one is forced to always encrypt.

Currently SSL and S-HTTP do not interoperate. Implementations of S-HTTP based clients and servers in commercial products are few. Implementations of SSL servers have been much more widespread. SSL based servers and clients which can perform client authentication are currently under beta-test.

## Secure Telemetry Data Sharing

Secure data sharing will occur using the client server paradigm. The server will hold the files to be shared. Each file has an access control list associated with it, indicating the individuals who may access each file. The list can be maintained as simple flat file, or a more complex structure such as a relational database, if other information regarding the files to be shared must be maintained (e.g. whether a file accessed should be transmitted encrypted or unencrypted, etc.).

Clients and servers are connected to the Internet. Access to the telemetry files is managed by a SSL or S-HTTP based Web server. A client wishing to access data connects to the server using his WWW SSL or S-HTTP capable Web browser. Authentication of both the server and client are necessary before data transactions can occur. The client must know that the server is authenticated so he can be assured the data he is receiving is "legitimate." The server must authenticate the client, so he can be assured that he is truly dispensing data to the individual listed on the access control list.

The CGI programming environments for both SSL and S-HTTP allow for such security authentication to be passed to a cgi-bin program. Thus, a cgi-bin dispatcher program can receive the authentication information, and if authentication is indicated, present the client with a form which allows him to request data. The dispatcher will typically

scan all access control information and present the authenticated user with a list of all possible files he may access. The user may then decide whether to select all files, or only a subset.

Once the files of interest are selected, the user must indicate a desired return format for the data (e.g. HTML table, space delimited ASCII file, etc.) and a return MIME type which will allow the secure client browser's helper application to detect data returned from the server. Typically, the helper application will merely collect data and place it in a format so it might be processed by an On Line Analytical Processing (OLAP) tool on the client platform.

## Advanced Concepts

With the emergence of "plug-ins" (like a helper application, except the plug-in communicates with the client browser via a programmatic API), the possibility exists to create an entire telemetry post processing environment that can be accessed through the web browser. One can also conceive of creating JAVA applets that could perform specific analytical functions on returned data. These applets could be offered by the server holding data to be analyzed. Finally, access control functions could be expanded to allow analysts to place post processed data back onto the server so specified individuals could gain access to the analysis results.

## Conclusions

The technology is now in place for geographically remote development teams to perform analysis on a sensitive telemetry product using S-HTTP/SSL client/servers on the Internet. Server CGI programs are the key to managing the data product securely; while client helper applications assure that data accessed can be stored and presented on the analysts local computer.

This work was performed under the auspices of the U.S. Dept. of Energy at LLNL under contract no. W-7405-Eng-48.

## References


[1] Tim Berners-Lee, *Hypertext Transfer Protocol,* Draft, June 1993, Internet Engineering Task Force.

[2] N. Borenstein, N. Freed, *MIME (Multipurpose Internet E-mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, September 1993, RFC1521.

[3] Tim Berners-Lee, Daniel Connolly, *Hypertext Markup Language (HTML)*, June 1993, Internet Engineering Task Force.

[4] Arthur van Hoff, Sami Shaio, Orca Starbuck, *Hooked on JAVA*, February 1996, Addison-Wesley Publishing Company.

[5] NCSA HTTPD Development Team, *The CGI Specification*, NCSA.

[6] Bruce Schneier, *Applied Cryptography*, 1994, John Wiley & Sons, Inc.

[7] The International Telegraph and Telephone Consultative Committee, *Data Communication Networks Directory, Recommendations X.500-X.521*, 1989, IXth Plenary Assembly.

[8] Alan O. Freier, Philip Karlton, Paul C. Kocher, *The SSL Protocol*, Version 3.0, March 1996, Internet Draft.

[9] E. Rescorla, R. Schiffman, *The Secure Hypertext Transfer Protocol*, May 1996, Internet Draft.

*Technical Information Department* • **Lawrence Livermore National Laboratory**
**University of California • Livermore, California 94551**