Authentication

via HTTP

In RESTful client-server communications services the sessions state for the RESTful setup is stateless. Therefore, the client must provude all information necessary to make requests and confirm authentication.

## **Basic Authentication**

**Basic authentication in HTTP simply uses a special header called ‘Authorization:’ which includes a username and password ‘username:password’ encoded (not encrypted) in base64. Since credentials are not encrpyted and the data is sent in plane HTTP, it is very insecure. Therefore, basic HTTP authentication should only be used over a secure connection such as SSL/TLS.**

**Basic authentication also requires password to be sent on every request and it does not safeguard agasint head and body tampering.**

## **HMAC**

Hash Based Message Authentication (HMAC) aims to avoid the downsides of basic authentication by sending a hashed version of the secret together with more information.

Example:

User John Doe wants to access his finacial records:

* Concatenate the GET request to fetch all the information required as a string, it is also common to add additional data to the end such as current date and a nonce (once time random expression) to make each request unique stopping eavesdroppers from using it again.

“GET+/users/johndoe/financialrecords+20apr201312:59:24+123456”

* Next a HMAC digest is created with the secret and concatenated request, and encrypted using sha256 or other standard:

digest = base64encode(hmac("sha256", "secret", "GET+/users/johndoe/fanan

cialrecords+20apr201312:59:24+123456"))

* This is then joined with user and encoded with base64, then sent via the HTTP authentication header, just like basic authentication request:

GET /users/johndoe/financialrecords HTTP/1.1

Host: example.org

Authentication: hmac johndoe:[digest]

* Server now knows the username, the HMAC digest, and the resource requested. Since the sever has the secret it can also create the HMAC digest. Any eavesdropper won’t be able to recreate the digest since it contains a secret they don’t know, and they can use it for anything else since it is unique to the request, date, and nonce. The server can also check the date is within a set range of the servers current time and that the nonce has not be used before since it can only be used once.

So the process is:

* Create encrypted digest from request, secret, date, and nonce
* Send to server to provide authentication

However, once secrets become known, this method become unsecure. Therefore, it is common to use temporary tokens.