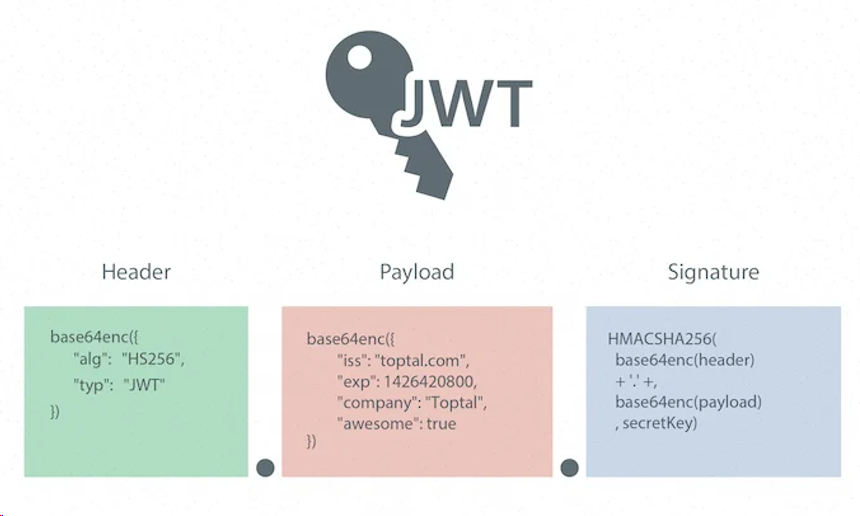
Research Paper Summaries

JSON WEB TOKENS

Ammar Khan 17010020010 BSCS-10



JSON Web Token (JWT) is an open standard (RFC-7519) that provides a mechanism for securely sending and receiving JSON objects. The data at exchanged at any point in time can be verified as it is digitally signed using a secret key with “Hashed Message Authentication Code” algorithm.

Instead of creating sessions, on logging-in, the system generates a JSON Web Token that you can store on client side as an authorization key to access the service. The token is embedded in the header sent from client-side which makes a claim that he is authorized to avail this service.

**Reasons to consider JWT:**

1. It provides mechanism to make your web servers / application servers / APIs to be completely stateless.
2. You can offload authentication to third-party and only worry about things you need to work on.
3. External Authentication server can be completely separate from running Application servers/ Web servers/ APIs and doesn’t have to share any secret key with other elements of the network.
4. No secret key installed on system protected by it.

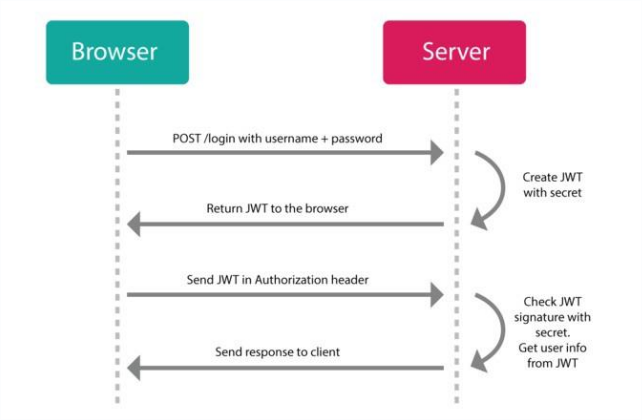
# Stateless Authentication with JSON Web Tokens using RSA-512 Algorithm (May, 2019)

In modern era, the way to go is to break down monoliths into microservices so that each component can be scaled appropriately. But, to make the service stateless is a hard task. Most of the microservices uses REST APIs as a mechanism to communicate with the client. But, to secure this API, you have to store session/cookies for ongoing communication. JWT helps with that by not only offloading authentication/authorization, but by embedding the tokens into client local storage.

JSON web tokens consist of 3 parts namely:

1. Header
2. Payload
3. Signature

Signature is encrypted using HMAC or RSA. In this paper, the working of JWT with RSA-512 is described as

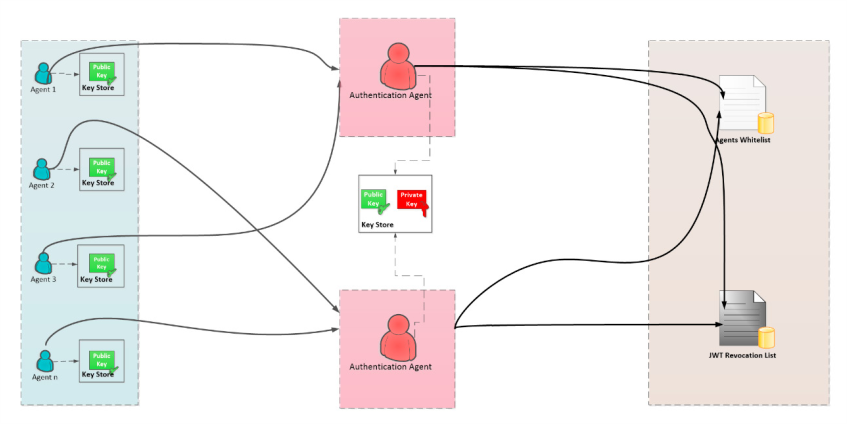


JWT header contains the type of token and the algorithm used to encrypt it. The testing done in the research paper concludes that JWT with RSA-512 is 24.69% more faster than JWT- SOAP.

## Authentication and Load Balancing Scheme based on JSON Token for Multi-Agent Systems (2018)

This research papers provides insights on how to provide load-balancing and security across distributed system behind load-balancers so that the communication is encrypted and stateless as well using asymmetric cryptographic technology. Using JWT, access tokens can be used for validating of subsequent client requests without making frequent calls to resource server or database. Access tokens can have limited validity by adding TTL to it. Access related claims can be embedded as a part of payload as well.

The research paper proposed this model:



Generation of JWT is as follow:

1. Agent X requests the Authentication Agent to generate a JWT for it to communicate with Agent Y.
2. The Authentication Agent verifies if Agent X belongs to whitelist by checking if his AID is registered in this list.
3. The Whitelist returns the response to the Authentication Agent.
4. If Agent X belongs to the whitelist, then the Authentication Agent generates a JWT for Agent X using its private key based on the following equation.

**f ( base 64 Encode )( header . payload . signature )**

1. Else the authentication agent returns a message to Agent X , rejected.
2. The Authentication Agent sends the generated JWT to Agent X. The generated JWT contains the URL to retrieve the public key that must be used for verification of all JWT’s generated by the Authentication Agent.
3. Agent X retrieves the JWT, then the public key via the URL indicated in this JWT to stores it in own key-store
4. Agent X tries to prove his identity to Agent Y by inserting the recovered JWT into the send message
5. Agent Y checks the validity of the JWT contained in the Agent X request.
6. Agent Y after checking the validity of JWT checks if the JWT is not registered in the JRL.
7. At this level, the Agent Y is sure that the message has not been tampered with and JWT is valid. To ensure authentication and non-repudiation, Agent Y can retrieve the Agent AID contained in the JWT and then compare it with AID of Agent X, and if both AIDs are the same, that means the JWT belongs to Agent X.

## An Analysis on the revoking mechanisms for JSON Web Tokens (2018)

This research paper highlights the JWT overhead when a client, issued with JWT, leaves the system. The invalidation process creates a lot of overhead to revoke the token. So, the proposed procedure provides a mechanism to revoke the token with little overhead without compromising the strong security constrains of JWT. The problem arises from the decentralization of the Authentication procedure and keeping it stateless, which is also its strength.

**The Logout Problem:**

One of the main problems with JWT is that it cannot be easily revoked at will, as token validity is not explicitly stored in the server. This makes invalidation much harder with JWT than in case of a traditional token-based solution. The source of the problem is that the validity of a token us determined by the contents and ability of the server to decrypt and unpack it. This means that distinguishing between a valid and invalid token is only possible based on the contextual formation and the data stored in the token itself.

**Introducing Short-lived Tokens:**To solve the token invocation, the paper proposes a TTL to be assigned with the JWT token itself and stop providing new tokens to logged out users by invalidating their token acquisition method.

**Changing the JWT Secret:**

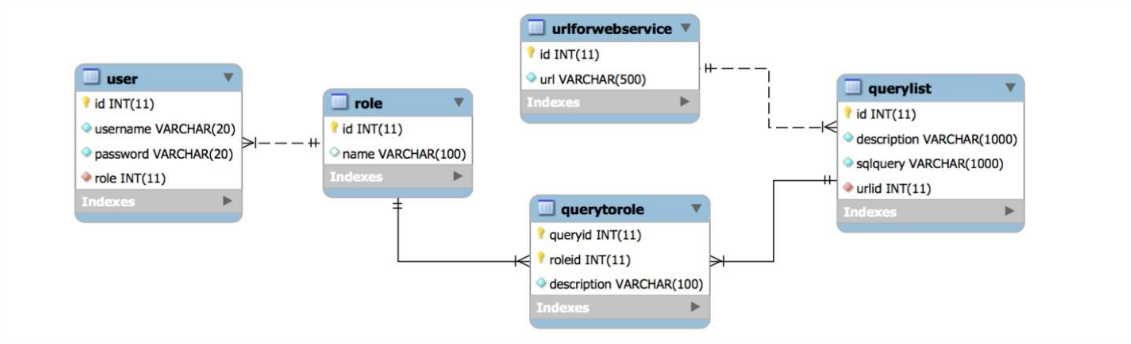
By changing JWT secret itself, invalidating every encryption token using old secret at the same time. The main advantage of this approach is that it does not require a centralized data storage and invalidation is instantons.

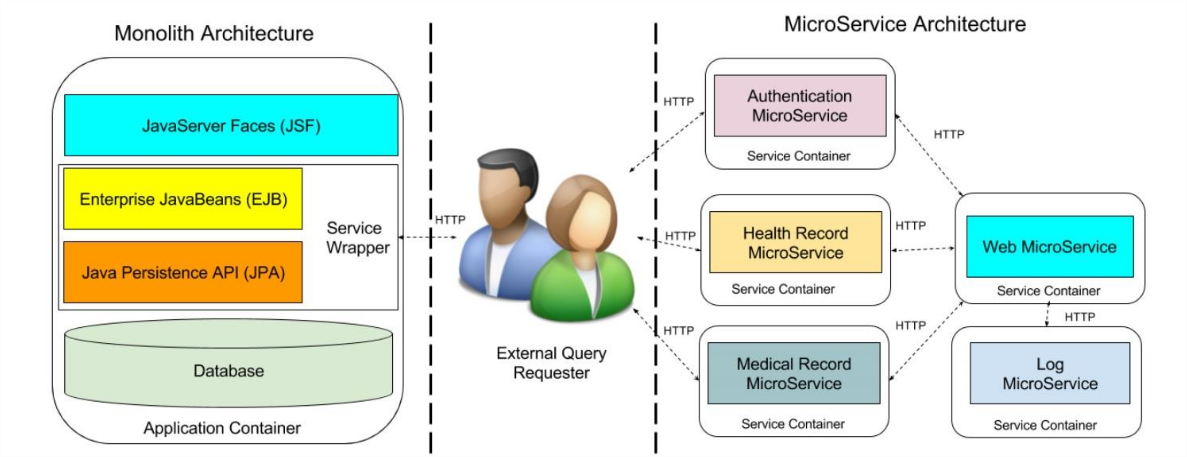
**Proposed Model:**

1. User Authorization and access service for storing and providing user authorization data, issuing new tokens of both types 1) access tokens 2) refresh tokens
2. Refresh token store. A data storage component accessible by the UAA service, used to store refresh tokens by the associated client identifiers. Typically, a key-value storage in implementation
3. Secured Service. A resource on which access control is required. A client can only access this service if they have a valid token, granting them this privilege. The token is unpacked using the JWT secret and access to this service is abed on its content.

## MicroShare: Privacy-Preserved Medical Resource Sharing Through Microservice Architecture (2018)

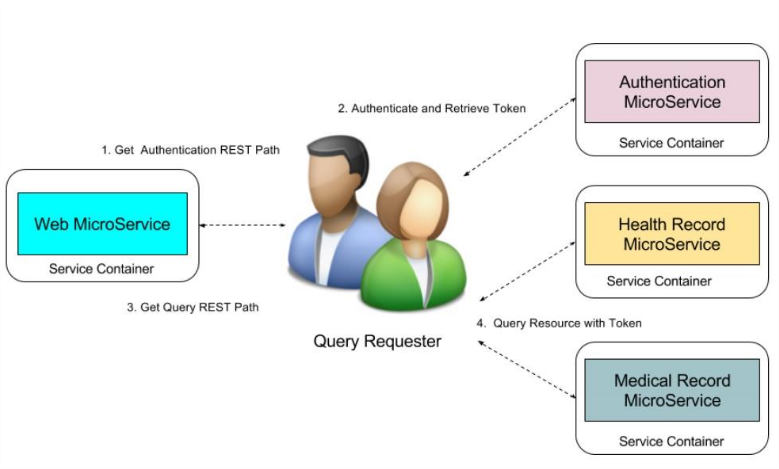
This paper presents JWT with role-based authentication to send and receive patient’s private medical data between multiple user-agents over RESTFul web service. The model proposed implements role-based access control. A Medical dataset, operations might include insert, delete, append and update. The data model of RBAC is based on five data types: users, roles, objects, permissions and executable operations by users on objects. A sixth data type, session is used to associate roles temporarily to users. A role considered a permanent position in an organization whereas a given user can be switched with another user for that role. Thus, rights are offered to roles instead of users. Roles are assigned to permissions tat can later be exercised by users playing these roles.





Role-based access control mechanism to display different queries to different level requesters such as administrators, researchers and organizations. But, RESTFul resource is navigated by exposed URI by API. JWT comes in action by guarding these URIs with authorization tokens. User must request authentication from the authentication JWT service to get its token, the token then provided entails the validation for the URI you want to use.

**Proposed JWT Role-Based System:**

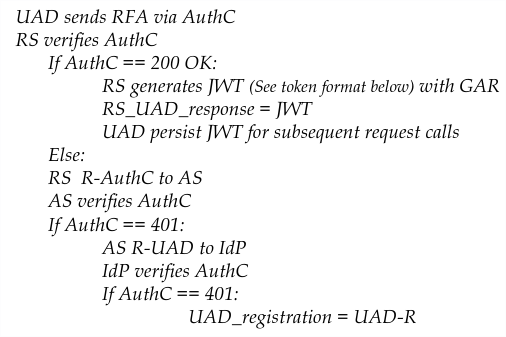


## A JSON Token-Based Authentication and Access Management Scheme for Cloud SaaS Applications (2018)

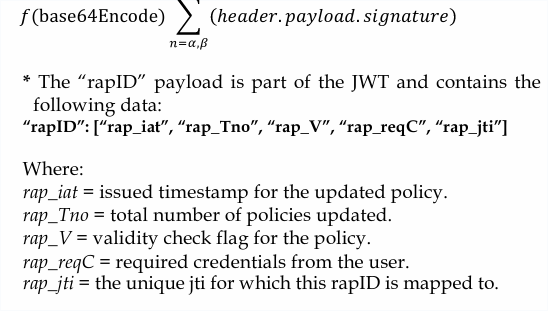
This paper discusses the Cloud Computing industry and its vast infrastructure which is distributed across multiple networks and proposes a solution which goes on with the elastic nature of Cloud SaaS applications. JWT for client authentication and session method is used to secure communication across fleets of software appliances. For authorization, the framework employs Policy Match Gate and Policy Activity Monitor with a subcomponent of Policy Validation Unit and Policy Proxy DB to optimized and secure service delivery. JWT token structure is encoded in a compact JSON serialization format using base64URL consisting of JSON Web Signature and JSON Web Encryption. Both serializations use different keys for signature and encryption. JSON Web Key and JSON Web Algorithm are cryptographically embedded in each JWT. Its seamless compatibility with X.509 key certificate makes it able to carry more information. The identity model uses Security Assertion Markup Language and 0Auth protocols. Both the components are digitally signed by JWT to ensure validity of the incoming requests. The system uses JWT and NameCoin, from block chain, for decentralized identity management framework for personal cloud. NameCoin is used for sharing public key.

1. Every access call goes to Policy Match Gate and each event or process is monitored by a Policy Activity Monitor which ensures that each request has the appropriate Resource Access Policy, even though the token is still valid.
2. Policy Control Module synchronizes calls to policy DB and policy validation module. As a Control and admin unit, it interfaces with Policy Access Module to determine if the policy update is required.
3. Policy Validation module checks if any policy is updated. If updated, generates a JWT token with appropriate accesses to the resources.

Algorithm for Authentication:



JWT Token Algorithm with above modules attached:



## Tools for testing encryption protocols strength:

1. THC-Hydra (Brute-Force Attacks)
2. JWT Tool (Public-Private key catch, and token validation)
3. John The Ripper (HMAC hash crack)
4. TrueCrack (RSA-512 and SHA-256 crack)