Xomorod.com Web API Authentication State Diagram

Sunday, January 31, 2016 11:09 PM

Encryption

There are two main types of encryption: symmetric key (also known as shared secret key) and asymmetric key (also known as public key or public-private key). SSL/TLS uses both symmetric key and asymmetric key encryption.

- **Symmetric Key**. In symmetric key encryption, the same key is used to encrypt and decrypt the message. If two parties want to exchange encrypted messages securely, they must both possess a copy of the same symmetric key. Symmetric key cryptography is often used for encrypting large amounts of data because it is computationally faster than asymmetric cryptography. Typical algorithms include DES (Data Encryption Standard), 3-DES (Triple DES), RC2, RC4, and AES (Advanced Encryption Standard).

- **Asymmetric Key**. Asymmetric or public key encryption uses a pair of keys that have been derived together through a complex mathematical process. One of the keys is made public, typically by asking a CA to publish the public key in a certificate for the certificate-holder (also called the subject). The private key is kept secret by the subject and never revealed to anyone. The keys work together, with one being used to perform the inverse operation of the other: If the public key is used to encrypt data, only the private key of the pair can decrypt it; if the private key is used to encrypt, the public key must be used to decrypt. This relationship allows a public key encryption scheme to do two important things. First, anyone can obtain the public key for a subject and use it to encrypt data that only the user with the private key can decrypt. Second, if a subject encrypts data using its private key, anyone can decrypt the data by using the corresponding public key. This is the foundation for digital signatures. The most common algorithm is RSA (Rivest, Shamir & Adleman).

SSL/TLS uses public key encryption to authenticate the server to the client, and optionally the client to the server. Public key cryptography is also used to establish a *session key*. The session key is used in symmetric algorithms to encrypt the bulk of the data. This combines the benefit of asymmetric encryption for authentication with the faster, less processor-intensive symmetric key encryption for the bulk data.

Hash Algorithms

During the Handshake process the hash algorithm is also agreed upon. A hash is a one-way mapping of values to a smaller set of representative values, so that the size of the resulting hash is smaller than the original message and the hash is unique to the original data. A hash is similar to a fingerprint: a fingerprint is unique to the individual and is much smaller than the original person. Hashing is used to establish data integrity during transport. Two common hash algorithms are Message Digest 5 (MD5) and Standard Hash Algorithm 1 (SHA-1). MD5 produces a 128 bit hash value and SHA-1 produces a 160 bit value.

The hash algorithm includes a value used to check the integrity of the transmitted data. This value is established using either a MAC or an HMAC. The MAC uses a mapping function to represent the message data as a fixed-length, preferably smaller, value and then hashes the message. The MAC ensures that the data has not been modified during transmission. The difference between a MAC and a digital signature is that a digital signature is also an authentication method. SSL uses a MAC.

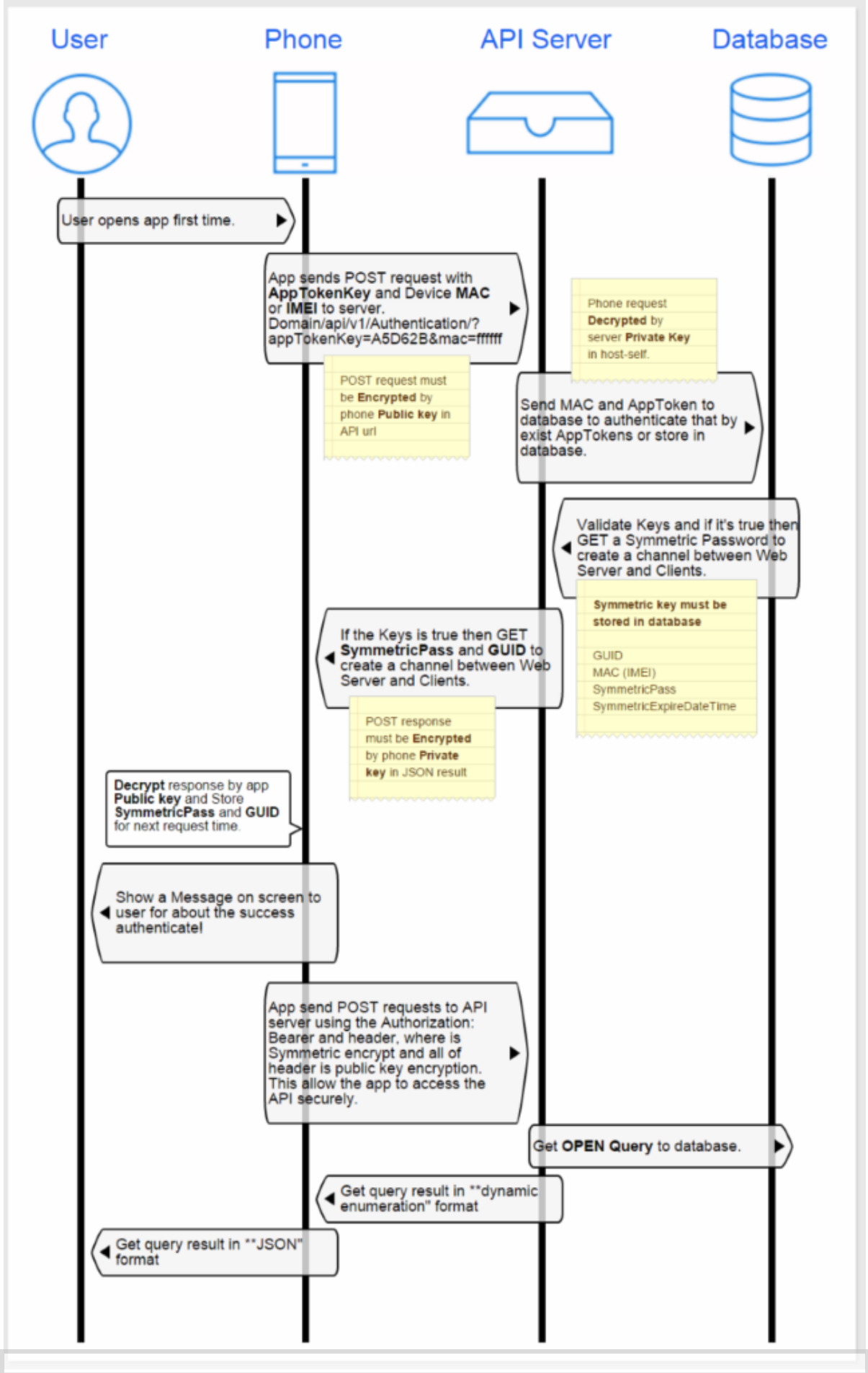
The HMAC is similar to the MAC but uses a hash algorithm in combination with a shared secret key. The shared secret key is appended to the data to be hashed. This makes the hash more secure because both parties must have the same shared secret key to prove the data is authentic. TLS uses an HMAC. For more information about HMAC see RFC 1024, "Keyed-Hashing for Message Authentication."

For more information about public key cryptography, see "Designing a Public Key Infrastructure" in *Designing and Deploying Directory and Security Services of the Microsoft® Windows® Server 2003 Deployment Kit*, (or see "Designing a Public Key Infrastructure" on the Web at http://go.microsoft.com/fwlink/?LinkID=4735).

The Record Layer

The protocol at the record layer receives and encrypts data from the application layer and delivers it to the Transport Layer. The Record Protocol takes the data, fragments it to a size appropriate to the cryptographic algorithm, optionally compresses it (or, for data received, decompresses it), applies a MAC or HMAC (HMAC is supported only by TLS) and then encrypts (or decrypts) the data using the information negotiated during the Handshake Protocol.

Reference


User
Phone
API Server
Database
User opens app first time.
App sends POST request with AppTokenKey and Device MAC or IMEI to server. Domain/api/v1/Authentication/?appTokenKey=A5D62B&mac=ffffff
POST request must be Encrypted by phone Public key in API url
Phone request Decrypted by server Private Key in host-self.
Send MAC and AppToken to database to authenticate that by exist AppTokens or store in database.
Validate Keys and if it's true then GET a Symmetric Password to create a channel between Web Server and Clients.
Symmetric key must be stored in database
GUID
MAC (IMEI)
SymmetricPass
SymmetricExpireDateTime
If the Keys is true then GET SymmetricPass and GUID to create a channel between Web Server and Clients.
POST response must be Encrypted by phone Private key in JSON result
Decrypt response by app Public key and Store SymmetricPass and GUID for next request time.
Show a Message on screen to user for about the success authenticatel
App send POST requests to API server using the Authorization: Bearer and header, where is Symmetric encrypt and all of header is public key encryption. This allow the app to access the API securely.
Get OPEN Query to database.
Get query result in **dynamic enumeration" format
Get query result in **JSON" format