SECURE INTERNET ACCESSIBLE MATHEMATICAL COMPUTATION FRAMEWORK

DONGDAI LIN[*]

State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China.

ZHIMIN SONG

ERCIST, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China.

PAUL S. WANG[†]

Institute for Computational Mathematics Department of Computer Science, Kent State University, Kent, Ohio 44242, U.S.A.

The *Internet Accessible Mathematical Computation* (IAMC) framework makes supplying/accessing mathematical computation easy on the Internet/Web. The security vulnerabilities of the current IAMC framework is discussed. A scheme for providing securities by using the SSL/TLS protocol is presented. The secure IAMC framework can provide cryptographic authentications, data privacy and integrity.

1 Background

The Mathematical computing is inevitably becoming distributed over the Internet – for easy distribution of mathematical materials; to make specialized computations widely accessible; to allow easy interoperability; and to aggregate functionalities from different systems. Indeed, making mathematical communication easy over the Internet has *many* potential applications.

Cooperating with other institutions worldwide, the Institute of Computational Mathematics(ICM) at Kent State University initiated an IAMC framework project to provide an infrastructure for bringing mathematical computational and educational services over the internet. The IAMC framework[2,5] includes an IAMC client, an IAMC server, and a layered protocol model for connecting IAMC clients and servers effectively and efficiently over the internet. The computation powers of an existing mathematical compute engine

[*]THIS WORK HAS BEEN SUPPORTED IN PART BY THE CHINESE NATIONAL 973 PROJECT NKBRSF G1998030609 AND 863 PROJECT 2001AA144030

[†]WORK REPORTED HEREIN HAS BEEN SUPPORTED IN PART BY THE NATIONAL SCIENCE FOUNDATION UNDER GRANT CCR-9721343 AND INT-9722919, AND IN PART BY THE OHIO BOARD OF REGENTS COMPUTER SCIENCE ENHANCEMENT FUNDS.

can be made available on the Web/Internet by an IAMC server.

Due to the vulnerabilities of present IP-based network, current IAMC framework, for all their virtues, makes it difficult to reliably limit access to sensitive data. Server and client often broadcast data indiscriminately to distant and unpredictable places, servers often assume that all clients to which they provide service are trustworthy, and so on. As a consequence, data and messages delivered between server and client are vulnerable to eavesdropping and interference from unauthorized intruders. Depending on the sensibility of the information and the type of tampering, the potential damage can be significant. To reduce these risks, cryptographic techniques can be used to limit data access while still taking advantage of insecure networks and services.

We consider the addition of security to the Internet Accessible Mathematical Computation framework that executes over insecure IP-based networks. The vulnerabilities of the current IAMC framework is discussed and a scheme for providing securities for IAMC framework by using SSL/TLS is presented. The new framework can then provide cryptographic authentication, data privacy and integrity.

2 IAMC Framework Architecture

The IAMC framework is designed to make mathematical computing easily accessible on the Web/Internet [1,2]. It follows a multi-layer architecture to gain performance and scalability. This architecture can also be expanded easily to meet the requirements for Web-based mathematical computation and education services. Figure 1 shows the overall IAMC framework architecture.

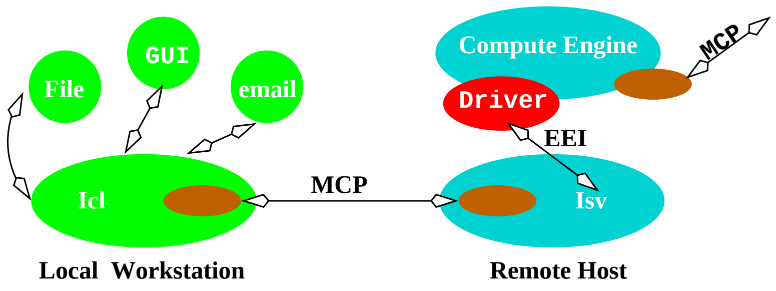

Figure 1. IAMC Architecture

The IAMC framework consists of these components:

1. IAMC client (Icl) — An end-user agent for accessing services provided by any IAMC server.

2. IAMC server (Isv) — A program to provide mathematical computation powers through the MCP protocol. An Isv may or may not employ an external compute engine to perform mathematical computations.

3. Mathematical Computation Protocol (MCP) — IAMC clients and servers are connected by the Mathematical Computation Protocol (Section 3). MCP aims to be a simple and efficient request-response protocol to support both one-time transactions and interactive sessions. A compute engine can obtain support from other servers through MCP.

4. Mathematical Data Encoding — Standard and user-defined mathematical data encodings can be used. Automatic negotiation between the Icl and Isv at the beginning of a computation session determines the encoding(s) used.

5. External Engine Interface (EEI) — A specification and API implementation for binding existing compute engines to IAMC servers.

Currently, IAMC depends on the standard TCP/IP protocol suite for security. Messages and data are easy to be eavesdropped and intercepted; Trusted hosts can be impersonated by exploiting existing vulnerabilities in IP-based networks. These vulnerabilities include attacks based on IP source routing, DNS database corruption and TCP sequence number prediction. Roughly speaking, users have to face the following three security threats in IAMC framework:

1. Overhear, intercept or even modify messages or data being passed between clients and servers.

2. Unauthorized use of important compute server, or impersonate an trusted user to use the compute service.

3. Impersonate a server to cheat clients to get valuable data or provide wrong computation results.

In order to protect mathematical computing in IAMC framework from eavesdropping, tampering or message forgery, and to avoid impersonation, we will introduce SSL/TLS, a security protocol, into IAMC to provide secure communication between clients and servers.

3 The Mathematical Computation Protocol (MCP)

MCP is a well-designed protocol for connecting IAMC clients to servers. It handles one-time computation requests and persistent computation sessions. As described in its original design[2], MCP uses HTTP-style requests and responses to support mathematical computation sessions. Each MCP message consists of a *control line*, a *header*, and an optional *body*. Each header entry is a *key-value pair* on one line terminated by a NEWLINE or a CR-NEWLINE. The header and body are separated by an empty line. The first line of an MCP message is a control line that specifies the *message type*, the *message class*, and a sequence number. The message control line takes one of two forms:

`Request` *Class seqNo*

or

`Response` *Class seqNo statusCode* [*statusString*].

Specific MCP requests belong to different classes that contain request methods to support well-defined operations. An MCP request identifies the request class and a sequence number. The request sequence number *seqNo* is generated by the request originator. A response indicates the class and *seqNo* to identify the request to which it is responding.

Following the OO philosophy, each MCP request class defines a set of methods that can be invoked. Standard MCP request classes include:

- `Initialization` — The initialization class supports session creation and configuration right after client-server connection. Initialization methods include:`welcome`, `connect`, `identification`, `version-negotiation`, `content-negotiation`, `dictionary-map`, `end-initialization`, `CanDo`.
- `Control` — The control class supplies MCP protocol control and management methods for both the client and server side.
- `Computation` — A class of server-side operations to perform application supported computations.
- `Dialog` — A class of client-side methods to solicit information from the end user.

As in HTTP, status codes and strings are used in responses to indicate error and exceptional conditions.

A `Method` header field indicates the class method to handle the request. The other header fields and any message body can be considered arguments to the method. A synchronous request returns with the response, whereas an asynchronous request returns immediately and any response will be delivered when available.

4 Cryptography and SSL/TLS Protocol

4.1 Cryptography

Cryptography has been known for centuries. It is a collection of techniques for keeping information secure. Using cryptography, we can achieve:

- **Confidentiality:** Encryption is used to scramble information sent over the Internet and stored on servers so that eavesdroppers cannot access the data's content. Some people call this quality "privacy".
- **Authentication:** Digital signatures are used to identify the author of a message; people who receive the message can verify the identity of the person who signed them. They can be used in conjunction with passwords or as an alternative to them.
- **Integrity:** Methods are used to verify that a message has not been modified while in transit. Often, this is done with digitally signed message digest codes.
- **Nonrepudiation:** Cryptographic receipts are created so that an author of a message cannot falsely deny sending a message.

Currently there are two dominant cryptographic techniques: secret-key cryptography and public key cryptography. Secret-key cryptography is presumed to be inexpensive and pervasive, and usually be used for encrypting data to provide confidentiality and integrity. Asymmetric (public) key cryptography is considered to be expensive and to be used for key distribution, authentication and certification.

4.2 SSL/TLS protocols

Secure Sockets Layer Version 3.0 (SSL 3.0) protocol is the most popular security protocol, which provides communication privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. SSL 3.0 was published as an Internet-Draft by Netscape Communications Corporation in March 1996, and so far it has been replaced by Transport Layer Security Version 1.0 (TLS 1.0) protocol. TLS 1.0 is a Proposed Standard, RFC2246, published by IETF TLS Working Group in January 1999. TLS 1.0 is totally based on SSL 3.0 and only a minor modification to SSL 3.0, so informally referred to as “SSL 3.1”. For simplicity, here we use the name TLS for the protocol.

The TLS protocol includes two sub-protocols: the TLS record protocol and the TLS handshake protocol. The TLS Record Protocol takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result. Received data is decrypted, verified, decompressed, and reassembled, then delivered to higher levels. The TLS Handshake Protocol allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data.

TLS is application protocol independent. It runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. Higher level protocols can layer on top of the TLS Protocol transparently(see Figure 2). It uses TCP/IP on behalf of the higher-level protocols, and in the process allows TLS-enabled server and TLS-enabled client to authenticate to each other, and allows both parties to establish an encrypted connection.

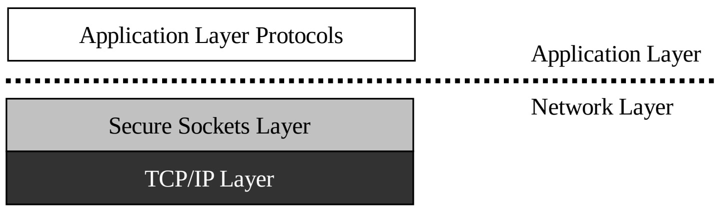

Figure 2. TLS runs above TCP/IP protocol and below application protocols

By using the TLS Protocol, we can achieve the following goals:

1. **Server authentication** allows a client to confirm a server’s identity, get and verify the server’s public key.

2. **Client authentication** allows a server to confirm a client’s identity, get and verify the client’s public key.

3. **An encrypted connection** requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality.

In addition, all data sent over an encrypted TLS connection is protected with a mechanism for detecting tampering–that is, for automatically determining whether the data has been altered in transit.

5 Secure Mathematical Computation Protocol (S+MCP)

IAMC clients and servers are connected by the Mathematical Computation Protocol (Section 2). In order to make IAMC framework secure, we need to extend MCP to be able to invoke TLS security service. The extension is done by adding a method, called `security-negotiation`, to the class of `Initialization`. Method `security-negotiation` is used to trigger a TLS negotiation between the client and server, it has three arguments:

- **Authentication:** This is a binary flag indicating the need of the authentication of the other party. For example, `"Authentication"="Yes"`.
- **Cipersuite Priority:** This argument gives the supported Ciphersuite list in the order of the party's preference. For example:
  `"Cipersuite Priority"={TLS_RSA_WITH_RC4_MD5,`
  `RSA_WITH_DES_CBC_SHA, TLS_DH_RSA_WITH_DES_CBC_SHA}`
- **Key Length:** This argument gives the key-lengths of the public key and the symmetric cipher that party prefers. For example,
  `"Key Length"=1024(public key),128(symmetric key).`

Cipersuite Priority in the S+MCP client's security-negotiation method offers a Cipersuite list which will be used by TLS client in its ClientHello message. Cipersuite Priority in the S+MCP server's security-negotiation method indicates the TLS server's choice of Cipersuite in the order of its preference which will be used by TLS server in its ServerHello message. The final key length of the public key or the symmetric cipher that will be used by TLS will be the maximum of the choices of S+MCP client and S+MCP server.

Note that standard TLS 1.0 fixes the key length for each Cipersuite[9], and there are 28 pre-defined Cipersuites. The 28 pre-defined Cipersuites include 12 exportable(out of US) Ciphersuites, with 512 the maximum of public key length and 40 the maximum of symmetric key length. In fact, public key of 512 bits and symmetric key of 40 bits are not secure, so when S+MCP works with TLS, we will not allow the use of exportable Ciphersuites.

When security-negotiation starts, a dialogue box is displayed for the end-user to choose among available Ciphersuites, the corresponding key lengths and authentication flags. The user input is passed to the client-side security-negotiation method. On the server side default arguments for S+MCP server's security-negotiation method are preset. A successful TLS negotiation sets up a secure channel, over which IAMC client and server can communicate securely.

During the initialization stage, the IAMC client and server begin the security negotiation immediately after the client-server connection. Figure 3 shows a typical initialization sequence. After the security-negotiation meth-

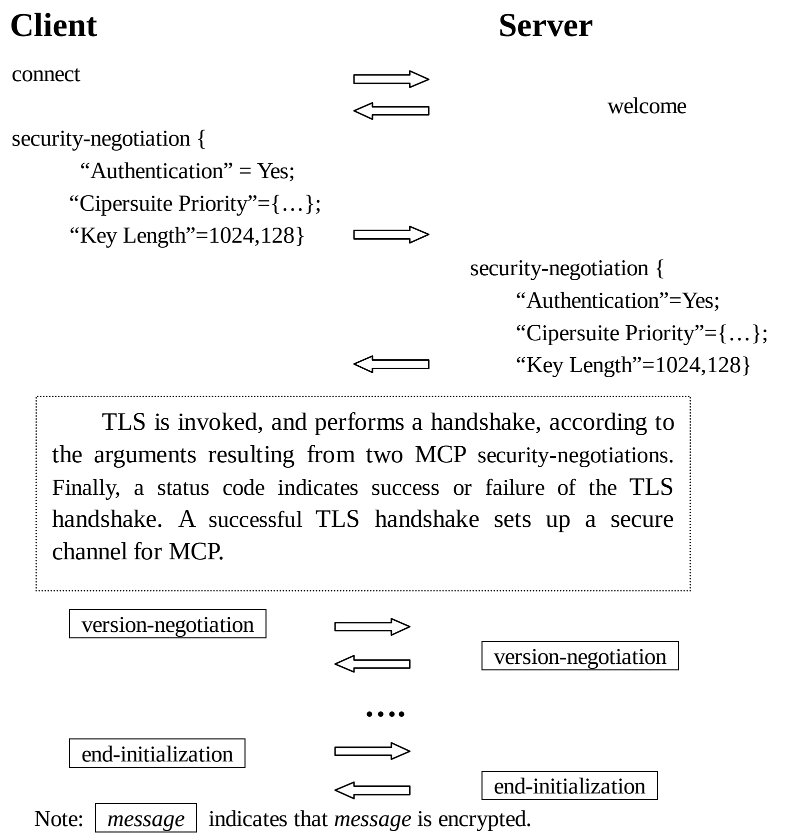
Client
Server
connect
welcome
security-negotiation {
"Authentication" = Yes;
"Cipersuite Priority"={...};
"Key Length"=1024,128}
security-negotiation {
"Authentication"=Yes;
"Cipersuite Priority"={...};
"Key Length"=1024,128}
TLS is invoked, and performs a handshake, according to the arguments resulting from two MCP security-negotiations. Finally, a status code indicates success or failure of the TLS handshake. A successful TLS handshake sets up a secure channel for MCP.
version-negotiation
version-negotiation
....
end-initialization
end-initialization
Note: message indicates that message is encrypted.

Figure 3. MCP initialization invoking a TLS handshake

ods, regular MCP messages can be sent over a secure channel.

6 How to use S+MCP to provide IAMC communication Security

In order to use secure MCP (S+MCP) with IAMC, IAMC clients and servers must first obtain their certificates from a Certificate Authority (CA). These certificates will be used for signing, for key exchange or for both. Notice that, for simplicity, here we have assumed that all certificates are from the same CA. Normal unsecure IAMC URLs are in the form

iamc://host:port/isv_name

identifying an IAMC sever on the Internet by giving its name, host and port.

When it is necessary for the client and server to authenticate each other or communicate over a secure channel, the letter `s` can be added to `iamc` to indicate the need to use S+MCP instead of MCP for the computation session.

iamcs://host:port/isv_name

The client and server will then use the S+MCP `security-negotiation` method to trigger a TLS negotiation, and establish a secure channel.

6.1 Three authentication modes

S+MCP supports three authentication modes: server and client both authenticated, server authenticated, and neither authenticated (total anonymity). End-user authentication can be thought of as the same as client authentication. Whenever the server is authenticated, the channel is secure against man-in-the-middle attacks, but total anonymous sessions are inherently vulnerable to such attacks. Anonymous servers cannot authenticate clients. Each party is responsible for verifying that the other's certificate is valid and has not expired or been revoked. Notice that the client authentication is important in practice, it is useful for servers that either charge a fee or provide highly secure computations.

6.2 Security considerations

For S+MCP to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors. The system is only as strong as the weakest key exchange and authentication algorithm supported, and only trustworthy cryptographic functions should be used. Short public keys, 40-bit symmetric encryption keys, and anonymous servers should be used with great caution. Implementations and users must be careful when deciding which certificates and certificate authorities are acceptable; a dishonest certificate authority can do tremendous damage.

7 Conclusions and Future Work

We incorporated the TLS in the MCP protocol to provide security for the IAMC framework. This is our first consideration to ensure secure communication between IAMC clients and servers. At present, we assumed that IAMC server and clients are within the same CA domain. Our future goal for security of IAMC is to enable IAMC clients to share mathematical computing powers supplied by servers in different domains. In such applications, clients and servers will hold certificates issued by different CAs, so inter-domain certification is inevitable.

We are also considering ways for a normal IAMC session to upgrade from MCP to S+MCP for security and to downgrade to open MCP for better transmission speed.

References

1. WANG, P. S. *Internet Accessible Mathematical Computation.* In the 3rd Asian Symp. on Computer Mathematics (ASCM'98), pp 1-13, Lanzhou Univ., China, 1998.
2. WANG, P. S. *Design and Protocol for Internet Accessible Mathematical Computation.* In. Proc. of ISSAC'99, pages 291–298. ACM Press, 1999.
3. LIAO, W. and WANG, P. S. *Dragonfly: A Java-based IAMC Client Prototype.* Lecture Notes on Computing Vol.8. (Proceedings of ASCM 2000.) World Scientific Press, pp. 281-290.
4. LIAO, W. and WANG, P. S. *Building IAMC: A Layered Approach.* Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'00). pp. 1509-1516.
5. WANG P. S., GRAY S., KAJLER N., LIN D., LIAO W. etc. *IAMC Architecture and Prototyping: A Progress Report.* Proc. of ISSAC'01, Univ. of Western Ontario, London, Ontario, Canada, July 22-25, 2001.
6. LIAO W., LIN D. and WANG P. S. *OMEI: An Open Mathematical Engine Interface. In Computer Mathematics.* Lecture Notes on Computing Vol.9. (Proceedings of ASCM 2001.) World Scientific Press, pp. 82-91.
7. SCHNEIER BRUCE, *Applied Cryptography, Second Edition: Protocols, algorithms, and source code in C*, John Wiley & Sons, Inc.
8. FRIER A., KARLTON P. and KOCHER P. *The SSL 3.0 Protocol*, Internet-Draft. Netscape Communications Corp.. Nov 18, 1996.
9. DIERKS, T. and ALLEN C. *The TLS Protocol Version 1.0.* RFC 2246. Jan. 1999.