IRJSE *Int. Res. J. of Science & Engineering,* 2020; Special Issue A7: 784-790 SJIF Impact Factor 6.68
ISSN: 2322-0015
RESEARCH ARTICLE **OPEN ACCESS**

Message Security in REST Web Services with Intermediary REST Relay Service.

Jivtode Manish

Department of Computer Science, Janata Mahavidyalaya, Chandrapur, MS, India.
Email: mljivtode@gmail.com

Manuscript Details
Available online on http://www.irjse.in
ISSN: 2322-0015

Cite this article as:
Jivtode Manish. Message Security in REST Web Services with Intermediary REST Relay Service., *Int. Res. Journal of Science & Engineering,* February 2020 | Special Issue A7:784-790.

ABSTRACT

Security of web services provides message integrity, privacy, authentication and confidentiality for message requests from client to service, and message responses from service to client. Intermediary relay web services with transport layer and message level security were used to test facts about point-to-point security and end-to-end security in REST web services respectively. In this paper, use high performance security encryption and decryption algorithms to measure the performance of algorithms in the message level security and tests were conducted on REST web service and client in an environment with intermediary REST relay web service.

Keywords: REST, Message security, REST relay service, HTTP, Request, Response, GET, POST, PUT, DELETE etc.

INTRODUCTION

Text REST web services currently have transport layer security. The main difference between transport layer and message level security is that message security includes any necessary credentials and claims along with the message itself. Contrast this with transport security, which uses handshaking or external resources (such as AD DS) to verify the credentials associated with a message.

A number of benefits are associated with using message security.

The biggest is the message is self-contained because it allows a number of scenarios that are not possible using transport security. Transport security secure messages from point to point only [1]. After the message has been received at the end of SSL tunnel, it is unencrypted by the transport channel itself. Message security provides end-to-end encryption. Even after a message has been received, it is still encrypted. Thus, message security keeps sensitive information encrypted if any intermediary relay web service is present until it reaches the final destination or the actual web service [2].

The reason for considering message security over transport security is the ability to provide multiple levels of security; different parts of the message are secured by using different encryption mechanisms [3]. Different sets of credentials are applied to encrypt different parts of the message. This enables a single message to have different audiences based on the credentials, or sending unencrypted information used by a router to deliver a message to the correct destination without compromising the security of other parts of the body.

Figure 1 shows transport level security architecture in REST web services where A is client, D is REST service, B and C is the transport channel or pipe. Sensitive data is protected within the channel or pipe only. When data passes the transport channel or pipe at B and C, it is decrypted automatically.

REPRESENTATIONAL STATE TRANSFER

REST is architecture for developing web services. REST architecture uses HTTP or similar protocols, by constraining the interface to a set of well-known standard operations (i.e., GET, PUT, POST, and DELETE for HTTP). REST architecture is designed to show how existing HTTP is enough to build a Web service and to show its scalability [4].

The main idea behind REST was to use well-developed HTTP for transferring data between machines, rather than using a protocol that works on top of the HTTP layer for message transfers. An application designed following REST principles would use HTTP to make calls between the machines, rather than relying on complex mechanisms like CORBA (Common Object Request Broker Architecture), RPC (Remote Procedure Call), or SOAP. Therefore, REST applications use HTTP request functions to post data, read data, and delete data, thus using the full functionality of HTTP Create, Read, Update, and Delete (CRUD) operations. REST can run on HTTPS, providing for the secured transmission of data.

The CRUD operations in conjunction with HTTP REST functions are shown –

The application of REST principles for web services requires HTTP protocols find it easy to understand and apply REST principles.

Some of the advantages of RESTful web services include:

i. **Light-weight:** RESTful web services directly utilize HTTP as the invocation protocol which avoids unnecessary XML markups or extra encapsulation for APIs and input/output. The response is the representation of the resource itself, and does not involve any extra encapsulation or envelopes [5]. As a result, RESTful web services are much easier to develop and consume than WSDL/SOAP web services. Additionally, they depends less on vendor software and mechanisms that implements the additional SOAP layer on top of HTTP. RESTful web services deliver better performance due to their light-weight nature.

ii. **Easy-accessibility:** URIs used for identifying RESTful web services are shared and passed around to any dedicated service clients or common purpose applications for reuse. The URIs and the representation of resources are self-descriptive and make RESTful web services easily accessible. RESTful web services have been widely used to build Web 2.0 applications and mashups.

iii. **Scalability:** The scalability of RESTful web services comes from its ability to naturally support caching and parallization / partitioning on URIs. The responses of GET (a side effect free operation) are cached exactly the same as web pages are currently cached in the proxies,

gateways and content delivery networks (CDNs). Additionally, RESTful web services provide a very simple and effective way to support load balancing based on URI partitioning. Compared to ad-hoc partitioning of functionalities behind the SOAP interfaces, URI-based partitioning is more generic and flexible, and could be easier to realize [6].

iv. **Declarative:** To imperative services from the perspective of operations, RESTful web services take a declarative approach and view the applications from the perspective of resources. Being declarative means that RESTful web services focus on the description of the resources themselves, rather than describing how the functions are performed. Declarative approach is considered to be a better choice to build flexible, scalable and loosely-coupled SOA systems.

The architecture of REST at the server side is as follows [7]:

- URL: Mandatory field to access web services running at the server.
- GET: All the methods of getting data from the server; the formats and interfaces the server supports for accessing the client details.
- POST: All the methods of adding details to the server; all the different interfaces and formats the server supports for adding data to its database.
- PUT: All the methods for updating the data at the serve; different types of interfaces and formats the server supports for adding data to the database.

DELETE: All the methods for deleting the data at the server. Different types of interfaces and formats the server supports for deleting data in the database.

MESSAGE SECURITY IN REST SERVICES

In intermediary scenario, the message itself is not protected once an intermediary reads from the wire and must be retransmitted to the ultimate receiver in out-of-band fashion, if necessary. This applies even if the entire route uses SSL security between individual hops [8].

Figure 2 shows message level security architecture in REST web services where A is client, D is REST service, B and C is the transport channel or pipe. Sensitive data is protected within and outside the channel or pipe also. When data passes the channel or pipe at B and C, it remains encrypted until it reaches its final destination [9].

INTERMEDIARY ARCHITECTURE

An intermediary is a component that lies between the client and the actual service. When the message is sent from the initial sender, it may pass through intermediate nodes before reaching its intended receiver. It basically intercepts the request from the client, routes it to the correct actual final web service. Similarly, it may intercept the response from the actual final web service and forward it to the client. Figure 3 shows an intermediary web service between the client and actual web service. It intercepts the client requests and forwards it to the actual web service.

In the above diagram, it is possible to combine intermediaries in several ways. In figure 3, a chain of intermediaries A and B intercepts the request from the client. Another intermediary C intercepts the response from the actual web service.

These intermediaries are increasingly getting recognized as the means to provide value added services like authentication, quality of service (QoS), auditing, management, aggregation [10] etc.

WHY INTERMEDIARY RELAY WEB SERVICE

i. Actual web service is the first directly web service exposed to the outside world attacks and to protect the actual web service from outside attacks.

ii. There may be a group of web services rather than only one web service.

iii. Intermediary relay web service used as a message route.

iv. There may be multiple organizations involved in the financial transactions like e-shopping

sites etc. This will require at least one web service per organization and hence a group of web services.

IMPLEMENTATION AND EVALUATION

Actual REST web service (C), REST client (A) and REST relay web service (B) are created in the cloud environment. REST web service (C) and REST clients (A) are normal HTTP/REST web service and clients respectively. A REST relay service is placed in the cloud environment as intermediary to REST service and client. Binding used is web Http Relay Binding. In this case, actual REST web service (C) is consumed by intermediary relay web service (B) and intermediary relay web service is consumed by REST client (A). Message level security is configured in A, B and C. Encryption and decryption algorithm was used in this experiment. With relay web service as intermediary, it is easy to examine the arrival and departure of sensitive data in it using a web debugging tool like Fiddler, Wire Shark or similar.

Message security protects data from point A to point C through point B, so arrival of data at intermediary is observed for encryption status. When encrypted data arrives at intermediary (B) it should remain encrypted until it reaches point C.

RESULTS AND DISCUSSION

The experiment result shows intermediary REST relay web service output where message security has been applied. It is observed that the sensitive data remains encrypted even as intermediary is passed or reads the data from the wire. Thus, sensitive data remains protected at intermediary.

This clearly indicates that message security provides end-to-end security and hence must be used in intermediary web services scenario.

Table 1: HTTP methods and CRUD action

CRUD operation	REST keywords (HTTP)
READ-read, retrieve	GET
CREATE-create or add new entries	POST
UPDATE-update or edit existing data	PUT
DELETE-delete existing data	DELETE

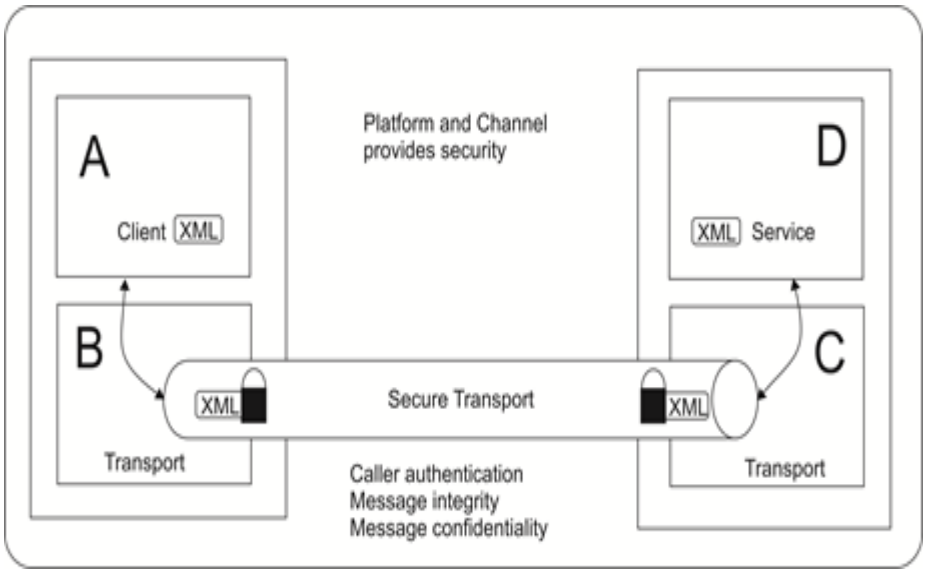


Figure 1: Transport Level Security in REST web services

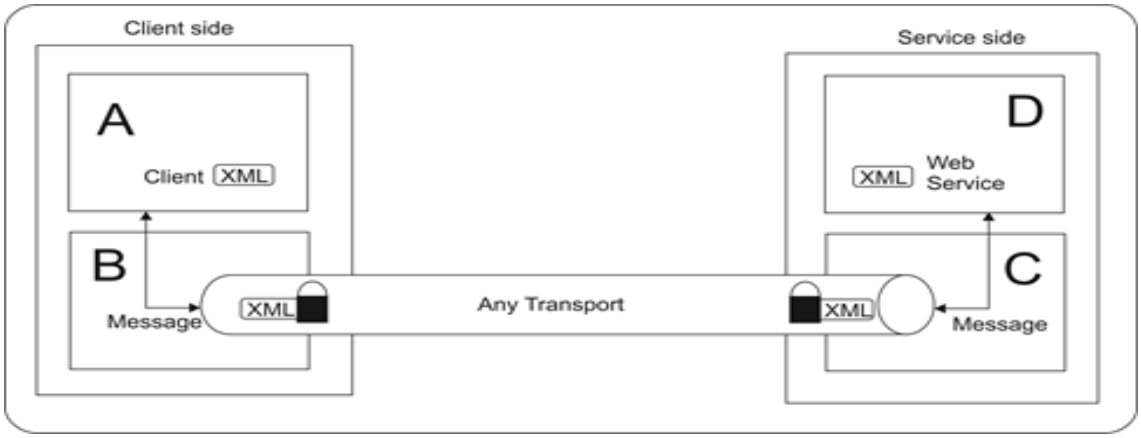


Figure 2: Message Layer Security in REST web services

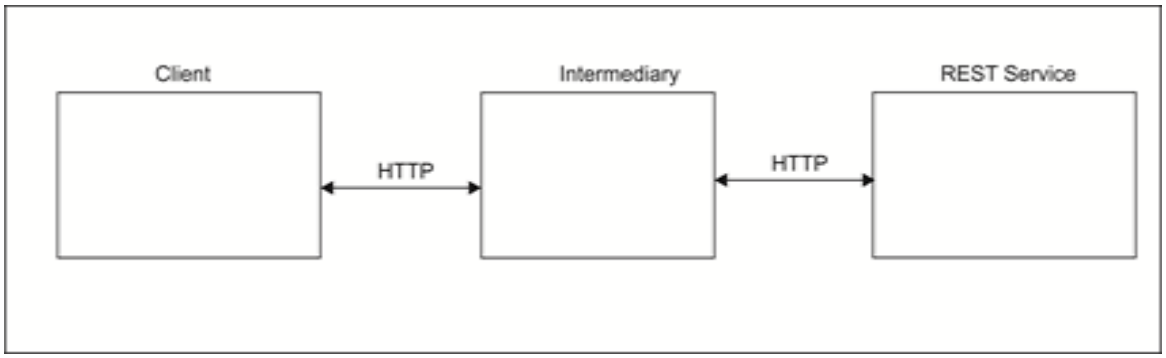


Figure 3: Intermediary between client and service during request

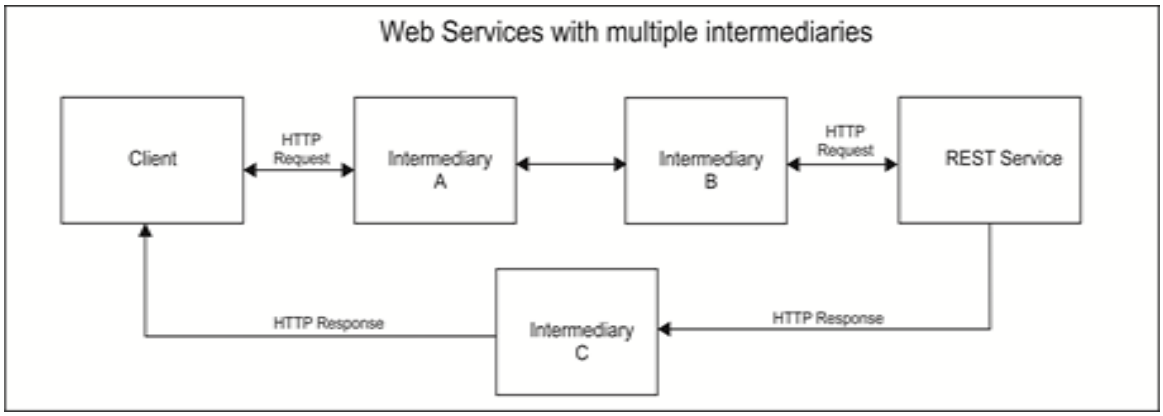


Figure 4: Intermediary between client and service during request and response

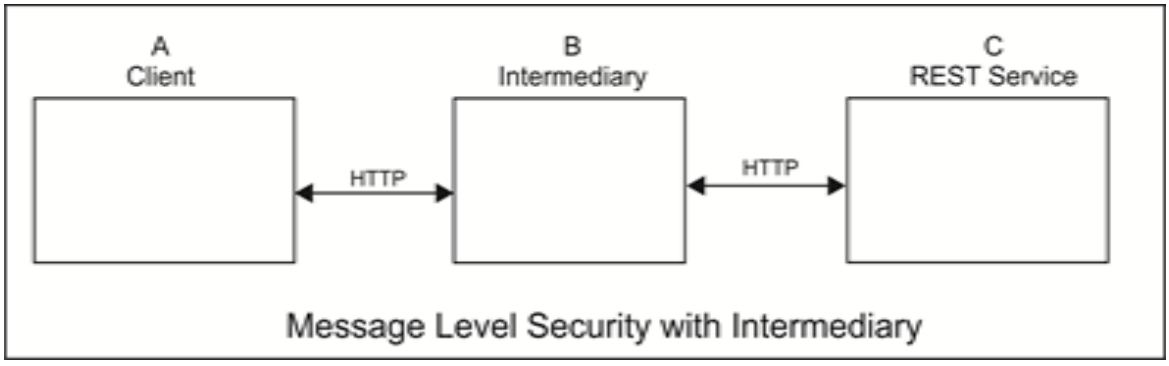


Figure 5: Message securities at REST web service with Intermediary

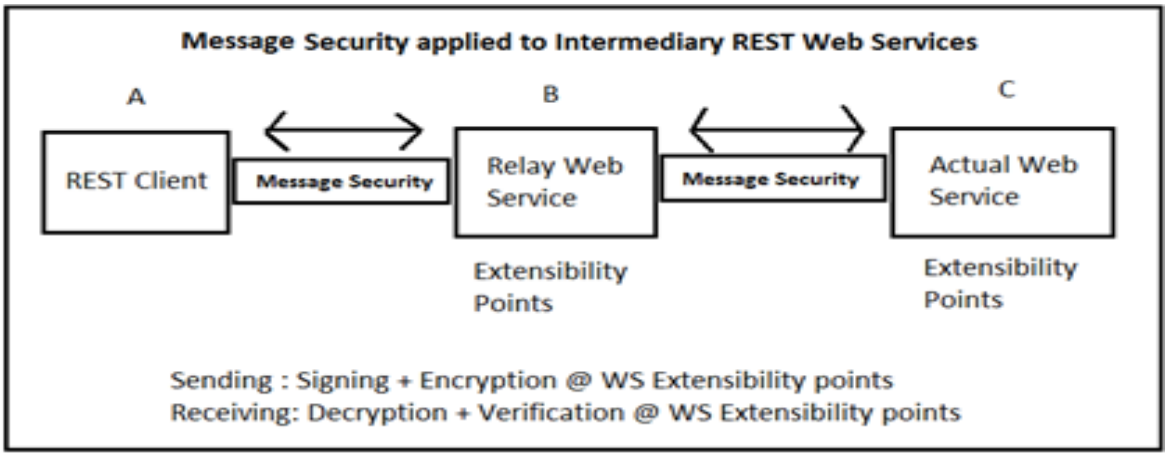
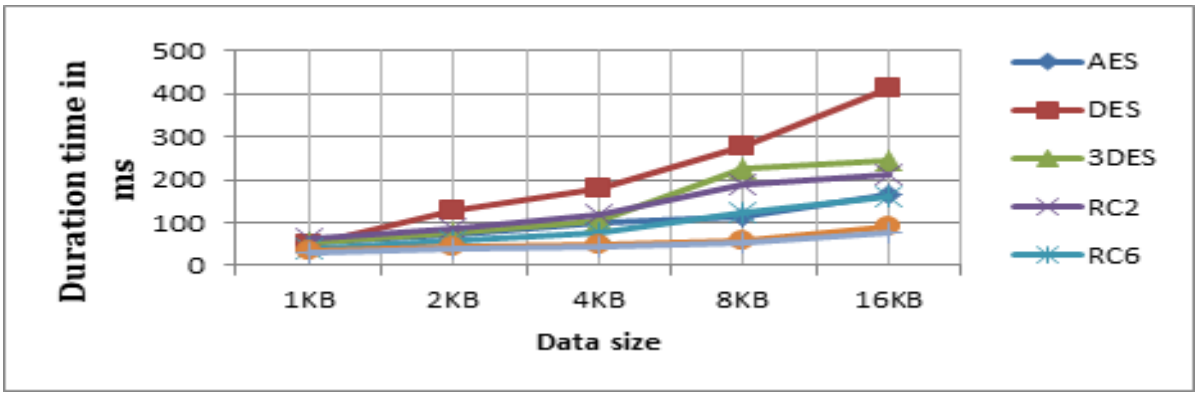


Figure 6: Message security applied to Intermediary Relay Web Service



Applications of message security may include enterprise service bus, router or relay web service as intermediaries where end-to-end security is essential. Screen shot taken at Intermediary for GET method

```
GET /account/101 HTTP/1.1
Accept: application/xml
X-ID: KHAGA Infotech
Host: khagainfotech.cloudapp.net

HTTP/1.1 200 OK
Server: IIS 7.0/10.0.0
Content-Type: application/xml
Content-Length: 137
<xml>
<account>
 s9d53o6OEoh7M0bvbmtGwdZE
 43b7FDibUjXvQdaL8UxVJhRQNzw7HhbLg5xp =
</account>
</xml>
```

The experimental results show the comparison of selected encryption algorithm for different data sets like text, image, audio and video and sizes, encryption/decryption speed, request/response time of various methods like GET, POST, PUT and DELETE.

CONCLUSION

It is concluded that, the performance measurement of encryption/decryption of various data sizes versus time for each algorithm in REST message security services for different operation using different algorithms.

The graph clearly indicate that as the data size increases from 1KB to 16KB, the encryption and decryption time also increases. The analysis of encryption/decryption of HPSEA performs better compared to others algorithms in terms of the request/response time. Thus, it indicates that data size is directly proportional to encryption/decryption time.

Hence, it can be proved, the transport layer security could not provide end-to-end security to sensitive data in intermediary web services environment. The message security provides end-to-end security in intermediary relay web service environment.

Conflicts of interest: The authors stated that no conflicts of interest.

REFERENCES

1. National Bureau of Standards. "Data Encryption Standard". FIPS Publication 46, 1977.
2. ANSI3.106. "American National Standard for Information Systems- Data Encryption Algorithm-modes of operation". American National Standards Institute, 1983.
3. William Stallings. "Network Security Essentials (Application and Standards)". Pearson Education, 2004.
4. J. Daemen, V. Rijmen. "Rijndael: The Advanced Encryption Standard". Dr. Dobb's Journal, March 2007.
5. S. Contini, R. L. Rivest, M. J. B. Robshaw and Y. L. Yin. " The Security of the RC6 Block Cipher". Version 1.0, August 20,2008.
6. Syed Zulkarnain Syed Idrus, Syed Alwee Aljunid, Salina Mohd Asi, Suhizaz Sudin and R. Badlishash Ahmad. " Performance Analysis of Encryption Algorithms Text Length size on web browsers". IJCSNS International Journal of Computer Science 20 and Network Security, Vol 8, No. 1, Page no 22-25,January 2008.
7. http://e-articles.info, Aug. 2008.
8. http://www.w3schools.com, Sep 2008.
9. Steve Resnick, Richard Crane and Chris Bowen. Essential Windows Communication Foundation for .NET Frame work 3.5, Addison- wesley, Microsoft .NET development series, Fe. 2013.
10. Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. "WSDL 1.1 Document". March 7, 2015.