International Journal For Technological Research In Engineering
Volume 4, Issue 12, August-2017 *ISSN (Online): 2347 - 4718*

AUTHENTICATION MANAGER WITH REST AUTHENTICATION INTERFACE PROFILE

Nagaveni Bhavi
Dept of CS&E, SJCE, Mysuru,

ABSTRACT: ***Authentication agents are software applications that securely pass user authentication requests to and from RSA Authentication Manager. Authentication agents are installed on each machine, such as a domain server, web server, or a personal computer, that you protect with Authentication Manager. For example, agent software residing on a web server intercepts all user requests for access to protected web pages. When a user attempts to access a protected URL, the agent requests the User ID and passcode and passes the User ID and passcode to the Authentication Manager for authentication. If the authentication is successful, the user is granted access to protected web pages. Adding the ability to allow users to authenticate into Authentication Manager using RSA securID 2FA and want to connect directly to customer's Authentication Manager Instance through a simple REST interface so that can start authenticating users. The Authentication REST Web Service will be deployed in AM, eliminating the need for standalone SDK all-together. The data exchanged will be in JSON format. Security is ensured by making the connections between REST Client and REST Service as SSL. This will eliminate the need to use CMS, Node Secret like in TCP, UDP SDKs.***

Keywords: authentication manager, agents, SDK, passcode, node secret.

I. INTRODUCTION

Overview of RSA SecurID

RSA SecurID provides an enterprise-wide authentication policy that protects most valuable applications, resources, and information of the organization [3].

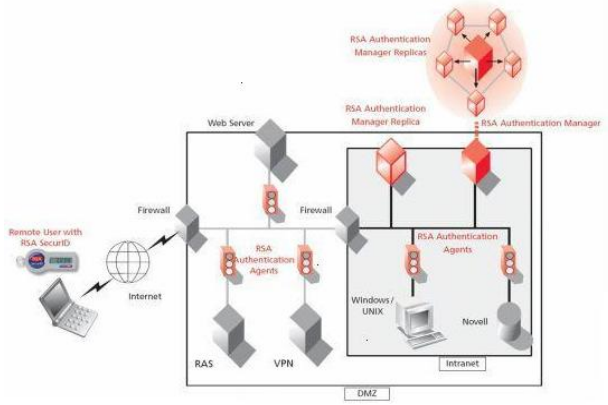

Figure 1.1: RSA SecurID overview

RSA SecurID consists of the following components as shown in figure 1.1

RSA Authentication Manager: The central two-factor authentication software that provides capabilities to manage security tokens, users, multiple applications, agents, and resources across physical sites. Authentication Manager verifies authentication requests and centrally administers authentication policies for enterprise networks.

RSA SecurID Software Authenticators: Provide software-based two-factor authentication security tokens to users on mobile smart phones, tablets, and PCs.

RSA SecurID Hardware Authenticators: Provide convenient, hardware-based two-factor authentication security tokens for your users.

RSA SecurID On-Demand Authenticator: Deliver a lightweight SMS or email two-factor authentication security token to a user anywhere in the world.

RSA SecurID Authentication Agents: Enable direct requests for two-factor authentication from key infrastructure to RSA Authentication Manager

RSA Authentication manager Architecture

In its simplest configuration, RSA authentication manager deployment consists of an authentication server and its associated database –"a primary instance"[4]. This basic structure processes requests from authentication agents and stores and manages users, agents and other system objects. Expanded a deployment can take advantage of multiple replica servers- sharing the function of authentication requests with each replica instance containing a copy of the deployment's database contents. Deployment can also take advantage of realm trust relationships to allow authentication from users existing in any deployment.

Primary instance

The authentication manager primary instance includes an embedded database. The overall structure of a replica instance is similar to primary instance. The replica server contains its own dedicated database synchronized to the primary.

Replica Instance

The replica instance provides additional load support for authentications. Transactions are temporarily stored in the database of the replica instance and are periodically sent to the primary. The primary then reconciles the database between itself and any other replica instances. Replicas also support disaster recovery by containing a near real time copy of the primary database contents. From these contents, a primary may be restored or a replica can be promoted to a primary. Authentication manager supports up to 15 replica instances. All replica instances do not provide read/write administration – they are read-only (Administrators can view the contents of a user account or view a report but cannot edit the objects)

 Copyright 2017.All rights reserved.

Authentication Agents

Authentication agents are installed at the point of user entry to a system and challenge a user for RSA SecurID credentials[3]. Agents are initially installed with the address of the primary server. Upon first contact with the primary, they receive information about other servers. Thereafter agents contact primary or replica servers as needed for authentication requests. Authentication agents are available for different platforms like windows agent, PAM agent for unix based systems and web agent for web server.

II. EXISTING SYSTEM

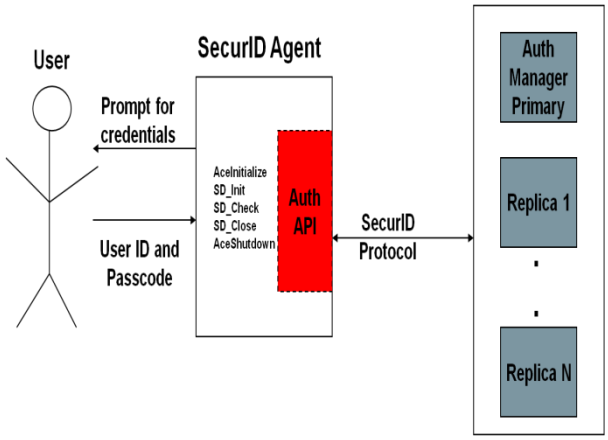

Figure 2.1 Client /server interaction through SecureID Agent with stand-alone SDK.

RSA Authentication Agents perform the following steps in a secure manner:

Intercept all access attempts, such as attempts to log on or access a URL.

Determine whether the specific requested resource is protected by RSA SecurID.

- If the requested resource is not protected, the Agent either ignores the request or, in the case of a custom Agent, takes whatever action is appropriate, such as writing an audit message in the UNIX syslog or the Windows Event Log.
- If the requested resource is protected by RSA SecurID, the Agent continues the authentication process.

Determine the user name of the person logging on, so that the RSA Authentication Manager can validate that the token code comes from the authentication device registered to that person.

Where RSA Authentication Manager Replicas are deployed, lock the user name to prevent replay attacks.

Request the user's passcode.

Combine the passcode with data known only to the Agent and its associated RSA Authentication Manager in the realm, and deliver the combined data to a Server for validation.

If an RSA Authentication Manager approves the request, the Agent permits access to the protected resource and takes any other appropriate actions.

If an RSA Authentication Manager denies access, the Agent prevents the user from logging on to the protected resource and takes any other appropriate actions.

III. PROPOSED SYSTEM

From the existing system we are going to remove authentication SDK. Deploying rest Authentication REST profile in AM, eliminating the need for standalone SDK all-together. The data exchanged will be in JSON format. Security is ensured by making the connections between REST Client and REST Service as SSL. This will eliminate the need to use CMS, Node Secret like in TCP, UDP SDKs. This product can be integrated with all other RSA products like via-access, Access manager etc. Clients who want to protect the resources can use the product. The resource may be file, device or web page.

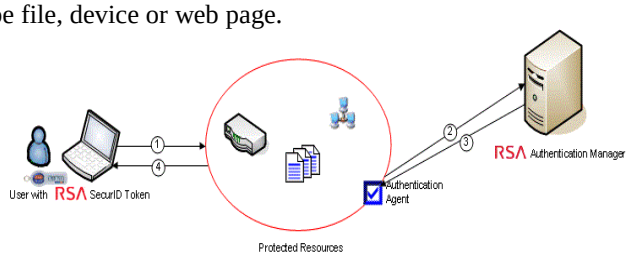

Figure 3.1 Basic architecture of agent/server model

IV. LITERATURE SURVEY

4.1 Single factor authentication

Network users often authenticate their login with a single authenticating factor- their network pass word[18]. The use of a password as an authentication mechanism presumes that the password is secret – known only to the user who is presenting it to the system.

However there is weakness to passwords – passwords stem from the fact that they may not always be secret. This occurs because passwords can be:

- Hacked or guessed
- Cracked
- Shared among users

4.2 Two factor authentication

To overcome the drawbacks of single factor authentication two factor authentication techniques uses two pieces of information for the purpose of authentication [3,18,19]. The two pieces of information are

- Something the user knows- PIN
- Something the user have – Token

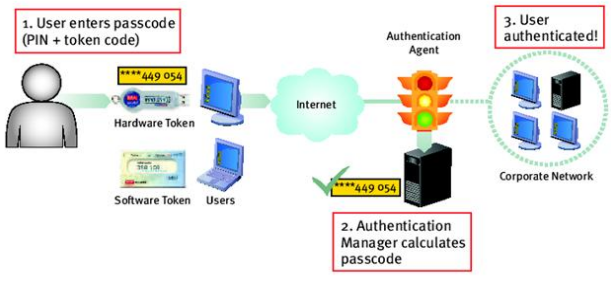

Figure 2.2 Process of two factor authentication

The process of two factor authentication shown in the figure 2.1 in which the user will provide passcode (PIN + token code) during Authentication. Then agent will send this information to authentication manager which will calculate the passcode and compares with one that was entered by user. When the passcodes match then user will be provided the access.

 Copyright 2017.All rights reserved.

4.3 RESTful Web services

Representational State Transfer (REST) has gained widespread acceptance across the Web as a simpler alternative to SOAP- and Web Services Description Language (WSDL)-based Web services. Key evidence of this shift in interface design is the adoption of REST by main stream Web 2.0 service providers—including Yahoo, Google, and Facebook—who have deprecated or passed on SOAP and WSDL-based interfaces in favor of an easier-to-use, resource-oriented model to expose their services. In this article, Alex Rodriguez introduces you to the basic principles of REST [5].

REST Web service follows four basic design principles:

- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer XML, JavaScript Object Notation (JSON), or both.

Stateful design

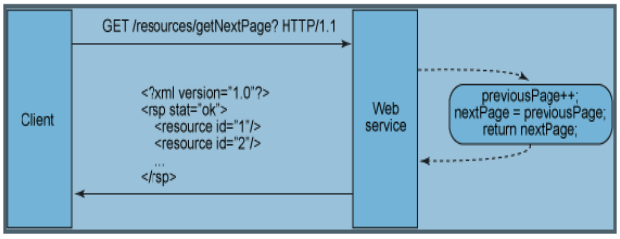

Figure 4.3.1 Stateful design

Stateful services like this get complicated. In a Java Platform, Enterprise Edition (Java EE) environment stateful services require a lot of up-front consideration to efficiently store and enable the synchronization of session data across a cluster of Java EE containers. In this type of environment, there's a problem familiar to servlet/JavaServer Pages (JSP) and Enterprise JavaBeans (EJB) developers who often struggle to find the root causes of java.io.NotSerializableException during session replication. Whether it's thrown by the servlet container during HttpSession replication or thrown by the EJB container during stateful EJB replication, it's a problem that can cost developers days in trying to pinpoint the one object that doesn't implement Serializable in a sometimes complex graph of objects that constitute the server's state. In addition, session synchronization adds overhead, which impacts server performance.

2.4.1.2 Stateless design

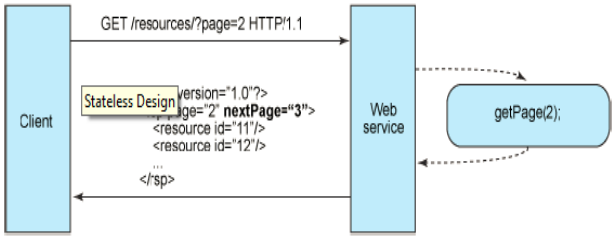

Figure 4.3.2 Stateless design

Stateless server-side components, on the other hand, are less complicated to design, write, and distribute across load-balanced servers. A stateless service not only performs better, it shifts most of the responsibility of maintaining state to the client application. In a RESTful Web service, the server is responsible for generating responses and for providing an interface that enables the client to maintain application state on its own. For example, in the request for a multipage result set, the client should include the actual page number to retrieve instead of simply asking for next

A stateless Web service generates a response that links to the next page number in the set and lets the client do what it needs to in order to keep this value around. This aspect of RESTful Web service design can be broken down into two sets of responsibilities as a high-level separation that clarifies just how a stateless service can be maintained:

Server

Generates responses that include links to other resources to allow applications to navigate between related resources. This type of response embeds links. Similarly, if the request is for a parent or container resource, then a typical RESTful response might also include links to the parent's children or subordinate resources so that these remain connected.

Generates responses that indicate whether they are cacheable or not to improve performance by reducing the number of requests for duplicate resources and by eliminating some requests entirely. The server does this by including a Cache-Control and Last-Modified (a date value) HTTP response header.

Client application

Uses the Cache-Control response header to determine whether to cache the resource (make a local copy of it) or not. The client also reads the Last-Modified response header and sends back the date value in an If-Modified-Since header to ask the server if the resource has changed. This is called Conditional GET, and the two headers go hand in hand in that the server's response is a standard 304 code (Not Modified) and omits the actual resource requested if it has not changed since that time. A 304 HTTP response code means the client can safely use a cached, local copy of the resource representation as the most up-to-date, in effect bypassing subsequent GET requests until the resource changes. Sends complete requests that can be serviced independently of other requests. This requires the client to make full use of HTTP headers as specified by the Web service interface and to send complete representations of resources in the request body. The client sends requests that make very few assumptions about prior requests, the existence of a session on the server, the server's ability to add context to a request, or about application state that is kept in between requests. This collaboration between client application and service is essential to being stateless in a RESTful Web service. It improves performance by saving bandwidth and minimizing server-side application state.

V. SYSTEM DESIGN

Authentication Process Flow

The following diagram shows the general flow of the interface during authentication.

 Copyright 2017.All rights reserved.

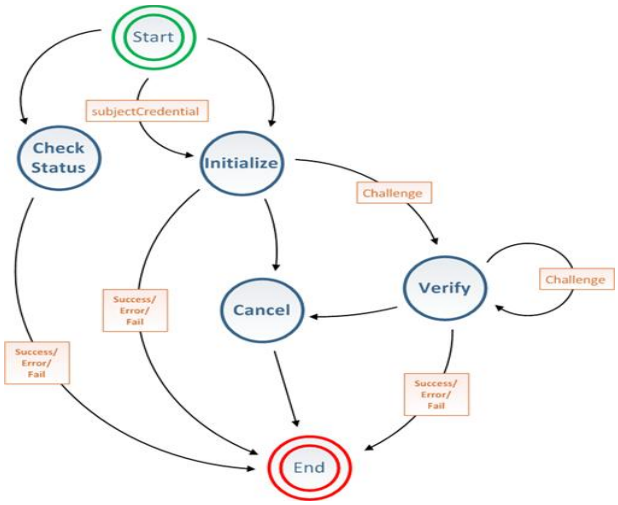

Figure 5.1 General flow of the interface during authentication

The following steps describe at a high level the client-server process flow during authentication. These steps apply to both the RSA Authentication Manager server and the Cloud Authentication Service.

- The client calls the Initialize interface. Calls to the Authentication Manager server use the user login ID (subjectName). Calls to the Cloud Authentication Service use email or the SecurID Username specified in the Identity Source configuration in the Cloud Administration Console. For Active Directory, the default is sAMAccountName. For other LDAP vendors, this is a vendor-specific value.
- The Initialize interface responds with the challenge method options and requirements for the user to complete authentication.
- The client collects challenge method credentials from the user and sends them to the server in a Verify interface call.
- The Verify interface responds in one of two ways:
  - If the user's authentication is complete and successful, the user is granted access to the requested resource.
  - If authentication is not complete, the Verify interface sends the additional challenge requirements the user needs to complete authentication.

VI. MODULE DESIGNS

6.1 System

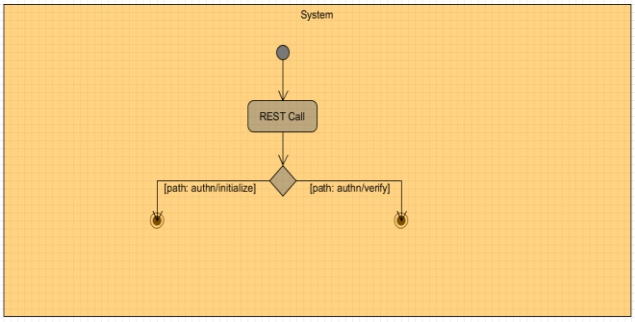

Figure 6.1 Calls to server

The first call to the server must be the initialize call. This starts the authentication process.
If it is not first call then it will verify the credentials provided by the client.

6.2 Initialize

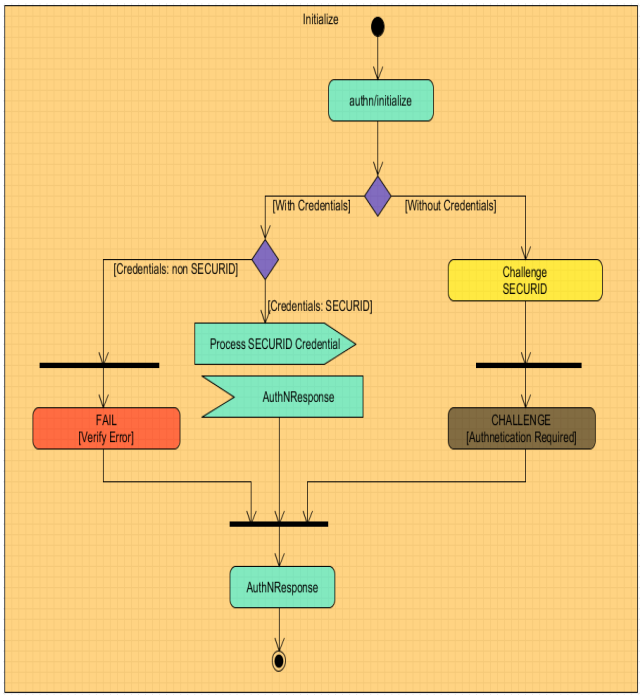

Figure 6.2 Initialize process

Call to initialize may be
Without credentials
With credentials

6.3 Process SecurID credential

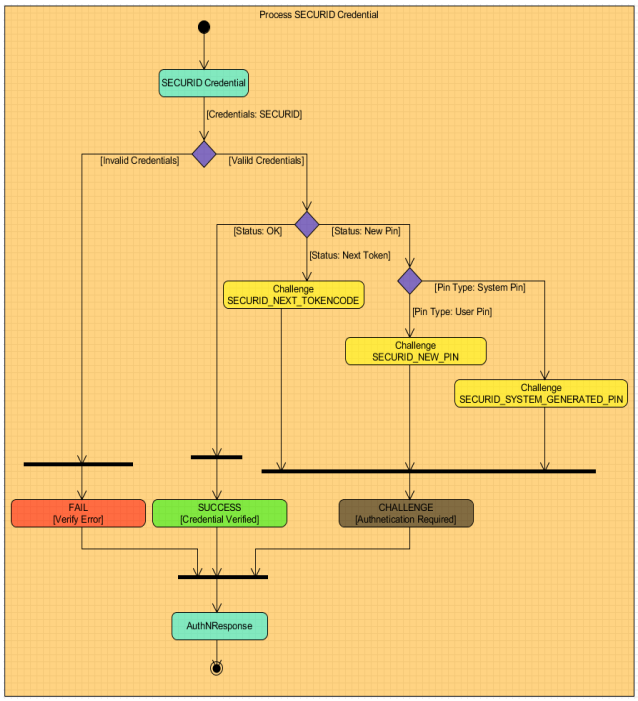

Figure 6.3 SECURID credential process

 Copyright 2017.All rights reserved.

For valid credentials if

- Status is OK then credential verified will be success.
- Status is new pin then it will ask for system pin or user pin and they are put for the challenge for authentication.
- Status is next pin then it will put for the challenge still authentication is required.

6.4 Verify

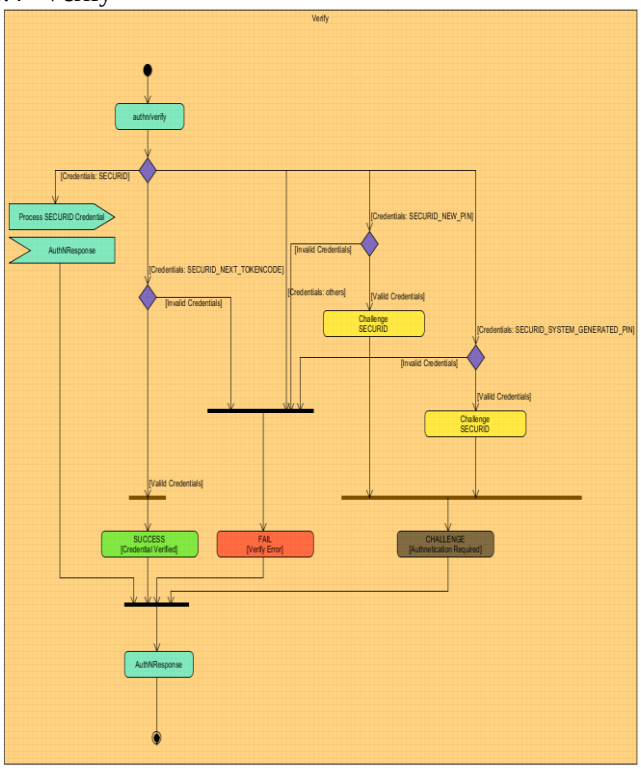

Figure 6.4 Verify process

Call to the verify will verify all the entered credentials for the pericular client and sends back appropriate status as response.

VII. RESULTS

Initialize Request

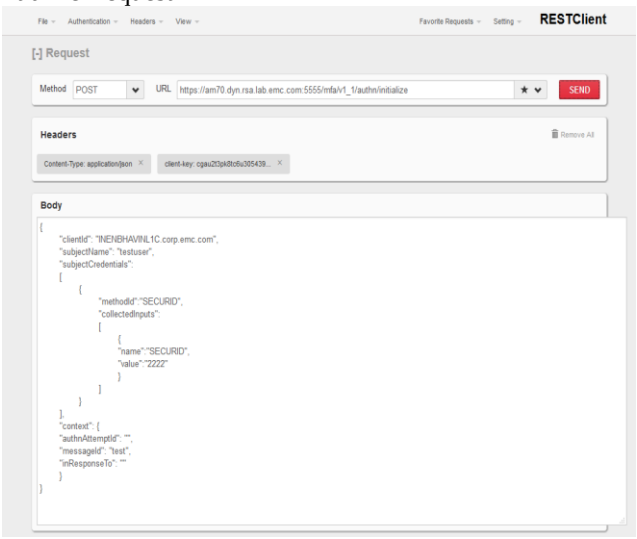

Figure 7.1 Initialize request for SecurID with credentials

Initialize Response (Response Headers)

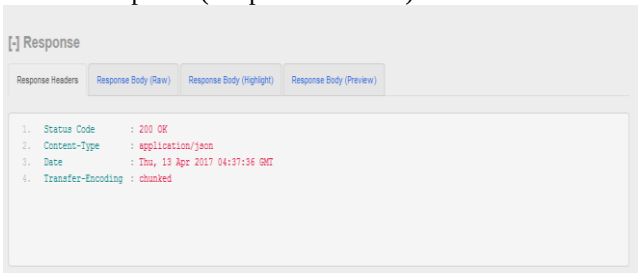

Figure 7.2 Initialize response(response header) for SecurID with credentials

Initialize Response (Response Body)

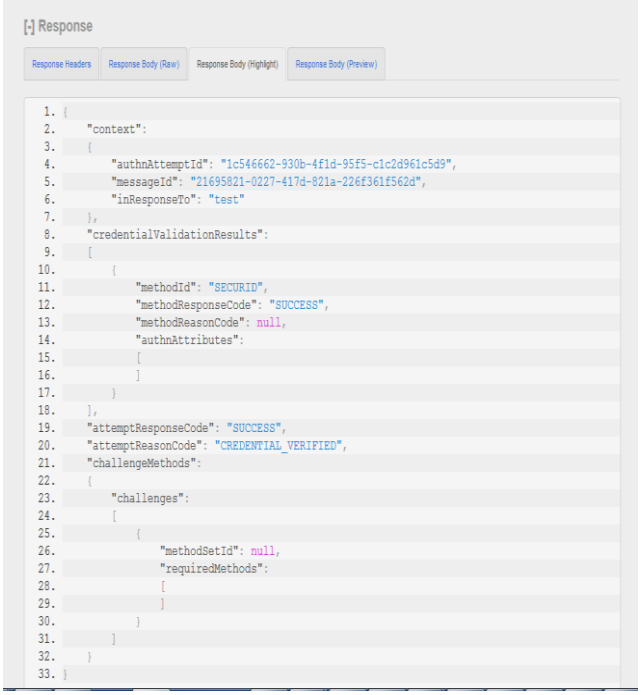

Figure 7.3 Initialize response(response body) for SecurID without credentials

Verify Request

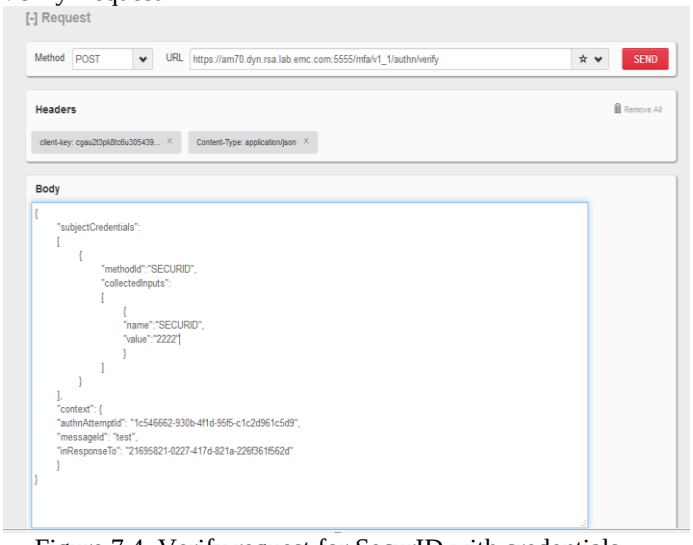

Figure 7.4 Verify request for SecurID with credentials

 Copyright 2017.All rights reserved.

Status

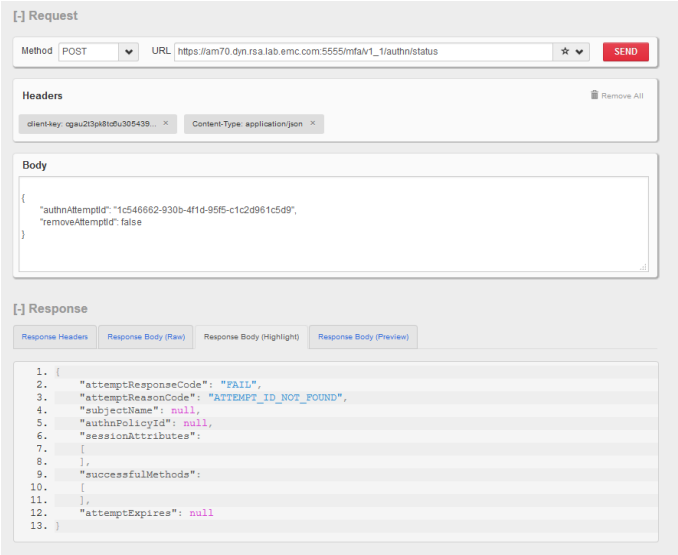

Figure 7.6 status for SecurID with credentials

VIII. CONCLUSION AND FUTURE WORK

From the existing system we are going to remove authentication SDK. Deploying rest Authentication REST profile in AM, eliminating the need for standalone SDK all-together. The data exchanged will be in JSON format. Security is ensured by making the connections between REST Client and REST Service as SSL. This will eliminate the need to use CMS, Node Secret like in TCP, UDP SDKs. This product can be integrated with all other RSA products like via-access, Access manager etc. Clients who want to protect the resources can use the product. The resource may be file, device or web page. Here as we consider two factor authentication for future work it can be extend to multifactor authentication which includes biometrics as one factor along with static password and passcode generated by SecurID token.

REFERENCES

[1] http://searchsecurity.techtarget.com/definition/multi factor-authentication-MFA
[2] https://wiki.na.rsa.net/display/AUTHA/Authenticati on+Agents+-+New+Joinee+Ramp+Up
[3] https://wiki.na.rsa.net/display/~balaks5/Authenticati on+Agents
[4] https://wiki.na.rsa.net/display/AM8X/AM+8.x+-+REST+Authentication+API
[5] http://mariusbancila.ro/blog/2013/08/19/full-fledged-client-server-example-with-cpprest-sdk-110/
[6] https://wiki.na.rsa.net/display/SIDASE/Authenticati on+Manager.
[7] McAfee Case Study "Securing the Cloud with Strong Two-Factor Authentication through McAfee One Time Password" http://www.mcafee.com/in/case-studies/cs-cloudalize.aspx.
[8] http://www.oneid.com/wp-content/uploads/2014/05/OneID_WhitePaper_Adv-of-Integrated-2FA-final.pdf.
[9] Dinei Florencio, Cormac Herley " A Large-Scale Study of Web Password Habits" Proceedings of the 16th international conference on the World Wide Web, ACM Digital Library, pp 657-666, 2007.
[10] Andrew Kemshall, Phil Undewood "White paper - Options for Two Factor Authentication" SecurEnvoy July 2007.
[11] Ziqing Mao, Dinei Florencio, and Cormac Herley "Painless Migration from Passwords to Two Factor Authentication" in 'WIFS' , IEEE, Brazil, pp. 1-6, Nov 29th-Dec 2nd, 2011.
[12] Manav Singhal and Shashikala Tapaswi "Software Tokens Based Two Factor Authentication Scheme" International Journal of Information and Electronics Engineering, Vol. 2, No. 3, pp. 383 - 386, May 2012.
[13] Olufemi Sunday Adeoye "Evaluating the Performance of two-factor authentication solution in the Banking Sector" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 2, July 2015.
[14] Goode intelligence "Two Factor Authentication Goes Mobile" www.goodeintelligence.com, September 2012.
[15] Sharifah Mumtazah Syed Ahmad, et al "Technical Issues and Challenges of Biometric Applications as access control tools of Information Security" International Journal of Innovative Computing, Information and Control Vol8, No. 11, pp 7983 - 7999 Nov 2015.
[16] Sans Securing the Human "Two Factor Authentication" the monthly Security awareness news letter for computer users November 2015
[17] Haichang Gao, Wei Jia, Fei Ye, Licheng Ma "A survey on the use of Graphical Passwords in Security", Journal of software, Vol. 8, No. 7, July 2013.
[18] Rahul Kale, Neha Gore, Kavita, Nilesh Jadhav, Swapnil Shinde " Review Paper on Two Factor Authentication Using Mobile Phone" International Journal of Innovative research and Studies, Vol. 2, Issue 5, pp. 164 - 170, May 2015.
[19] Alexandra Dmitrienko, Christopher Liebchen, Christian Rossow, and Ahmad-Reza Sadeghi "On the (In) Security of Mobile Two-Factor Authentication" Lecture Notes in Computer Science, pp. 365-383, Nov 2015.
[20] S. Vaithyasubramanian, A. Christy "A practice to create user friendly secured password using CFG" International Conference on Mathematics and Engineering Sciences, Chitkara University, Punjab, p. 39, March 2015.
[21] S. Vaithyasubramanian, A. Christy, D. Lalitha "Generation of Array Passwords Using Petri Net for Effective Network and Information Security" Advances in Intelligent Systems and Computing, Springer India, Vol.1, pp. 189 - 200, July 2016.
[22] S. Vaithyasubramanian, A. Christy "A Scheme to

 Copyright 2017.All rights reserved.

Create Secured Random Password Using Markov Chain" Advances in Intelligent Systems and Computing, Springer India, Vol. 325, pp. 809-814, 2016.

[23] S. Vaithyasubramanian, A. Christy, D. Lalitha "Two factor Authentication for Secured Login Using Array Password Engender by Petri net" Accepted for Procedia Computer Science, Elsevier 2016.

 Copyright 2017.All rights reserved.